

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/21 (2006.01)



[12] 发明专利说明书

专利号 ZL 200380104998.X

[45] 授权公告日 2009 年 3 月 18 日

[11] 授权公告号 CN 100470535C

[22] 申请日 2003.11.29

[21] 申请号 200380104998.X

[30] 优先权

[32] 2002.12.4 [33] US [31] 10/309,537

[86] 国际申请 PCT/US2003/038137 2003.11.29

[87] 国际公布 WO2004/051423 英 2004.6.17

[85] 进入国家阶段日期 2005.6.3

[73] 专利权人 国际商业机器公司

地址 美国纽约

[72] 发明人 威廉·鲁肯 埃蒂内·罗伊

[56] 参考文献

US6381273B1 2002.4.30

审查员 王燕_1

[74] 专利代理机构 中国国际贸易促进委员会专利
商标事务所

代理人 李德山

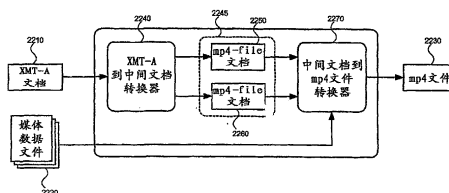
权利要求书 2 页 说明书 154 页 附图 54 页

[54] 发明名称

用于从 MPEG-4 文本表示创建 MPEG-4 中间格式的方法及系统

[57] 摘要

本发明公开了一种用于将可扩展 MPEG-4 文本(XMT)文档(2210)转换为二进制 MPEG-4(mp4)文件(2230)的方法、系统和计算机程序产品。该 XMT 文档(2210)可包括 0 或多个相关媒体数据文件(2220)。本发明包括生成表示 mp4 文件的中间文档(2245)，并根据中间文档(2245)和相关媒体数据文件(2220)创建 mp4 文件(2230)。第一转换器(2240)配置成输入 XMT 文档(2210)，并生成至少一个表示 mp4 文件的结构的中间文档(2245)。第二转换器(2270)配置成输入中间文档(2245)和任何相关的媒体文件(2220)，并生成 mp4 文件(2230)。



1. 一种用于将可扩展 MPEG-4 文本文档转换为二进制 MPEG-4 文件的方法, 该可扩展 MPEG-4 文本文档具有 0 或多个相关媒体数据文件, 该方法包括:

生成表示 MPEG-4 文件、情景描述流、对象描述符流或其组合的中间文档; 以及

根据上述中间文档和相关媒体数据文件执行二进制编码来创建 MPEG-4 文件,

其中生成中间文档进一步包括: 生成一个或多个表示 MPEG-4 文件的情景描述流、对象描述符流或其组合的附加中间文档。

2. 权利要求 1 的方法, 其中生成附加中间文档进一步包括生成表示情景描述流的 mp4-bifs 文档。

3. 权利要求 2 的方法, 其中上述 mp4-bifs 文档是与表示该 MPEG-4 文件的中间文档分开的不同文档。

4. 权利要求 1 的方法, 其中生成附加中间文档进一步包括生成表示对象描述符流的文档。

5. 权利要求 4 的方法, 其中上述表示对象描述符流的文档是与表示上述 MPEG-4 文件的中间文档分开的不同文档。

6. 权利要求 1 的方法, 其中创建 MPEG-4 文件包括确定表示每一个相关媒体数据文件中的每一个时间元素的字节数。

7. 权利要求 1 的方法, 其中创建 MPEG-4 文件包括确定每一个相关媒体数据文件的每一个时间元素的持续时间。

8. 一种用于将可扩展 MPEG-4 文本文档转换为二进制 MPEG-4 文件的系统, 该可扩展 MPEG-4 文本文档具有 0 或多个相关媒体文件, 该系统包括:

第一转换器, 其配置成输入可扩展 MPEG-4 文本文档, 并生成至少一个表示 MPEG-4 文件、情景描述流、对象描述符流或其组合的结构中间文档; 以及

第二转换器，其配置成输入上述中间文档和任何相关的媒体文件，该第二转换器进一步配置为执行二进制编码来生成MPEG-4文件，

其中上述中间文档包括：表示 MPEG-4 文件的情景描述流、对象描述符流或其组合的附加中间文档。

9. 权利要求 8 的系统，其中上述附加中间文档包括表示情景描述流的 mp4-bifs 文档。

10. 权利要求 8 的系统，其中上述附加中间文档包括表示对象描述符流的文档。

11. 权利要求 10 的系统，其中上述表示对象描述符流的文档包含在上述表示 MPEG-4 文件的中间文档内。

12. 权利要求 10 的系统，其中上述表示对象描述符流的文档是与上述表示 MPEG-4 文件的中间文档分开的不同文档。

13. 权利要求 12 的系统，其中上述表示 MPEG-4 文件的中间文档引用表示对象描述符流的文档。

用于从MPEG-4文本表示创建MPEG-4 中间格式的方法及系统

技术领域

本发明一般涉及多媒体信息的数据表示，更具体地，涉及一种称作“MPEG-4 文本表示”的多媒体信息表示的形式到另一种称作“MPEG-4 中间格式”的多媒体信息表示的形式的转换。

背景技术

计算机通用于呈现多种数字媒体，包括图像、音频样本（声音）和视频媒体、以及文本和几何形状。这些媒体类型中的每一种都可被单独呈现，或者这些媒体元素中的多种可以以称作组合多媒体表示的方式一起呈现。

对于基于各种媒体类型的信息的传播来说，创建和分发组合多媒体表示的能力是相当重要的。此外，已经创建了表现组合多媒体表示的标准化手段，以使得许多编辑者能够创建可以在诸如个人计算机、机顶盒及其他设备的各种计算机平台上再现的表示。

由运动图像专家组（MPEG）开发的两种公知的组合多媒体表示的标准化格式为可扩展 MPEG-4 文本（XMT）格式和二进制编码 MPEG-4（mp4）格式。XMT 格式很适于编辑组合多媒体表示，而 mp4 格式很适于组合多媒体表示的压缩存储和传送。因此，期望能够有效地将 XMT 格式化的表示转换为 mp4 格式化的表示。

发明内容

如下面所描述的，本发明提供了一种用于将可扩展MPEG-4文本（XMT）格式转换为二进制编码MPEG-4（mp4）格式的方法、系统和装置。本发明使用包括相对较少数量的软件且其只需要最少量资源

的有效工具来实现从XMT格式到mp4格式的组合多媒体表示转换。

这样，本发明的一个方面涉及一种用于将可扩展MPEG-4文本（XMT）文档转换为二进制MPEG-4（mp4）文件的方法。该XMT文档可能包括零或多个相关媒体数据文件。该方法包括生成表示mp4文件的中间文档，及根据该中间文档和相关媒体数据文件创建mp4文件。

本发明的另一方面在于一种用于将具有0或多个相关媒体文件的可扩展MPEG-4文本（XMT）文档转换为二进制MPEG-4（mp4）文件的系统。该系统包括第一转换器，其设置成输入XMT文档，并生成至少一个表示mp4文件的结构的中间文档。第二转换器设置成输入上述中间文档和任何相关的媒体文件，并生成mp4文件。

本发明的又一方面在于一种实现于有形介质中用于将具有0或多个相关媒体文件的可扩展MPEG-4文本（XMT）文档转换为二进制MPEG-4（mp4）文件的计算机程序产品。该计算机程序执行操作：生成表示mp4文件的中间文档，并根据上述中间文档和相关媒体数据文件创建mp4文件。

从下面对如附图中所示出的本发明各实施例的更具体描述中，本发明的上述和其他特征、用途和优点将是显而易见的。

附图说明

图1A示出本发明的一个实施例所使用的示例性XMT-A文档；

图1B示出示例性XMT-A初始对象描述符；

图2A示出示例性XMT-A par元素；

图2B示出示例性XMT-A odsm命令元素；

图3A示出示例性XMT-A插入命令；

图3B示出示例性XMT-A删除命令；

图3C示出示例性XMT-A替换命令；

图4示出示例性XMT-A BIFS节点元素；

图5A示出示例性MT-A BIFS节点；

图5B示出示例性重用的XMT-A BIFS节点；

图6A示出示例性XMT-A对象描述符;

图6B示出示例性XMT-A ES描述符;

图6C示出sdsm (BIFS) 的示例性解码器详细信息;

图7A示出本发明的一个实施例生成的示例性mp4二进制文件;

图7B示出示例性mdat原子;

图7C示出示例性字节块;

图7D示出示例性moov原子;

图8A示出示例性mp4文件idos原子;

图8B示出示例性Mp4fInitObjectDescr;

图8C示出示例性ES_ID_Inc;

图9A示出示例性trak原子;

图9B示出示例性样本表格原子;

图10A示出示例性二进制ES描述符;

图10B示出示例性解码器配置描述符;

图10C示出示例性解码器详细信息描述符;

图10D示出示例性二进制SL配置描述符;

图11A示出示例性sdsm二进制字节块;

图11B示出示例性sdsm命令帧;

图12A示出示例性BIFS插入命令;

图12B示出示例性BIFS删除命令;

图12C示出示例性BIFS替换命令;

图12D示出示例性BIFS情境替换命令;

图13A示出示例性节点插入命令;

图13B示出示例性索引值的插入命令;

图13C示出示例性路由插入命令;

图14A示出示例性节点删除命令;

图14B示出示例性索引值的删除命令;

图14C示出示例性路由删除命令;

图15A示出示例性节点替换命令;

图15B示出示例性字段替换命令；
图15C示出示例性索引值的替换命令；
图15D示出示例性路由替换命令；
图16示出示例性BIFS情境；
图17A示出示例性SFN节点（重用的）；
图17B示出示例性SFN节点（屏蔽节点）；
图17C示出示例性SFN节点（列表节点）；
图17D示出示例性MF字段（列表形式）；
图17E示出示例性MF字段（向量形式）；
图18A示出示例性路由（列表形式）；
图18B示出示例性路由（向量形式）；
图18C示出示例性路由；
图19A示出示例性odsm二进制字节块；
图19B示出示例性odsm二进制样本；
图20A示出示例性对象描述符更新命令；
图20B示出示例性对象描述符移除命令；
图21A示出示例性二进制对象描述符；
图21B示出示例性二进制EsI dRef描述符；
图22示出本发明设想的示例性XMT-A到MPEG-4中间文件转换器；
图23A示出示例性mp4file文档；
图23B示出示例性mp4fiods组件；
图24A示出示例性mdat组件；
图24B示出示例性sdsm组件；
图24C示出示例性odsm组件；
图24D示出示例性媒体文件组件；
图25A示出示例性odsm字节块组件；
图25B示出示例性odsm样本组件；
图25C示出示例性odsm命令组件；

图26A示出示例性trak组件;

图26B示出示例性stbl组件;

图27示出示例性ES_Descr;

图28A示出示例性mp4bifs文档;

图28B示出示例性mp4bifs命令帧元素;

图29A示出示例性mp4bifs bifs命令元素;

图29B示出示例性mp4bifs替换情景元素;

图30A示出示例性mp4bifs 源节点元素;

图30B示出示例性mp4bifs 条件节点元素;

图30C示出示例性mp4bifs 重用节点元素;

图31A示出示例性处理XMT-A文档流程;

图31B示出示例性处理XMT-A头流程;

图32示出示例性处理XMT-A Descr元素流程;

图33示出示例性处理XMT-A esDescr元素流程;

图34示出示例性处理XMT-A ES_Descr流程;

图35示出示例性创建mdat元素流程;

图36A示出示例性创建trak元素流程;

图36B示出示例性创建stbl元素流程;

图37示出示例性创建esds元素流程;

图38示出示例性处理BIFS配置流程;

图39A示出示例性对象表;

图39B示出示例性BIFS NodeID表;

图39C示出示例性BIFS RouteID表;

图39D示出示例性替换情景时间表;

图39E示出示例性分类对象表;

图40示出示例性处理XMT-A体元素(第1遍或第2遍)流程;

图41示出示例性处理XMT-A par元素(第1遍或第2遍)流程;

图42示出示例性处理XMT-A 命令元素(第1遍)流程;

图43示出示例性处理XMT-A par元素(第2遍)流程;

图44示出示例性处理插入命令流程;
 图45示出示例性处理删除命令流程
 图46示出示例性处理替换命令流程;
 图47示出示例性创建替换情景命令流程;
 图48示出示例性处理XMTA BIFS节点流程;
 图49示出示例性处理odsm的XML表示流程;
 图50示出示例性mp4原子结构创建流程;
 图51示出示例性mp4对象结构创建流程;
 图52示出示例性处理mdat元素流程;
 图53示出示例性处理mediaFile元素流程;
 图54示出示例性构建同步样本表流程。

标题表:

MPEG-4文本表示	1.0
MPEG-4 中间格式文件	2.0
情景描述流 (sdsm)	3.0
对象描述符流 (odsm)	4.0
mp4 文件文档	5.0
mp4-bifs 文档	6.0
xmta-mp4 转换器	7.0
根据 XMT-A 文档创建中间文档	7.1
创建 XMT-A、mp4 文件、和 mp4bifs 文档	7.1.1
为 mp4bifs 文档创建新的“bifsConfig”元素	7.1.2
为 mp4file 文档创建新的“moov”元素	7.1.3
处理 XMT-A“头”元素	7.1.4
处理 XMT-A 体元素 (第 1 遍)	7.1.5
处理 XMT-A par 元素 (第 1 遍)	7.1.5.1
处理 XMT-A 命令元素 (第 1 遍)	7.1.5.2
“处理 ODUpdate 命令-1”过程	7.1.5.3

“处理 ODRmove cmdnd”过程	7.1.5.4
为 odsm 创建编辑列表	7.1.6
处理 XMT-A 体元素（第 2 遍）	7.1.7
处理 XMT-A par 元素（第 2 遍）	7.1.7.1
处理 ODUpdate 命令-2	7.1.7.2
处理插入命令	7.1.7.3
“创建插入路由命令”过程	7.1.7.4
“创建插入节点命令”过程	7.1.7.5
处理删除命令	7.1.7.6
处理替换命令	7.1.7.7
“创建替换路由命令”过程	7.1.7.8
“创建替换情景命令”过程	7.1.7.9
“处理 XMTA BIFS 节点”过程	7.1.7.10
数据格式转换	7.1.7.11
将命令帧插入到 mp4bifs 文档中	7.1.8
将 OD 命令插入到 odsm 的 mdat 元素中	7.1.9
更新 mp4-bifs 和 mp4-file 文档的 bifsConfig	7.1.10
处理 ES 描述符	7.1.10.1
创建 trak 元素	7.1.10.2
创建初始样本表元素	7.1.10.3
处理 BIFS 配置	7.1.10.4
根据中间 XML 文档创建 mp4 二进制文件	7.2
建立输入文档和输出目的地	7.2.1
创建 mp4 原子的过程	7.2.1.1
创建 mp4 对象结构的过程	7.2.1.2
创建工作数组	7.2.2
处理“mdat”元素	7.2.3
插入媒体文件数据	7.2.3.1
插入媒体数据块	7.2.3.2

插入 odsm 数据	7.2.3.3
ObjectDescrUpdate 元素	7.2.3.4
ObjectDescrRemove元素	7.2.3.5
插入 sdsm 数据	7.2.3.6
节点插入 BIFS 命令	7.2.3.7
索引值插入 BIFS 命令	7.2.3.8
路由插入 BIFS 命令	7.2.3.9
节点删除命令	7.2.3.10
索引值删除 BIFS 命令	7.2.3.11
路由删除 BIFS 命令	7.2.3.12
节点替换BIFS命令	7.2.3.13
字段替换BIFS命令	7.2.3.14
索引值替换BIFS命令	7.2.3.15
路由替换 BIFS 命令	7.2.3.16
情景替换BIFS命令	7.2.3.17
Route结构	7.2.3.18
SFNode结构	7.2.3.19
SFField 结构	7.2.3.20
处理“moov”元素	7.2.4
处理 mp4fiods 元素	7.2.4.1
处理每一个 trak 元素	7.2.4.2
处理 mdia 元素	7.2.4.3
处理 hdlr 元素	7.2.4.4
处理 minf 元素	7.2.4.5
处理stbl元素	7.2.4.6
处理 stsc 元素	7.2.4.7
处理 stsc 元素	7.2.4.8
处理stco元素	7.2.4.9
处理 stsz 元素	7.2.4.10

处理stss元素	7.2.4.11
处理 stsd 元素	7.2.4.12
处理 ES_Descr 元素	7.2.4.13
处理BIFS DecoderConfig元素	7.2.4.14
处理JPEG DecoderConfig元素	7.2.4.15
处理 VisualConfig 元素	7.2.4.16
处理AudioConfig元素	7.2.4.17
处理媒体头元素	7.2.4.18
处理tref元素	7.2.4.19
处理edts元素	7.2.4.20
处理可选用户数据元素	7.2.5
更新 odsm 缓存大小	7.2.6

具体实施方式

本发明是一种用于将可扩展MPEG-4文本(XMT)格式(这里也称作XMT-A文档和MPEG-4文本表示)转换为二进制编码的MPEG-4(mp4)格式(也称作MPEG-4中间二进制格式)的方法、系统和计算机程序。本发明使用只需要相对较少数量的软件和仅最小的资源的新的方案来实现从XMT-A到mp4的转换。这里参照图1-54来描述本发明。

1.0 MPEG-4文本表示

MPEG-4文本表示包括表现多媒体表示的结构的“文本文件”。多媒体表示包括声音、静止图像、视频剪辑和其他元素的同步组合或序列。文本文件是包括字母、数字和标点符号的二进制编码序列的电子数据结构。通常可使用统称为“文本编辑器”的软件来解释文本文件。存在许多文本编辑器的例子，包括用于基于Windows(r)操作系统的计算机的称作“NotePad.exe”的软件和用于使用统称为UNIX的各种操作系统的计算机的“vi”。Windows是位于华盛顿的Redmond的微软

公司的注册商标。包括MPEG-4文本表示的特定类型的文本文件称为“XMT-A”文件。

在文本文件的范围内，XMT-A文件是可扩展标记语言（XML）文件的一个例子。XML文件是根据万维网联盟规定的原理（参见<http://www.w3.org/TR/2000/REC-XML-20001006>）的结构化文档。如国际标准化组织和国际电工委员会所规定的（参见2000年10月2日可于http://mpeg.telecomitalia.com/working_documents.htm和国际标准化组织（ISO），1,rue de Varembe,Case postal 56,CH-1211 Geneva 20,Switzerland获得的ISO/IEC文档14496 - 1: 2000 Amd.），XMT-A文件表现XML文件的特定实例。XMT-A规范的每一部分的完整描述篇幅都是很长的。因此，下面对XMT-A文件的描述仅限于对本发明进行描述所需规范的那一部分。读者应该参考所引用的用于XMT-A文件结构的完整描述的XMT规范文档。

像任何XML文件一样，XMT-A文件包括“元素”的层次集合。每一元素可能包括称作子元素的下级元素。此外，每一元素可能拥有一组称作“属性”的数据值。每一属性都具有名称和值。任何特定元素所拥有的特定属性名称和可能的子元素依赖于该元素的类型。每一属性值的解释依赖于相应的属性名称和拥有该属性的元素。

如图1A所示，XMT-A文件100包括两个主要部分，头元素110和体元素120。头元素110包括定义为InitialObjectDescriptor元素130的单个子元素。体元素120包括一个或多个作为子元素的“par”元素140。

InitialObjectDescriptor 具有一个属性， ObjectDescriptorID（ODID）130，且其值为字符串。如图1B中所示，该元素具有两个子元素，简档元素150和Descr元素160。简档元素150不具有子元素。简档元素150拥有包括“includeInclineProfileLevelFlag”、“sceneProfileLevelIndication”、“ODProfileLevelIndication”、“audioProfileLevelIndication”、“visualProfileLevelIndication”、及“graphicsProfileLevelIndication”若干属性。

Descr元素160可能具有若干类型的子元素。仅对本发明必要的类

型是唯一的“esDescr”元素170。esDescr170可能拥有一个或多个“ES_Descriptor”子元素180、190。ES_Descriptor元素指出“基本流”的某一属性，“基本流”是MPEG-4文档中定义的一个概念。下面简要说明ES_Descriptor元素的结构。

附属于InitialObjectDescriptor元素130的esDescr元素170拥有一个或两个ES_Descriptor元素180、190。在各种情况下，应该存在用于定义为“sdsm”或“情境描述流（scene description stream）”的基本流的ES_Descriptor 180。此外，还可能存在用于定义为“odsm”或“对象描述符流”的基本流的第二ES_Descriptor 190。用于odsm的ES_Descriptor元素190仅由依赖于音频数据、视频数据或sdsm中未指定的其他类型的媒体数据的XMT-A文件所需要。

如图2A所示，每一par元素140、200都包含一个或多个“par-child”元素210。“par-child”元素可能是另一par元素、odsm命令、或bifs命令。每一par元素还包括一个具有名称“begin”的属性。begin属性的值指出par元素内的odsm或bifs命令将要执行的时间。par元素的begin属性所确定的时间值是相对于任何父元素所暗示的时间值计算的，体元素120暗示零开始时间。

par-child元素210可能包含图2B中所示的两类odsm命令元素的实例。这些包括ObjectDescriptorUpdate元素220和ObjectDescriptorRemove元素250。ObjectDescriptorUpdate元素220包含单个OD子元素230，而OD元素230包含单个ObjectDescriptor子元素240。下面更详细地描述ObjectDescriptor元素240。ObjectDescriptorRemove元素250具有一个属性，且无子元素。ObjectDescriptorRemove元素250的属性称为“ODID”。

Par-child元素210可能包含图3中所示的三类bifs命令元素的实例。这些包括插入元素300、删除元素310和替换元素320。如图3A所示，插入元素300可能具有“xmtaBifsNode”子元素330或“ROUTE”子元素340。删除元素310不具有子元素。替换元素320可能具有“xmtaBifsNode”350子元素，“ROUTE”子元素360或“Scene”子元素

370。Scene元素具有“xmtaTopNode”子元素380。Scene元素还可能具有一个或多个ROUTE子元素390。

ROUTE元素340、390不具有子元素。ROUTE元素340、390的属性包括“fromNode”、“fromField”、“toNode”和“toField”。

术语“xmtaBifsNode元素”330表示任何所定义的大约100个BIFS节点元素中的任何一个。这些元素中的每一个具有图4中所示的一般结构400。每一个xmtaBifsNode元素400表示BIFS节点、其是在2001年8月MPEG-4系统规范ISO-IEC文档ISO/IEC 14496-1: 2001的第9章中定义的一个二进制数据结构。关于该文档的信息可在<http://mpeg.telecomitalialab.com/documents.htm>和国际标准化组织 (ISO) 1,rue de Varembe,Case postale 56,CH-1211 Geneva 20,Switzerland处获得。每一xmtaBifsNode元素400的元素标签是基于MPEG-4系统规范中定义的相应节点名称的。某些类型的xmtaBifs节点元素可能具有基于相应BIFS节点的某些属性的下级(子)元素。这些称作nodeField元素410。每一nodeField元素可能具有一个或多个包括其他xmtaBifsNode元素420的下级元素。这一排布可能循环重复,以描述BIFS节点的一个层次树。对这一层次的深度并没有限制。

每一BIFS节点具有若干称作“字段”的属性。这些字段的每一个具有已定义的字段名(串)和字段数据类型(布尔型、整型、浮点型等)。一种字段数据类型是“节点”。所有除了节点之外的字段数据类型由诸如xmtaBifsNode元素400的相同名字的属性表示。每一个具有“节点”类型的字段由xmtaBifsNode元素400的子元素410的相同名字的属性表示。xmtaBifsNode元素400的每一个子元素410可能具有作为子元素的一个或多个xmtaBifsNode元素420(xmtaBifsNode父元素400的孙子元素)。

图5中示出了XMT-A BIFS节点的XML表示。每一个XMT-A BIFS节点元素由nodeName标签500、570标识,这些标签唯一标识100多个可能类型的XMT-A BIFS节点中的一个。每一个节点元素可能是原始节点元素500或重用节点元素570。在原始节点元素500的情况下,

可选属性“DEF”510可用于提供特定节点的唯一字母数字描述。如果提供了这一属性，则将该节点分类为“可重用”。

原始XMT-A BIFS节点元素还拥有一组字段属性520，一个字段属性用于一个为NodeName类型的节点定义的并具有除“节点”或“缓冲区”之外的节点数据类型的属性字段。这些属性在图5A中表示为“字段0”、“字段2”、“字段3”和“字段5”。这些属性中每一个的实际名称由MPEG-4系统规范中为“NodeName”类型的节点定义的相应属性字段名所确定。指派给这些属性的每一个的值必须代表具有MPEG-4系统规范中定义的节点数据类型（布尔型、整型、浮点型等）的数据值。

此外，原始XMT-A BIFS节点元素500可能具有一个或多个字段值子元素530、540，其具有对应于具有“节点”或“缓冲区”数据类型的属性字段的字段名的元素标签。每一个这样的字段值元素都具有起始标签530和结束标签540。这种字段值元素530、540的例子以图5A中的元素标签<字段1>、</字段1>和<字段4>...</字段4>表示。

在具有“节点”数据类型的属性字段的情况下，字段值元素可能包含一个或多个对应于BIFS节点元素550的子元素。这种NIFS节点子元素的例子以元素标签<NodeName1.../>、<NodeName2.../>和<NodeName3.../>表示。

在具有“缓冲区”数据类型的属性字段的情况下，字段值元素可能包含一个或多个对应于BIFS命令元素300、310、320的子元素。

如果一个XMT-A BIFS节点元素包括任何字段值子元素，则该节点元素将遵循标准XML原理，以</NodeName>结束标签560结束。

将上述 XMT-A BIFS 节点元素的定义递推地应用到每一个从属 BIFS 节点元素（<NodeName1>等），允许创建节点的分层树。对于 XMT-A BIFS 节点元素的这种树的深度没有限制。

在重用节点 570 的情况下，节点元素只具有一个属性，而且没有孩子节点。这唯一的属性是“USE”属性 580，其值 590 是节点 ID 串。该作为 USE 属性的值的节点 ID 串必须与指定为具有相同 NodeName 的原始节点元素 500 的 DEF 属性 510 的节点 ID 串相匹配。

术语“xmtaTopNode”表示允许作为 Scene 元素的子元素的 xmtaBifsNode 元素的已定义子集中的一个。

如图 6A 所示，ObjectDescriptor 元素 240、600（ObjectDescriptorUpdate 元素 220）类似于上面描述的 InitialObjectDescriptor 元素 130。与 InitialObjectDescriptor 元素 130 不同，ObjectDescriptor 元素 240、600 缺少简档子元素 150。与 InitialObjectDescriptor 元素 130 相同，ObjectDescriptor 元素 240、600 具有“ObjectDescriptorID”（ODID）属性 606。典型的 ObjectDescriptor 元素 240、600 具有单个 Descr 子元素 610，Descr 元素 610 具有单个 esDesrc 子元素 620，而 esDesrc 元素 620 具有单个 ES_Descriptor 子元素 630。Descr 元素 610、esDesrc 元素 620 及 ES_Descriptor 元素 630 类似于 InitialObjectDescriptor 元素 130 的对应子元素 160、170、180、190。

ES_Descriptor 元素 180、190、630 可包含在 ObjectDescriptor 元素 600 或 InitialObjectDescriptor 元素 130 内。在每一种情况下，ES_Descriptor 元素 180、190、630 具有图 6B 中所示的结构 640。ES_Descriptor 元素 640 的“ES_ID”属性 636 的值是字母数字串，其对于每一流来说是唯一的。ES_Descriptor 元素 640 通常具有 decConfigDescr 子元素 646 和 slConfigDescr 子元素 660。如果 ES_Descriptor 元素 630、640 从属于 ObjectDescriptor 元素 600，则 ES_Descriptor 元素 630、640 还具有 StreamSource 子元素 670。如果 ES_Descriptor 元素 180、190、640 从属于 InitialObjectDescriptor 元素 130，则 ES_Descriptor 元素 180、190、640 不具有 StreamSource 子元素 670。

decConfigDescr 元素 646 具有 DecoderConfigDescriptor 子元素 650。DecoderConfigDescriptor 元素 650 具有若干属性，包括指示父 ES_Descriptor 元素 640 代表音频、视频、sdsm、odsm 还是其他类型的媒体的“流类型”和“objectTypeIndication”。DecoderConfigDescriptor 元素 650 还可具有依赖于流类型和

objectTypeIndication 的值的 **decSpecificInfo** 子元素 656。

在 **sds**m (情境描述流) **ES_Descriptor** 元素 180 的情况下, **DecoderConfigDescriptor** 元素 650 具有 **decSpecificInfo** 子元素 656。如图 6C 中所示, **decSpecificInfo** 子元素 680 元素具有 **BIFSConfig** 子元素 686。 **BIFSConfig** 元素 686 拥有若干指出怎样对 **BIFS** 节点进行编码的属性。 **BIFSConfig** 元素 686 还拥有命令流元素 690, 而命令流元素 690 拥有“大小”元素 696。

slConfig 元素 660 具有 **SLConfigDescriptor** 子元素 666。 **SLConfigDescriptor** 元素 666 具有一个称作“预定”的属性, 并且不具有子元素。“预定”属性通常具有值“2”。

StreamSource 元素 670 具有一个“url”属性, 并且不具有子元素。**url** 属性的值指出一个文件名或互联网地址 (**URL**, 统一资源定位器), 其表示包含音频数据、视频数据或其他为特定流定义实际的声音、图像等的数据的媒体数据文件的位置。对于 **sds**m (情景描述流) 或 **odsm** (对象描述流) 来说, 不存在 **StreamSource** 元素 670, 因为这些流都是由 **XMT-A** 文件定义的。

2.0 MPEG-4 中间格式文件

MPEG-4 中间格式文件是具有 2001 年 8 月的 **MPEG-4** 系统规范文档 **ISO-IEC 14496-1: 2001** 的第 13 章中定义的结构和组成的电子数据的形式。这种电子数据结构的形式是一般称作“二进制文件”的形式的一个例子, 因为它包括不限于字母、数字和标点的表示的二进制数据值序列。这允许比诸如 **XMT-A** 文件的典型文本文件所提供的要多的压缩数据结构。具有 **MPEG-4** 中间格式所定义的结构电子数据的存储形式称作“**mp4** 二进制文件”。与 **XMT-A** 文件不同, **mp4** 二进制文件不能由大多数文本编辑软件所解释。

MPEG-4 中间格式源自于苹果计算机公司在 1996 年所定义的、并可在 <http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/refFileFormat96.htm> 和 <http://developer.apple.com/techpubs/quicktime/qtdevdocs>

/PDF/QTFileFormat.pdf 处在线获得的 QuickTime (r) 文件。
QuickTime 是苹果计算机公司的注册商标。

因为它的 QuickTime (r) 继承性, MPEG-4 中间格式保留多种源自 QuickTime (r) 规范的特性。这些特性包括作为数据结构的单位的“原子”概念。每一个原子具有两部分, 头和体。头包含指出包括该原子 (包括头) 的字节数的原子大小值。头还包含指出原子的类型的原子 ID。原子的体包含由该原子承载的数据。此数据可包括从属原子。在原子的基本形式中, 原子包含由四个字节 (无符号整数) 组成的原子大小值和也包括四个字节 (字符) 的原子 ID。在 MPEG-4 规范中还定义了具有不只 4 个字节的原子大小值和原子 ID 值的原子的扩展形式。

如图 7A 中所示, mp4 二进制文件 700 包括一个或多个“mdat”原子 706 和一个“moov”原子 712。moov 原子 712 可在 mdat 原子 706 之前或在其之后。如图 7B 中所示, 每一个 mdat 原子 718 包括原子大小值 724, 在其之后有四字节原子 ID“mdat”730 和称作“块”的数据块序列 736。如图 7C 中所示, 每一个块 742 包括媒体数据“样本”序列 748。每一个样本 748 指出与单个媒体流的特定时间点相关的数据块。在单个块内的所有样本代表相同的媒体数据流。从对 mdat 原子 700 的检查中识别出各个样本 748 或块 736、742 是肯定可能的。可使用存储在 mp4 二进制文件之内的其他地方的表识别每一个样本 748 和块 736、742。

如图 7D 所示, moov 原子 758 包括原子大小值 760, 其后有四字节原子 ID“moov”766 和若干从属原子, 包括“mvhd” (moov 头) 原子 772、“iods” (初始对象描述符) 原子 778、和一个或多个“trak”原子 790。“moov”原子 712、754 包括一个用于每一个数据流的“trak”原子 790, 该数据流包括 sdsm (情景描述流) 和 odsm (对象描述流), 如果存在的话。“moov”原子 712、754 还可包括可选的“udta” (用户数据) 原子 784。“udta”原子 784 可用于在 mp4 二进制文件中嵌入诸如版权消息的可选信息。

mvhd 原子 772 包括原子大小值，其后有四字节原子 ID“mvhd”和若干数据值，包括时间日期戳、时标值、及文件持续时间值。**Mvhd** 原子的原子大小值通常为 108。时间日期戳指示该文件是何时创建的。时标值指示用于表示文件的时间值的每秒滴答数。文件持续时间值指示以时标值所指出的时间单位的呈现该文件中的材料所需的总时间。

如图 8A 中所示，**iods** 原子 800 包括原子大小值 804，其后有四字节原子 ID“iods”808、8 位版本值 812、24 位标志值 816、及 **MP4fInitObjDescr** 数据结构 820。如图 8B 所示，**MP4fInitObjDescr** 数据结构 824 包括一字节 **MP4_IOD_TAG** 值 828、随后数据块的字节数 832、10 位 **ObjectDescriptorID** 836、两个标志位 840、844、四个保留位 848、及五个简档级别指示值 852、856、860、864、868。简档级别指示值之后有一个或两个 **MPEG-4 ES_ID_Inc** 数据结构 872。一个 **ES_ID_Inc** 数据结构表示 **trak** 原子 790 对应于 **sds**m(情景描述流)的 **ES_ID**。第二个 **ES_ID_Inc** 结构只在 **odsm** 出现时才出现。如图 8C 所示，每一个 **ES_ID_Inc** 数据结构包括一字节 **ES_ID_IncTag** 值 880、随后数据的字节数（通常为 4）884、和 32 位 **ES_ID** 值 888。

如图 9A 中所示，每一个 **trak** 原子 900 包括原子大小值 903，其后有四字节原子 ID“trak”906、“**tkhd**”（track header（跟踪头））原子 910、及“**mdia**”（media（媒体））原子 912。在代表 **odsm**（object descriptor stream（对象描述流））的 **trak** 原子的情况下，**trak** 原子还包括“**tref**”（track reference（跟踪基准））原子 940。在具有定时开始的跟踪的情况下，提供一个“**edts**”（edit list（编辑列表））原子 945，以指示何时开始跟踪。**mdia** 原子 912 包括“**mdhd**”（media header（媒体头））原子 915、“**hdlr**”（handler（处理器））原子 918、“**minf**”（media information（媒体信息））原子 920、“**stbl**”（sample tables（样本表））原子 933、和媒体信息头原子 936。“**mhd**”标签表示若干媒体信息头原子类型中的任意一种，这些类型包括“**nmhd**”（用于 **sds**m 和 **odsm** 跟踪）、“**smhd**”（用于音频跟踪）、“**vmhd**”（用于视频跟踪）等。

tkhd 原子 910、mdhd 原子 915 和 hdlr 原子 918 包含若干数据值，包括跟踪 ID 号、时间日期戳、媒体时标和媒体持续时间值。每一个跟踪具有它自己的时标，其可以不同于在 mvhd 原子 772 中指出的全局时标。

如图 9B 所示，样本表原子 950 包括原子大小值 954，其后有四字节原子 ID“stbl”957 和一系列样本表原子 960、963、966、970、974、978。各样本表原子可以处于任何顺序。这些包括“stsc”（sample-to-chunk-table（样本-块表））原子 960、“stts”（time-to-sample table（时间-样本表））原子 963、“stco”（chunk offset table（块偏移表））原子 966、“stsz”（sample size table（样本大小表））原子 970、可能的 stss（sync sample table（同步样本表））原子 974、和“stds”（sample description table（样本描述表））原子 978。这些样本表原子中的每一个包含描述存储于相关 mdat 原子 706、718 中的二进制媒体数据 736、748 的属性的数据。

样本到块表（stsc 原子）960 包括原子大小值、四字节原子 ID（“stsc”）、32 位无符号整数（numStscEntries）、及样本-块数据记录序列。NumStscEntries 的值指出在样本-块数据记录的序列中的项数。每一个样本-块数据记录包括三个 32 位无符号整数，其指出起始块号、每个块的样本数、及样本描述索引。样本描述索引是对样本描述表 978 中的项的索引。每个块的样本数指出在由起始块号指定的块 736，及由下一项指定的起始块之前的所有后续块中的样本 748 的数量。

时间-样本表原子（stts 原子）963 包括原子大小值、四字节原子 ID（“stts”）、32 位无符号整数（numSttsEntries）、及时间-样本数据记录的序列。NumSttsEntries 的值指出在时间-样本数据记录的序列中的项数。每一个时间-样本数据记录包括两个 32 位无符号整数，其指出样本计数和跟踪时标单位的样本持续时间。样本计数值指出具有相应样本持续时间的连续样本 748 的数量。

块偏移表原子（stco 原子）966 包括原子大小值、四字节原子 ID

(“stco”)、32 位无符号整数 (numStcoEntries)、及块偏移值序列。NumStcoEntries 的值指出在块偏移值序列中的项数。这一序列中的每一个项包括一个 32 位无符号整数, 其指出在 mp4 文件的起始与 mdat 原子 718 内的相应块 736 的起始之间的字节数。

样本大小表原子 (stsz 原子) 970 包括原子大小值、四字节原子 ID (“stsz”)、及 32 位无符号整数 (iSampleSize), 其指出与这一 trak 原子 900 相关的所有媒体数据样本 748 的大小。如果与这一 trak 原子相关的媒体数据样本在大小上不全部相等, 则 iSampleSize 的值被指定为 0, 32 位无符号整数 (numStszEntries) 和样本大小值的序列也随之被指定为 0。NumStszEntries 的值指出样本大小值的序列中的项数。在这一序列中的每一项包括一个 32 位无符号整数, 其指出在与这一 trak 原子 900 相关的媒体数据 718 内的相应样本 748 中的字节数。

同步样本表原子 (stss 原子) 974 (如果存在的话) 包括原子大小值、四字节原子 ID (“stss”)、32 位无符号整数 (numStssEntries)、和样本索引值的序列。NumStssEntries 的值指出样本索引值的序列中的项数。这一序列中的每一项包括一个 32 位无符号整数, 其指出对于“随机访问样本”的样本索引。随机访问样本索引识别对应于与这一 trak 原子 900 相关的媒体数据中的一个点的媒体数据样本 748, 该点是媒体播放器可以开始处理媒体数据而无需考虑之前的样本的点。这一表中的样本索引值应当是单调递增的。

样本描述表原子 (stsd 原子) 978 包括原子大小值、四字节原子 ID (“stsd”)、32 位无符号整数 (numStsdEntries)、及样本描述数据记录序列。NumStsdEntries 的值指出样本描述数据记录的序列中的项数。每一个样本描述数据记录指出用于对样本-块数据记录中相应的索引所标识的媒体数据样本进行编码的手段。样本描述数据记录的序列通常具有单个项 (numStsdEntries = 1), 其指出媒体类型 (音频、视频、sdsm、odsm)、用于音频和视频样本的压缩算法等。每一个样本描述表项包含于“mp4*”原子 982 中, 其中“mp4*”是“mp4s” (用于 sdsm 和 odsm 样本)、“mp4a” (用于音频样本)、和“mp4v” (用于

视频样本)的通称。每一个“mp4”原子 982 包含“esds”(基本流描述符)原子 986,且每一个 esds 原子 986 包含 MPEG-4 基本流描述符(ES_Descr)数据结构 990。

图 10A 中示出了 MPEG-4 基本流描述符 1000 的结构。这一数据结构包括一字节标签(ES_DescrTag) 1004,其后有该数据结构的余项中的字节数的指示 1008、三个 1 位标志(streamDependenceFlag、URL_Flag 和 OCRstreamFlag) 1016、及 5 位流优先级值 1020。如果这三个标志 1016 中的任何一个是非 0,则在流优先级值 1020 之后可以有其他数据值(未示出)。这些可选数据值对于本发明来说不是所需的。

流优先级值 1020 之后有解码器配置描述符数据结构 1024 和同步层配置描述符数据结构 1028。图 10B 中示出了解码器配置描述符 1024、1032。这一数据结构包括一字节标志(DecoderConfigDescrTag) 1036,其后有该数据结构的余项中的字节数的指示 1040、一系列数据值:对象类型 1044、流类型 1048、upStream 位 1052、保留位 1056、bufferSizeDB 1060、maxBitrate1064、和 avgBitrate1068。这些值之后可以有取决于解码器详细信息数据结构 1072 的流类型和对象类型。解码器详细信息数据结构 1072 对于 sdsm 来说是所需的,但对于 odsm 来说不是所需的。大多数音频和视频媒体数据流在解码器配置描述符 1032 之内还具有解码器详细信息数据结构。

图 10C 中示出了解码器详细信息数据结构 1072、1076。这一数据结构包括一字节标志(DecoderSpecificInfoTag) 1080,其后有该数据结构中余项中的字节数的指示 1084。剩余字节依赖于对象类型和流类型。在 sdsm(情景描述流或 BIFS)的情况下,解码器详细信息 1072、1076 包括用于编码节点 ID 值的位数、和用于编码路由 ID 值的位数的指示。这些值中的每一个由 5 位无符号整数表示。

图 10D 中示出了同步层配置描述符 1028、1088 的结构。这一数据结构包括一字节标志(SLConfigDescrTag) 1090,其后有该数据结构中余项中的字节数的指示(通常为 1) 1094、及指示要用于同步层

的预定配置的单个数据字节（值 2，“预定的”）1098。

3.0 情景描述流 (sdsm)

在 XMT-A 规范或 MPEG-4 中间格式文件规范中未规定用于对音频和视频数据流的特定字节进行编码或解码的手段，因此这里未涉及怎样对这些流进行编码的细节。XMT-A 文档包含关于流描述流 (sdsm) 和对象描述流 (odsm) 的内容的详细信息。因此，每一个 XMT-A 文档密切涉及包含在 MPEG-4 中间文件内的 sdsm 和 odsm 流。这一部分描述 sdsm 数据的结构，下一部分描述 odsm 数据的结构。

与任何其他媒体数据流一样，sdsm 数据包括一个或多个块，每一个块包括一个或多个样本。如图 11A 中所示，sdsm 二进制块 1100 内的每一个样本被定义为“命令帧”1110。每一个命令帧 1110 是字节对齐的。如图 11B 中所示，每一个命令帧 1110 包括一个或多个“BIFS 命令”1120。“BIFS”代表“流的二进制格式”。每一个 BIFS 命令 1120 之后有继续位 1130、1140。如果继续位的值是 (1)，则其后有另一个 BIFS 命令。1140 在与其不同的情况下时，继续位后有足够数量的空填充位 1150 填满最后的字节。除了命令帧中的第一个 BIFS 命令之外，各个 BIFS 命令 1120 一般不是字节对齐的。

如图 12 中所示，存在四字节的 BIFS 命令：“插入”、“删除”、“替换”和“情景替换”。BIFS 插入命令 1200 包括两位插入代码(值 = “00”) 1206，其后有两位参数类型代码 1210 和插入命令数据 1216。BIFS 删除命令 1220 包括两位删除代码(值 = “01”) 1226，其后有两位参数类型代码 1230 和删除命令数据 1236。BIFS 替换命令 1240 包括两位替换代码(值 = “10”) 1244，其后有两位参数类型代码 1250 和替换命令数据 1260。BIFS 情景替换命令 1270 包括两位情景替换代码(值 = “11”) 1280，其后有 BIFS 情景数据结构 1290。

如图 13 中所示，存在三种类型的 BIFS 插入命令：(a)节点插入命令，(b)索引值插入命令，和(c)路由插入命令。插入命令的类型由参数类型值 1210 确定。节点插入命令 1300 包括两位插入代码(值 = “00”)

1304, 其后有两位参数类型代码 (值 = “01”, 类型 = 节点) 1308、节点 ID 值 1312、两位插入位置代码 1316、及 SF 节点数据结构 1324。如果插入位置代码 1316 的值为 0, 则在插入位置代码 1316 之后有 8 位位置值 1320。节点 ID 值 1312 在 BIFS 命令中的其他位置定义的一组可更新节点中指定其中的一个。在用于 sdsM 流的解码器详细信息 1072 中指出了用于对此和其他节点 ID 值进行编码的位数。SF 节点数据结构 1324 的结构在下面解释。

索引值插入命令 1328 包括两位插入代码 (值 = “00”) 1332, 其后有两位参数类型代码 (值 = “10”, 类型 = IndexedValue) 1336、节点 ID 值 1340、inField 值 1344、两位插入位置代码 1348、和字段值数据结构 1356。如果插入位置代码 1348 的值为 0, 则在插入位置代码 1348 之后有 8 位位置值 1352。节点 ID 值 1340 在 BIFS 命令的其他位置定义的一组可更新节点中指定其中的一个。在用于 sdsM 流的解码器详细信息 1072 中指出了用于对节点 ID 值 1340 进行编码的位数。inFieldID 值 1344 标识由节点 ID 1340 的值指定的 BIFS 节点的数据字段中的某一个。用于对 inField 值 1344 进行编码的位数取决于包含在 MPEG-4 系统规范中的表。

字段值数据结构的内容依赖于指定 BIFS 节点的指定数据字段的字段数据类型 (布尔型、整型、浮点型、字符串型、节点型等)。这可以如一位那样的简单, 或可以如 SF 节点数据结构那样的复杂。包含在 MPEG-4 系统规范中的表中为每一个 BIFS 节点的每一个数据字段指定了字段数据类型。

路由插入命令 1360 包括两位插入代码 (值 = “00”) 1364, 其后有两位参数类型代码 (值 = “11”, 类型 = Route) 1368、“isUpdateable”位 1372、出发节点 ID 值 1380、出发字段 ID 值 1384、到达节点 ID 值 1388、和到达字段 ID 值 1392。如果“isUpdateable”位的值是 (1), 则路由 ID 值 1376 跟随“isUpdateable”位 1372 之后。在用于 sdsM 流的解码器详细信息 1072 中指出了用于对出发节点 ID 值 1380 和到达节点 ID 值 1388 进行编码的位数。用于对出发字段 ID 值 1384 进行编

码的位数及用于对到达字段 ID 值 1392 进行编码的位数取决于包含在 MPEG-4 系统规范中的表。

如图 14 中所示, 存在三种类型的 BIFS 删除命令: (a) 节点删除命令, (b) 索引值删除命令, 及 (c) 路由删除命令。删除命令的类型由参数类型值 1230 确定。节点删除命令 1400 包括两位删除代码 (值 = “00”) 1406, 其后有两位参数类型代码 (值 = “00”, 类型 = Node) 1412 和节点 ID 值 1418。节点 ID 值 1418 在 BIFS 命令中其他位置定义的一组可更新节点中指定其中的一个。在用于 sdsM 流的解码器详细信息 1072 中指出了用于对节点 ID 值 1418 进行编码的位数。

索引值删除命令 1424 包括两位删除代码 (值 = “01”) 1430, 其后有两位参数类型代码 (值 = “10”, 类型 = IndexedValue) 1436、节点 ID 值 1442、inFieldID 值 1448、和两位删除位置值 1454。节点 ID 值 1418 在该 BIFS 命令中其他位置定义的一组可更新节点中指定其中的一个。在用于 sdsM 流的解码器详细信息 1072 中指出了用于对节点 ID 值 1418 进行编码的位数。

路由删除命令 1466 包括两位删除代码 (值 = “10”) 1472, 其后有两位参数类型代码 (值 = “11”, 类型 = Route) 1478 和路由 ID 值 1484。路由 ID 值 1484 在该 BIFS 命令的其他位置定义的一组可更新路由中指定其中的一个。在用于 sdsM 流的解码器详细信息 1072 中指出了用于对路由 ID 值 1484 进行编码的位数。

如图 15 所示, 存在四种类型的替换命令: (a) 节点替换命令, (b) 字段替换命令, (c) 索引值替换命令, 和 (d) 路由替换命令。插入命令的类型由参数类型值 1210 确定。节点替换命令 1500 包括两位替换代码 (值 = “10”) 1504, 其后有两位参数类型代码 (值 = “01”, 类型 = Node) 1508、节点 ID 值 1510、及 SF 节点数据结构 1514。节点 ID 值 1510 在该 BIFS 命令的其他位置定义的一组可更新节点中指定其中的一个。在用于 sdsM 流的解码器详细信息 1072 中指出了用于对节点 ID 值 1510 进行编码的位数。SF 节点数据结构 1514 的结构在下面解释。

字段替换命令 1520 包括两位替换代码（值 = “10”）1524，其后有两位参数类型代码（值 = “01”，类型 = Field）1528、节点 ID 值 1530、inFieldID 值 1534、及字段值数据结构 1538。节点 ID 值 1530 在该 BIFS 命令中的其他位置定义的一组可更新节点中指定其中的一个。在用于 sdsm 流的解码器详细信息 1072 中指出了用于对节点 ID 值 1530 进行编码的位数。inFieldID 值 1534 标识由节点 ID 1530 的值指定的 BIFS 节点的数据字段之一。用于对 inFieldID 值 1534 进行编码的位数取决于包含在 MPEG-4 系统规范中的表。

索引值替换命令 1540 包括两位替换代码（值 = “10”）1544，其后有两位参数类型代码（值 = “10”，类型 = 索引值）1548、节点 ID 值 1550、inFieldID 值 1554、两位替换位置代码 1558、和字段值数据结构 1564。如果替换位置代码 1558 的值为 0，则在替换位置代码 1558 之后有 8 位位置值 1560。节点 ID 值 1550 在一组由该 BIFS 命令的其他位置定义的可更新节点中指定其中的一个。在用于 sdsm 流的解码器详细信息 1072 中指出了用于对节点 ID 值 1550 进行编码的位数。inFieldID 值 1554 识别由节点 ID 1550 指定的 BIFS 节点的数据字段中的一个。用于对 inFieldID 值 1554 进行编码的位数取决于包含在 MPEG-4 系统规范中的表。

路由替换命令 1570 包括两位替换代码（值 = “10”）1574，其后有两位参数类型代码（值 = “11”，类型 = Route）1578、路由 ID 值 1580、出发节点 ID 值 1584、出发字段 ID 值 1588、到达节点 ID 值 1590、及到达字段 ID 值 1594。路由 ID 值 1580 在该 BIFS 命令的其他位置定义的一组可更新路由中指定其中的一个。在用于 sdsm 流的解码器详细信息 1072 中指出用于对路由 ID 值 1590 进行编码的位数。在用于 sdsm 流的解码器详细信息 1072 中指出了用于对出发节点 ID 值 1584 和到达节点 ID 值 1590 进行编码的位数。用于对出发字段 ID 值 1588 进行编码的位数和用于对到达字段 ID 值 1594 进行编码的位数取决于包含在 MPEG-4 系统规范中包含的表。

如图 12D 中所示，替换情景 BIFS 命令 1270 包括两位情景替换

代码（值 = “11”）1280，其后有 BIFS 情景数据结构 1290。如图 16 中所示，BIFS 情景数据结构 1600 包括 6 位保留字段 1610、两个一位标志（USENAMES 1620 和 protoList 1630）、SFTopNode 数据结构 1640、和一位标志（hasRoute）1650。如果 protoList 标志 1630 为真（1），则在 protoList 标志之后有 MPEG-4 系统规范中定义的其他数据。SFTopNode 数据结构 1640 是图 17 中所示的 SF 节点数据结构的特殊情况。如果 hasRoute 标志 1650 为真（1），则在 hasRoute 标志之后有路由数据结构 1660。图 18 中示出了路由数据结构的结构。

如图 17A、17B 和 17C 中所示，SF 节点数据结构可具有下面三种形式中的一种：（a）重用，（b）mask 节点、和（c）列表节点。所有这三种形式都以一位标志（isReused）开始。在重用 SF 节点 1700 的情况下，isReused 标志的值为“1”（真）1704，且该 SF 节点数据结构的余项包括 nodeIDref 值 1708。NodeIDref 的值 1708 应当匹配该 sdsM 数据中其他位置定义的可更新 SF 节点的节点 ID 值。

如果 isReused 标志为假（0）1712、1732，则 SF 节点类型可具有图 17B 和 17C 中所示两种形式中的一个，这取决于 maskAccess 标志位的值 1722、1742。在每一个种情况下，该 SF 节点的数据包括本地节点类型值（localNodeType）1714、1734、一位标志（isUpdateable）1716、1736、及第二个一位标志（maskAccess）1722、1742。用于对本地节点类型 1714、1734 进行编码的位数取决于 MPEG-4 系统规范中指定的表。如果 isUpdateable 标志 1716、1736 为真（1），则在 isUpdateable 标志之后有节点 ID 值 1718、1738。如果 isUpdateable 1716、1736 为真（1），且在相关 BIFS 情景数据结构 1600 中的 USERNAME 标志 1620 也为真（1），则在节点 ID 值 1718、1738 之后有以空结尾的字符串（“name”）1720、1740。

如果 maskAccess 位为真（1）1722，则 SF 节点具有“mask 节点”结构 1710。在这一情况下，如图 17B 中所示，maskAccess 位 1722 之后有 mask 位 1726 的有序序列，每一个在 MPEG-4 规范中为具有本地节点类型 1714 的值所给定的节点类型的 BIFS 节点定义的 nField 属性

字段具有一个 mask 位 1726。在这种 mask 位为真（1）的每一种情况下，mask 位之后有二进制字段值 1728，其根据由本地节点类型 1734、字段号（在 mask 位的序列内的位置）和 MPEG-4 规范种定义的表多确定的字段数据类型（整型、布尔型、字符串型、节点型等）。

如果 maskAccess 位为假（0）1742，则 SF 节点具有“列表节点”结构 1730。在这一情况下，如图 17C 中所示，maskAccess 位 1742 之后有一个或多个字段引用记录。每一个字段引用记录以一位结束标志 1744、1750 开始。如果结束标志为假（0）1744，则结束标志 1744 之后有用于为本地节点类型 1734 定义的属性字段的字段引用索引号（fieldRef）1746，且 fieldRef 值 1746 之后有根据本地节点类型 1734 和由 fieldRef 值 1746 指示的属性字段确定的字段数据类型（整型、布尔型、字符串型、节点型等）编码的二进制字段值 1748。用于对 fieldRef 值 1746 进行编码的位数由在 MPEG-4 系统规范中定义的表所确定。如果结束标志为真（1）1750，则结束字段值的列表。

每一个包括在 SF 节点结构内的属性字段值可包括单个数据值（SF 字段数据结构）或多个数据值（MF 字段数据结构）。每一个 MF 字段数据结构保留 0 或更多个 SF 字段成分。如图 17D 及 17E 中所示，根据 isList 位的值 1766、1786，存在两种形式的 MF 字段结构：列表形式 1760 和向量形式 1780。这两种形式以 isList 位 1766、1786 之后的一位保留位 1762、1782 开始。

如果 isList 位具有值（1）1766，则 MF 字段数据结构具有列表形式 1760。在这一情况下，isList 位 1766 之后有一位结束标志值的序列 1770、1772。如果结束标志位的值为“0”1770，则结束标志位之后有 SF 字段数据结构 1774。如果结束标志位的值为“1”1772，则 MF 字段数据结构结束。

如果 isList 位具有值（0）1786，则 MF 字段数据结构具有向量形式 1780。在这一情况下，isList 位 1786 之后有 5 位字段（nBits）1790，其指出后面的字段计数值（nField）1792 中的位数。这之后有 nFields SF 字段结构 1796。

每一个 SF 字段值的结构取决于与如 MPEG-4 系统规范中指定的表所指示的相应属性字段关联的特定字段数据类型。例如，布尔型字段包括单个位。在 MPEG-4 系统规范中定义并描述了其他包括整型、浮点型、字符串型、SF 节点型的情况。

BIFS 情景数据结构 1600 的最后部分是可选的路由数据结构 **1660**。如图 18A 和 18B 中所示，存在两种形式的路由数据结构：列表形式 **1800** 和向量形式 **1830**。这两种形式的路由数据结构均以一位列表标志 **1805**、**1835** 开始。如果列表标志的值为真（1）**1805**，则路由数据结构具有列表形式 **1800**。在这一情况下，列表位 **1805** 之后有一个或多个路由数据结构 **1810**，且每一个具有数据结构 **1810** 之后有一位 **moreRoutes** 标志 **1810**、**1820**。如果 **moreRoutes** 标志的值为真（1）**1810**，则之后有另一个路由数据结构 **1810**。如果 **moreRoutes** 标志的值为假（0）**1820**，则路由数据结构 **1800** 结束。

如果路由数据结构中的列表标志的值为假（0）**1835**，路由数据结构具有向量形式 **1830**。在这一情况下，列表位 **1835** 之后有 5 位 **nBits** 字段 **1840**。包含在 **nBits** 字段中的无符号整数值指出用于对后面的 **numRoutes** 值 **1845** 进行编码的位数。在后面的 **numRoutes** 值 **1845** 中编码的无符号整数指出在 **numRoutes** 值 **1845** 之后的路由数据结构 **1850** 的数量。

如图 18C 中所示，路由数据结构 **1860** 包括一位标志（**isUpdateable**）**1865**、**outNodeID** 值 **1880**、**outFieldRef** 值 **1885**、**inNodeID** 值 **1890**、及 **inFieldRef** 值 **1895**。如果 **isUpdateable** 标志 **1865** 的值为真（1），则 **isUpdateable** 标志 **1865** 之后有 **routeID** 值 **1870**。如果 **isUpdateable** 标志 **1865** 的值为真（1），且相应的 BIFS 情景数据结构 **1600** 中的 **USENAMES** 标志 **1620** 也为真（1），则路由 ID 值 **1870** 之后有以空结尾的字符串（路由名）**1875**。在用于 **sdsm** 流的解码器详细信息 **1072** 中指出了用于对 **outNodeID** 值、**inNodeID** 值、和 **routeID** 值进行编码的位数。用于对 **outFieldRef** 和 **inFieldRef** 进行编码的位数由 MPEG-4 系统规范中定义的表确定。

4.0 对象描述符流 (odsm)

与任何其他 MPEG-4 基本流相同, odsm (对象描述符流) 包含于一个或多个块 736 的序列中。如图 19 中所示, 每一个 odsm 块 1900 包括 odsm 样本 1920 的序列, 且每一个 odsm 样本 1940 包括 odsm 命令 1960 的序列。每一个 odsm 块 1900 中的 Odsm 样本 1920 的数量由用于对象描述符流的 trak 原子 790、900 中的样本-块表原子 (stsc) 的内容所确定。每一个 odsm 样本 1940 中的 odsm 命令 1960 的数量由用于对象描述符流的 trak 原子 790、900 中的样本大小表原子 (stsz) 970 所确定。

存在两种可能的 odsm 命令: 对象描述符更新命令和对象描述符移除命令。如图 20A 中所示, 对象描述符更新命令 2000 包括一字节对象描述符更新标签 2010、该命令中余项中的字节数的指示 (numBytes) 2020、及对象描述符序列 2030。对象描述符的结构概括于图 21 中。如图 20B 中所示, 对象描述符移除命令 2040 包括一字节对象描述符移除标签 2050、该命令的余项中字节数的指示 (numBytes) 2060、对象描述符 ID 值 2070、和 2 到 6 个填充位 2080。

每一个 numBytes 值 2020、2060 指出 odsm 命令的余项中的字节数。如果 numBytes 的值小于 12B, 则将 numBytes 的值编码在单个字节中。否则, numBytes 的值编码在大小字节的序列中。每一个大小字节中的高位指示其后是否有另一个大小字节。如果该高位为“1”, 则后面有另一个大小字节。每一个大小字节中的剩余 7 位指出所得到的 numBytes 的无符号整数值的 7 位。

每一个对象描述符 ID 值 2070 编码于 10 位中, 并将在对象描述符移除命令 2040 中发现的 10 位对象描述符 ID 值的序列打包成字节序列。如果对象描述符 ID 值的数量不是 4 的倍数, 则在最后的对象描述符 ID 值之后跟随两个、四个或六个空位 2080, 以填充这一命令中的最后字节。

如图 21A 中所示, 对象描述符更新命令 2000 中的对象描述符

2100 包括一字节 MP4_OD_Tag2100, 其后有 numBytes 值 2116、十位对象描述符 ID 值 2124、一位 URL_Flag 值 2132、五位保留字段 (0x1f) 2140、及 ES_Descr 数据结构或 EsIdRef 数据结构 2148。在对象描述符的这一形式中, URL_Flag2132 通常为假 (0)。NumBytes 值 2116 指出包括对象描述符的余项的字节数, 且其以为在对象描述符更新命令 2000 或对象描述符移除命令 2040 中发现的 numBytes 值指定的相同方式被编码。

图 10A 中示出了 ES_Descr 数据结构 1000 的结构。如图 21B 所示, EsIdRef 数据结构 2160 包括一字节 ES_ID_RefTag2170、numBytes 值 2180、和 16 位基本流 ID(ES_ID)值 2190。在这一情况下, numBytes 的值通常为“2”, 且该值被指定为 8 位整数。

图 22 中示出了本发明的操作。本发明 2200 根据 XMT-A 文档 2210 和相关媒体数据文件 2220, 创建 MPEG-4 中间文件 2230。输出 MPEG-4 中间文件 2230 还可称作“mp4 二进制文件”或“mp4file”。输入 XMT-A 文档 2210 可包括基于 XMT-A 规范的文本文件或这样一个文件的一组数据结构表示, 其中的 XMT-A 规范可在 ISO/IEC 14496-1:2000 Amd.2 中发现。相关媒体数据文件 2220 代表由包含在 XMT-A 文档 2210 中的流源引用 696 所标识的音频、视频和图像数据。媒体数据文件 2220 的数量可以是 0 或更多。

本发明所执行的逻辑操作 2200 可以实现为 (1) 运行在计算机系统上的计算机执行的步骤序列, 和/或 (2) 计算系统内互连的机器模块。该实现的一个问题是依赖于应用本发明的系统的性能需求的选择。因此, 构成这里所描述的本发明的实施例的逻辑操作可选地称作操作、步骤或模块。

此外, 由本发明所实现的操作可以是实现为计算机可读介质的计算机可读程序。例如, 但不限于, 计算机可读介质可以包括计算机存储介质和通信介质。计算机存储介质包括以任何用于存储诸如计算机可读指令、数据结构、程序模块或其他数据的方法或技术实现的易失性和非易失性、可移动及不可移动介质。计算机存储介质包括但不限

于 RAM、ROM、EPROM、快闪存储或其他存储技术、CD-ROM、数字通用盘 (DVD) 或其他光存储器、盒式磁带、磁带、磁盘存储器或其他磁存储设备、或任何其他可用于存储所需信息并能由计算机访问的介质。通信介质通常在调制的数据信号 (诸如载波或其他传输机制) 中实现计算机可读指令、数据结构、程序模块或其他数据, 并包括任何信息呈递介质。术语“调制的数据信号”意指具有信号特性集中的一种或多种的信号、或以对该信号中的信息进行编码的方式改变的信号。例如, 但并不限于, 通信介质包括有线介质 (诸如有线网络或直接有线连接), 或无限介质 (诸如声音、RF、红外线和其他无线介质)。上述的任何的组合也应包括在计算机可读介质的范围之内。

如图 22 中所示, 创建 mp4file 2230 的过程以两步来实现。在第一个步骤中, XMT-A 到中间文档转换器 2240 解释输入 XMT-A 文档 2210, 并创建一个或多个表示 MPEG-4 中间文件 2230 的中间文档 2245。在本发明的一个实施例中, 由中间文档转换器创建一对中间文档 2250 和 2260。中间文档包括 mp4-file 文档 2250 和 mp4-bifs 文档 2260。在第二步骤中, 中间文档到 mp4file 转换器 2270 根据与该 XMT-A 文档 2210 相关的 mp4-文件文档 2260、mp4-bifs 文档 2260 和媒体数据文件 2220 (如果存在的话) 生成输出 mp4 文件 2230。

将这一过程 2200 分为两步的一个原因在于, 尽管输出 mp4 文件 2230 表示由输入 XMT-A 文档 2210 和媒体数据文件 2220 所表示的相同信息, 但输出 mp4 文件 2230 的组织 and 结构很大程度上不同于输入 XMT-A 文档的组织 and 结构。例如, mp4 文件的结构密切相关于 Quicktime (r) 媒体数据文件的结构, 但 XMT-A 的结构不具有一个 Quicktime (r) 媒体数据文件的特性。XMT-A 文档包含情景描述流 (sdsm) 和对象描述符流 (odsm) 描述, 但这些可以任何顺序混合。Mp4 文件还包含 sdsm 和 odsm 的描述, 但这些中的每一个被表示为临时排序样本的分离流。

因为 mp4 文件 2230 的结构和组织如此不同于 XMT-A 文档 2210 的结构和组织, 所以将根据 XMT-A 文档 2210 创建 mp4 文件 2230 的

过程分为至少两步是有利的，即（a）重组，和（b）二进制编码。在第一个步骤中，包含在 XMT-A 文档 2210 中的信息组织为反映 mp4 文件的结构和组织的形式。在第二个步骤中，通过遍历所得到的组织的信息而同时执行对该信息的二进制编码来创建输出 mp4 文件 2230，其中的信息是以输出该 mp4 文件 2230 所需的顺序组织的。以这一方式，第一步可以在不考虑 mp4 文件的二进制编码的需求的情况下进行，而第二步可以在不考虑 XMT-A 文档的结构的情况下进行。

为了实现将根据 XMT-A 文档 2210 创建 mp4 文件 2230 的过程分为这两步的目的，定义新的结构化文档是必须的，该新结构化文档表示（a）mp4 文件的结构和组织，（b）如在 mp4 文件中表示的流描述流（sdsm）的结构和组织，及（c）如在 mp4 文件 2230 中表示的对象描述符流（odsm）的结构和组织。这可通过定义三种新类型的结构化文档来实现，这三种结构化文档一个表示 mp4 文件、一个表示 sdsm、和一个表示 odsm。在这些当中，表示 mp4 文件和 sdsm 所需的结构化文档是相对复杂的，但表示 odsm 所需的结构化文档是非常简单的。因此，将 odsm 的描述并入用于表示 mp4 文件的结构化文档中是有利的。因此，在本发明的一个实施例中，引入了新的结构化文档，一个用于 mp4 文件和 odsm，一个用于 sdsm。这两个结构化文档被标识为 mp4-文件文档 2250 和 mp4-bifs 文档 2260。这些结构化文档统称为中间文档。

应该指出，为方便起见，选择使用了两种类型的结构化文档。通过定义三种类型的结构化文档（仅 mp4 文件、仅 odsm 及仅 sdsm）、或所有三种类型的信息可以合并为单个复合结构化文档，可以实现该相同目的。

在本发明的一个实施例中，被创建来表示重组的信息的特定类型结构化文档是基于“XML”（可扩展标记语言）技术之上的。其是有利的，因为：

- （a）XMT-A 文档的定义是基于 XML 技术的；
- （b）XML 技术提供表示结构化文档的标准化手段；以及

(c) 存在标准化软件工具，用于作用于基于 XML 技术的结构化文档。

这样，在本发明的一个实施例中，对包含在 XMT-A 文件中的信息进行重组的过程被简化为 XML 到 XML 的转换。此外，作用于 XML 文件的标准化软件使得对输入 XMT-A 文档以及中间文档进行管理，而无需开发新的执行该相同功能的专用软件成为可能。

尽管此实施例是基于对表示中间文档的 XML 技术的使用之上，但创建使用其他类型的结构化文档的本发明可选实施例是可能的。

下面的资料描述 (1) mp4-文件文档 2250 的结构，(2) mp4-bifs 文档 2260 的结构，(3) XMT-A 到中间文档转换器 2240 的操作，和 (4) 中间文档到 mp4 文件转换器 2270 的操作。

5.0 mp4 文件文档

如图 23A 中所示，mp4-文件文档 2300 的结构相当类似于 mp4 二进制文件 700 的结构。Mp4-文件文档 2300 包含一组一个或多个媒体数据 (mdat) 元素 2310 和单个 moov 元素 2320。Moov 元素 2320 包括 mp4fiods (mp4 文件初始对象描述符) 元素 2330 和一个或多个 trak 元素 2350。Moov 元素 2320 还可包括可选的用户数据 (udta) 元素 2340。Udta 元素 2340 可用于包括诸如版权公告的信息。Mvhd 原子 772 可由 moov 元素 2320 的属性所表示。

如图 23B 中所示，mp4fiods 元素 2360 拥有 objectDescriptorID 属性 2370。Mp4fiods 元素 2360 还拥有若干图 23B 中未示出的其他属性。这些其他属性包括布尔型属性“includeInlineProfilesFlag”、和整型属性“sceneProfileLevelIndication”、“ODProfileLevelIndication”、“audioProfileLevelIndication”、“visualProfileLevelIndication”、和“graphicsProfileLevelIndication”。Mp4fiods 元素 2360 还包括一个或多个 EsIdInc 元素 2380。每一个 EsIdInc 元素 2380 拥有 trackID 属性 2390，其匹配相关 trak 元素 2340 的 trackID 属性。

如图 24A 中所示，每一个 mdat 元素 240 可包含下面元素中的一

个或多个：**sds**m 元素 2410、**odsm** 元素 2420、和 **mediaFile** 元素 2430。这些元素中的每一个拥有唯一的“**trackID**”属性，其匹配相关 **trak** 元素 2340 的 **trackID** 属性。每一个 **mediaFile** 元素 2430 具有“**name**”属性，其指出包含相关媒体数据（音频数据、视频数据等）的外部二进制文件的文件名。每一个 **sds**m 元素 2410 具有“**xmlFile**”属性，其指出表示相关 **mp4-bifs** 文档 2260 的 XML 文件的名称。在一个实施例中，对表示 **mp4**-文件文档和/或 **mp4-bifs** 文档的 XML 文件的创建对于诊断目的是有用的，但这种文件对于本发明的操作不是所需的。

如图 24B 和 24D 所示，每一个 **sds**m 元素 2440 和每一个 **mediaFile** 元素 280 包括一个或多个块元素 2450、2490。每一个块元素 2450、2490 拥有“**size**”属性，其指示相关的二进制数据块中的字节数，如果已知的话。每一个块元素 2450、2490 还拥有“**offset**”属性，其指示二进制 **sds**m 数据或媒体数据文件的开始与该二进制 **sds**m 数据或媒体数据文件内的当前块的数据开始之间的字节数，如果已知的话。其他描述情景描述流（**sds**m）的信息包含在 **mp4-bifs** 文档内。

如图 24C 所示，每一个 **odsm** 元素 2460 包含一个或多个 **odsmChunk** 2470 元素。每一个 **odsmChunk** 元素 2470 拥有“**size**”属性，其指示对象描述符流中的相关部分的字节数，如果已知的话。每一个 **odsmChunk** 元素 2470 还拥有“**offset**”属性，其指示在相关对象描述符流的开始与该流的当前块的数据开始之间的字节数，如果已知的话。

如图 25A 所示，每一个 **odsmChunk** 元素 2500 包含一个或多个 **odsmSample** 元素 2510。如图 25B 所示，每一个 **odsmSample** 元素 2520 包含一个或多个 **odsm** 命令元素 2530。如图 25C 所示，每一个 **odsm** 命令元素可以是 **ObjectDescrUpdate** 元素 2540 或 **ObjectDescrRemove** 元素 2570。每一个 **ObjectDescrUpdate** 元素 2540 包含 **ObjectDescriptor** 元素 2550，包含在 **ObjectDescrUpdate** 元素 2540 中的 **ObjectDescriptor** 元素 2550 包含 **EsIdRef** 元素 2560。

每一个 **odsmSample** 元素 2510、2520 拥有“**time**”属性，其以秒指出当包含在 **odsmSample** 元素 2510、2520 中的命令要执行时的时间。

每一个 **ObjectDescriptor** 元素 2550 和每一个 **ObjectDescrRemove** 元素 2570 拥有“**ODID**”属性，其示出数字的对象描述符 ID。每一个 **EsIdRef** 元素 2560 拥有“**EsId**”属性，其指出数字的基本流 ID。

如图 26A 所示，**Trak** 元素 2350、2600 的结构相当类似于 **mp4** 文件 700 内的 **trak** 原子 790、900 的结构。每一个 **trak** 元素 2600 包含 **mdia** 元素 2604。**Trak** 元素 2600 还包含 **tref** (track reference (跟踪基准)) 元素 2636 和/或 **edts** (edit list (编辑列表)) 元素 2644。不存在类似于 **mp4** 文件中的 **tkhd** 原子 910 的 **tkhd** 元素。而是，包含在 **tkhd** 原子 910 中的属性表示为 **trak** 元素 2600 的属性。

Mdia 元素 2604 包含 **hdlr** 元素 2608 和 **minf** 元素 2612。**Mdhd** 原子 915 的属性表示为 **mdia** 元素 2604 的属性。**Minf** 元素 2612 包含 **dinf** 元素 2616、**stbl** 元素 2628、和媒体头元素 2632。媒体头元素 (“***mhd**”) 2632 可具有一种或几种依赖于相关数据流中的数据类型形式。与 **sds**m 或 **ods**m 相关的 **trak** 元素中的媒体头元素 2632 由 “**nmhd**”元素代表。与音频流相关的 **trak** 元素中的媒体头元素 2632 由 “**smhd**”元素代表，与视频流相关的 **trak** 元素内的媒体头元素 2632 由 “**vmhd**”元素代表。

如图 26B 所示，**stbl** (sample tables (样本表)) 元素 2628、2652 包含 **stsc** (sample-to-chunk-table (样本-块表)) 元素 2656、**stts** (time-to-sample table (时间-样本表)) 2660、**stco** (chunk offset table (块偏移表)) 元素 2664、**stsz** (sample size table (样本大小表)) 元素 2668、和 **stsd** (sample description table (样本描述表)) 元素 2676。**Stbl** 元素 2664 还可包括依赖于流或媒体类型的 **stss** (sync sample table (同步样本表)) 元素 2672。**Stsd** 元素 2676 可包含图 26B 中表示为 “**mp4***元素”2680 的若干类型从属元素中的一种。在包含在与 **sds**m 或 **ods**m 流相关的 **trak** 元素 2600 内的 **stsd** 元素 2676 的情况下，**stsd** 元素 2680 包含 “**mp4s**”元素。在包含在与音频流相关的 **trak** 元素 2600 内的 **stsd** 元素 2680 的情况下，**stsd** 元素 2680 包含 “**mp4a**”元素。在包含在与视频流相关的 **trak** 元素 2600 内的 **stsd** 元素 2680 的情况下，

stds 元素 2680 包含“mp4v”元素。在每一个情况下，“mp4*”元素 2680 都包含 esds 元素 2684，而 esds 元素 2684 包含 ES_Descr 元素 2688。

如图 27 所示，ES_Descr 元素 2700 包含解码器配置描述符元素 2710 和 SLConfigDescriptor 元素 2760。解码器配置描述符元素 2710 可包含若干类型的解码器详细信息元素中的一种，包括 BIFS_DecoderConfig 元素 2720、JPEG 解码器配置 2730、视频配置 2740、或音频配置 2750。这各种解码器详细信息中的每一种都表示包含在二进制解码器配置描述符结构 1032 中的解码器详细信息数据结构 1072 的一种形式。二进制 ES_Descr 结构 1000、解码器配置描述符结构 1032、SLConfigDescriptor 结构 1088、和解码器详细信息结构 1076 的属性由 mp4-文件文档 2300 的相应元素 2700、2710、2760、2720、2730、2740、2750 的属性代表。

6.0 mp4-bifs 文档

如图 28A 所示，mp4-bifs 文档 2800 包含单个 bifsConfig 元素 2810，其后有一个或多个命令帧元素 2820 的序列。如图 28B 所示，每一个命令帧元素 2830 包含一个或多个 mp4bifs bifs 命令元素 2840。每一个命令帧元素 2820、2830 拥有属性“time”，其以秒指出当包含在命令帧元素中的命令要执行时的时间。

每一个 mp4bifs bifs 命令元素 2840 代表 11 种可能的 MPEG-4 BIFS 命令：插入节点、插入索引值、插入路由、删除节点、删除索引值、删除路由、替换节点、替换字段、替换索引值、替换路由和替换情景中的一种。如图 29A 所示，mp4bifs bifs 命令元素 2910 可包含一个或多个 mp4bifs 节点元素 2920。在 11 种类型的 bifs 命令元素中，插入节点、插入索引值、替换节点、替换字段、替换索引值、和替换情景可包括从属 mp4bifs 节点元素 2920。

如图 29B 所示，替换情景 bifs 命令元素 2930 可包括仅单个从属 mp4bifs 节点元素，则这肯定是“TopNode”元素 2940。TopNode 元素 2940 对应于 MPEG-4 BIFS 节点的特定子集的一个成员。这个子集被

定义于 MPEG-4 系统规范中。此外，替换情景 bifs 命令元素 2930 还可包括从属“Routes”元素 2950，该“Routes”元素 2950 可包含一个或多个从属“Route”元素 2960。Mp4bifsRoute 元素 2960 具有属性“routeId”、“arrivalNodeId”、“arrivalField”、“departureNodeId”、“departureField”。

除了可能的从属 mp4bifs 节点元素之外，每一种类型的 mp4bifs bifs 命令元素拥有下列属性值：

1. 插入节点：“父 Id”、“插入位置”和“位置”
2. 插入索引值：“节点 Id”、“inFieldName”、“插入位置”“位置”和“值”
3. 插入路由：“路由 Id”、“departureNode”、“departureField”、“arrivalNode”和“arrivalField”
4. 删除节点：“节点 Id”
5. 删除索引值：“节点 Id”、“inFieldName”、“删除位置”和“位置”
6. 删除路由：“routeId”
7. 替换节点：“父 Id”
8. 替换字段：“节点 Id”、“infieldName”和“值”
9. 替换索引值：“节点 Id”、“infieldName”、“插入位置”、“位置”和“值”
10. 替换路由：“路由 Id”、“departureNode”、“departureField”、“arrivalNode”和“arrivalField”
11. 替换情景：“USENAMES”（布尔值）

对于 bifs 命令元素插入索引值、替换字段、和替换索引值，如果由“infieldName”属性指定的属性字段具有节点数据类型“Node”（按照 MPEG-4 规范），则该元素将包含一个或多个从属 mp4bifs 节点元素 2920，且“值”属性将包含与每一个从属节点元素相关的节点名的列表。

Mp4bifs 节点元素 2920 代表多种类型的 MPEG-4 BIFS 节点数据结构中的一种。在 MPEG-4 系统规范中定义了 100 多种不同的 BIFS

节点类型。每一种类型的 MPEG-4 BIFS 节点具有特定的节点名和属性字段集。

存在两种基本类型的 mp4bifs 节点元素：原始节点元素和重用节点元素。如图 30A 所示，mp4bifs 原始节点元素 3000 由对应于 MPEG-4 系统规范中定义的其中一种 BIFS 节点的节点名属性的“NodeName”所标识。

Mp4bifs 原始节点元素 3000 可具有可选的节点 Id 属性 3010。如果对于节点 Id 属性 3010 指定了一个值，则节点元素 3000 被分类为“重用节点”。节点 ID 属性的值 3010（如果指定了的话）是 1 到当前情景中定义的重用节点数的范围内的一个整数。如果对于节点 ID 属性 3010 已指定了一个值，且相关替换情景命令的“USENAMES”属性的值为“真”，则该节点元素将还具有“名称”属性 3016。

除了节点 ID 3010 和名称 3016 属性之外，每一个原始节点元素具有多个属性字段属性 3020。每一个属性字段属性 3020 对应于 MPEG-4 系统规范中为特定节点元素的 NodeName 所标识的节点类型定义的属性字段中的一种。每一种属性字段具有已定义的字段数据类型，诸如布尔型、整型、浮点型等。可能的字段数据类型的集合包括“SFNode”和“MFNode”。如果特定原始节点元素的 NodeName 对应于 MPEG-4 BIFS 节点具有字段数据类型“SFNode”和“MFNode”的属性字段或字段，则该节点元可拥有一个或多个从属节点元素 3030。如果是这样，则相应属性字段属性的值包括与该属性字段相关的每一个从属节点元素的 NodeName 串。

如果例如具有 NodeName“Group”的特定 mp4bifs 节点元素拥有具有与“children”属性相关的 NodeName“Transform2D”、“Valuator”和“TimeSensor”的从属 mp4bifs 节点元素，则“children”属性的值将是“Transform2D Valuator TimeSensor”。

在条件 BIFS 节点的特殊情况下，属性字段中的一个具有属性字段名“buffer”，“buffer”属性字段的字段数据类型是“command buffer”，且“buffer”属性字段的值包括一个或多个 BIFS 命令。在这一

情况下，相应 mp4bifs 节点元素 3040 的 NodeName 为“conditional”。条件节点元素 3040 的节点 ID 属性 3050 和名字属性 3056 的值可以指定为用于任意其他的 mp4bifs 原始节点元素 3000。代替从属节点元素 3030，条件节点元素拥有一个或多个从属 bifs 命令元素 3070，且“buffer”属性的值包括从属 bifs 命令元素 3070 的命令名的有序列表。

如果例如特定条件节点元素拥有其后有从属删除节点 bifs 命令元素的从属插入路由 bifs 命令元素，则“buffer”属性的值将为“插入路由 删除节点”。

原始节点元素拥有从属节点元素或 bifs 命令元素的能力可对 BIS 命令和节点元素的层次集合循环重复。

如图 30C 所示，重用的节点元素 3080 具有 NodeName“重用节点”。重用节点元素 3080 不具有从属元素。重用节点元素 3080 具有单个属性，称为“nodeRef”3090。NodeRef 属性 3090 的值必须与可重用原始节点元素 3000、3040 中的一个的节点 ID 属性 3010、3050 的值匹配。

7.0 xmta-mp4 转换器

如上面所述，本发明的一个实施例根据 XMT-A 文档 2210 和一组 0 个或多个二进制媒体数据文件 2220，创建 MPEG-4 中间二进制文件（“mp4 file”）2230。

这一过程包括两个主要步骤：

- a. 第一步骤 2240，其中根据 XMT-A 文档 2210 创建一对中间文档 2250、2260，以及
- b. 第二步骤 2270，其中根据中间文档 2250、2260 和在 XMT-A 文档 2210 中指定的任何二进制媒体数据文件 2220，创建 MPEG-4 中间二进制文件 2230。

媒体数据文件 2220 仅用于第二步骤中。第一步骤 2240 可使用媒体数据文件的名称，但这些媒体数据文件自身不用于第一步骤 2240 中。

这些主要步骤示于图 22 中，下面对其进行详细描述。

7.1 根据 XMT-A 文档创建中间文档

图 31A 中概括了创建中间文档 2250 的过程 2240。设想过程 2240 可以硬件、如见和硬件和软件的组合来实现，以满足特定应用的需要。硬件实现趋向于工作更快速，而软件实现通常在制造上是低成本的。由过程 2240 实现的逻辑操作可是实现为（1）运行于计算机系统上的计算机实现步骤的序列和、或（2）计算机系统内互连的机器模块。该实现是依赖于应用本发明的系统的性能需求的选择问题。因此，这里所描述的构成本发明实施例的逻辑操作可选地称作操作、步骤、或模块。

7.1.1 创建 XMT-A、mp4 文件、和 mp4bifs 文档

过程 2240 开始于操作 3100。可通过读取和解释表示 XMT-A 文档的 XML 文件来创建 XMT-A 文档 100。可使用标准 XML 手段读取这样的文件，并产生表示包含在该 XML 文件内的所有信息的 XMT-A 文档。这是一个标准 XML 操作，且不是本发明专有的。可选地，可将先前从其他诸如基于 XMTA 的 MPEG-4 编辑工具的手段获得的 XMT-A 文档提供为对实现下面步骤的软件或其他手段的依据。

使用标准 XML 手段来创建空 mp4file 文档 2300 和空 mp4bifs 文档 2800。这些文档中的每一个包含顶级元素，其不具有子元素，且除了可能的默认属性之外无其他属性。可向字符串量“sdsmFileName”指派一个值。该值仅在中间文档要保存到外部文本文件中时使用。可从输入 XMT-A 文档的名字中提取出一个适当值，如果存在的话。否则，将值“mp4bifs.xml”指派给量“sdsmFileName”。在操作 3100 完成之后，控制流传递给操作 3106。

7.1.2 为 mp4bifs 文档创建新的“bifsConfig”元素

在操作 3106，为 mp4bifs 文档创建新的“bifsConfig”元素。使用

标准 XML 手段创建空的“bifsConfig”元素 2810，并将其插入 mp4bifs 文档 2800 的顶级元素中。使用标准 XML 手段向该“bifsConfig”元素的下列属性指派值。向“nodeIdBits”属性指派值“0”。向“routeIdBits”属性指派值“0”。向“protoIdBits”属性指派值“0”。向“commandStream”属性指派值“真”。向“pixelMetric”属性指派值“真”。向“pixelHeight”属性指派值“0”。向“pixelWidth”属性指派值“0”。向“useBifsV2Config”属性指派值“假”。向“use3DmeshCoding”属性指派值“假”。向“usePredictiveMFField”属性指派值“假”。这些仅是临时值，其在后面将由从 XMT-A 文档中获取出的值替换。在操作 3106 完成之后，控制流传递给操作 3110。

7.1.3 为 mp4file 文档创建新的“moov”元素

在操作 3110，为该 mp4file 文档创建新的“moov”元素。使用标准 XML 手段创建空“moov”元素 2320，并将其插入到该 mp4file 文档 2300 的顶级元素中。向量“nextTrackID”和量“nextEsId”指派值“1”。使用标准 XML 手段向该“moov”元素的下列属性指派值。

a.“creationTime”和“modificationTime”：这些属性的值将指出自从 1904 年 1 月 1 日以来所过去的秒数，表示为无符号整数。优选地，这些值应由当前时钟时间所确定。如果当前时钟时间不能获得，则可将这些值设置为任意值。这里所指出的实际值仅是参考性的，对所得到的 MPEG-4 二进制文件的处理不会有影响。

b.“timeScale”：该属性指出用于度量该 MPEG-4 二进制文件内的时间的每秒时钟滴答数。值 1000 暗示时间将以微秒来规定。

c.“duration”：向该属性指派值 0。该值将在后面更新。

d.“nextTrackID”：向该属性指派量“nextTrackID”的值。该属性的值将在后面每一个新的“trak”元素被添加到该文档中且量“nextTrackID”的值增加时被更新。

此时，添加可选的“udta”（用户数据）元素 2340 是可能的。这可用于将版权消息插入在后面的主步骤 2270 中创建的 MPEG-4 二进

制文件中。其他诸如作者标识符的信息也可在此时指出。在操作 3110 完成之后，控制流传递给操作 3116。

7.1.4 处理 XMT-A“头”元素

在操作 3116, 对 XMT-A 头进行处理。该步骤在“moov”元素 2320 中创建“mp4fiods”元素 2330、2360。将为情景描述流 (sds) 创建“mdat”元素 2310 和“trak”元素 2350。如果存在任何对象，则将为对象描述符流 (ods) 创建另一个“mdat”元素 2310 和另一个“trak”元素 2350。

可使用标准 XML 手段获得 XMT-A 文档 100 中的“头”元素 110。然后，图 31B 中所示的步骤用于对该 XMT-A“头”元素进行处理。该 XMT-A“头”处理子操作开始于 InitialObjectDescriptor 处理操作 3150。

然后，在操作 3150，可使用标准 XML 手段获得从属于 XMT-A“头”元素 110 的 XMT-A“InitialObjectDescriptor”元素 130。“InitialObjectDescriptor”元素 130 可具有称为“objectDescriptorID”的属性，其具有“IODID:nnn”形式的值，其中子串“nnn”代表正整数。可选地，该元素可具有称为“binaryID”具有值“nnn”的属性。在每一种情况下，由“nnn”表示的值都将指派给量“ODID”。如果这些属性中的每一个都不存在，则向量“ODID”指派值“1”。在操作 3150 完成之后，控制流传递给操作 3160。

在操作 3160，创建新的“mp4fiods”元素 2330、2360，并将其插入到 mp4file 文档 2300 中的“moov”元素 2320 中。然后，将从 XMT-A 文档 100 中的“InitialObjectDescriptor”元素 130 中提取出的量“ODID”的值指派给“mp4fiods”元素 2330、2360 的“objectDescriptorID”属性 2370。在操作 3160 完成之后，控制流传递给操作 3170。

在操作 3170，使用标准 XML 手段获得从属于“InitialObjectDescriptor”元素 130 的“Profiles”元素 150。然后，根据

“Profiles”元素 150 的同名属性的值设置“includeInlineProfiles”、“sceneProfileLevelIndication”、“ODProfileLevelIndication”、“audioProfileLevelIndication”、“visualProfileLevelIndication”和“graphicsProfileLevelIndication”属性的值。

“mp4fiods”元素 2360 的“includeInlineProfiles”属性的值必须为“真”或“假”。如果“Profiles”元素 150 中的“includeInlineProfiles”属性的值为“真”或“假”，则将同一值指派给“mp4fiods”元素 2360 的“includeInlineProfiles”属性。否则，将值“假”指派给“mp4fiods”元素 2360 的“includeInlineProfiles”属性。

“mp4fiods”元素 2360 的五个简档和级别表示属性的值必须表示从-255 到+255 的数字值。“Profiles”元素 150 的相应属性可能具有相等的值，或它们可能具有由字母数字串指定的值。如果“Profiles”元素 150 的任何简档和级别表示属性具有字符串值“none”，则向“mp4fiods”元素 2360 的同名属性指派数字值“-1”或“255”。如果“Profiles”元素 150 的任何简档和级别表示属性具有字符串值“unspecified”，则向“mp4fiods”元素 2360 的同名属性指派数字值“-2”或“254”。在 MPEG-4 系统规范中包含的表中定义了“Profiles”元素 150 的这些属性的其他字母数字值，并使其与数字值相关联。如果 Profiles 元素 150 的任何简档和级别表示属性的值与 MPEG-4 系统规范中定义的字母数字简档和级别字符串中的一个匹配，则将在这些表中指定的相应的数字值指派给“mp4fiods”元素 2360 的相应属性。如果 Profiles 元素 150 的任何简档和级别表示属性的值包括不与 MPEG-4 系统规范中包含的简档和级别值的表中的任何项匹配的字母数字串，则向“mp4fiods”元素 2360 的相应属性指派值“-2”或“254”。在操作 3170 完成之后，控制流传递给操作 3176。

在操作 3176，使用标准 XML 手段获得从属于“InitialObjectDescriptor”元素 130 的“Descr”元素 160。执行过程“处理 Descr 元素”3176，以识别从属于该 Descr 元素 160 的 esDescr 元素 170。执行过程“处理 esDescr 元素”3180 以识别每一个从属于 esDescr

元素的 ES_Descriptor 元素 180、190。对于每一个从属于 esDescr 元素 170 的 ES_Descriptor 元素执行过程“处理 ES_Descriptor”3186、3190。

如图 32 所示,过程“处理 Descr 元素”3176 开始于向索引“i”3200 指派值“0”。在该过程中,将索引“i”的值与量“numDescrChildren”3210 的值进行比较。量 numDescrChildren 的值指示 Descr 元素 160 所拥有的从属元素的数量。如果索引“i”的值等于 numDescrChildren 的值,则过程“处理 Descr 元素”3176 完成 3260,过程“处理 XMT-A 头”3116 完成,过程“处理 XMT-A 文档”前进到第 1 遍 XMT-A 体处理操作 3120,下面进行描述。

如果索引“i”的值不等于 numDescrChildren,则使用标准 XML 手段获得从属于 Descr 元素 160 的第 i 个元素,且所得到的从属元素标识为“DescrChildren”3220。如果 DescrChildre 的元素名为“esDescr”,则执行“处理 esDescr 元素”3240。随后,索引“i”的值递增 1 3250,并重复“i”的值与 numDescrChildren 3210 的值的比较。期望每一个 Descr 元素生成单个从属 esDescr 元素。

如图 33 所示,过程“处理 esDescr 元素”3180 开始于向索引“i”3300 指派值“0”。该索引“i”不同于在过程“处理 Descr 元素”3180 中定义的模拟量。在这一过程中,将索引“i”的值与量“numEsDescrChildren”3310 的值进行比较。量 numEsDescrChildren 的值指示 esDescr 元素 170 所拥有的从属元素的数量。如果索引“i”的值等于 numEsDescrChildren,则过程“处理 esDescr 元素”3180 完成,且过程“处理 Descr 元素”3176 通过递增该过程中的索引“i”3250 的值而继续。

如果索引“i”的值不等于 numDescrChildren 的值,则使用标准 XML 手段获得从属于 esDescr 元素 170 的第 i 个元素,并将所得到的从属元素标识为“esDescrChildren”3320。如果 esDescrChildren 的元素名为“ES_Descriptor”,则执行过程“处理 ES_Descriptor”3340。图 34 中示出了过程“处理 ES_Descriptor”,并在下面的“处理

ES_Descriptor”之下描述该过程的操作。

随后，将索引“i”的值递增 1 3350，并重复“i”的值与 numEsDescrChildren 3310 的值的比较。

期望每一个从属于 Descr 元素 160（而 Descr 元素 160 从属于 InitialObjectDescriptor130）的 esDescr 元素 170 生成一个或两个 ES_Descriptor 元素 180、190，一个用于情景描述流（sdsm）180，而可能的一个用于对象描述符流（odsm）190。在 XMT-A 文档 100 包括音频、视频或其他对象时，期望有用于 odsm 的 ES_Descriptor 元素 190。使用过程“处理 ES_Descriptor”处理每一个 ES_Descriptor 元素 180、190。这一过程在下面进行描述。

7.1.5 处理 XMT-A 体元素（第 1 遍）

在操作 3120，构建一组列举 XMT-A 文档 100 的体元素 120 中定义的所有媒体对象、可重用 BIFS 节点和可重用路由的表。这些表用于确定媒体对象的数量、BIFS 节点的数量和路由的数量。这些表还用于确定媒体对象的某些属性，用于解决对媒体对象、BIFS 节点和路由的引用。

每一个媒体对象都由 ObjectDescriptor 元素 240 所定义。每一个 ObjectDescriptor 元素 240 都包含在对象描述符更新命令元素 220 中，而该命令元素包含在 XMT-A 文档 100 的体元素 120 内的“par”元素 140、200 中。媒体对象的属性包括 ObjectDescriptorID（ODID）、对象开始时间、对象结束时间和对象持续时间。ObjectDescriptorID 是指定为“ObjectDescriptor”元素 240 的“ObjectDescriptorID”属性的字母数字串。

对象开始时间由附加于“par”元素 200 的“begin”属性确定。

对象结束时间由附加于对象描述符移除命令元素 250 的“par”元素 200 的“begin”属性所确定。在对象描述符移除命令元素 250 中指定的 ObjectDescriptorID（ODID）属性的值必须与在相应的 ObjectDescriptor 元素 240 中指定的 ObjectDescriptorID 属性的值相

匹配。对象持续时间是对象结束时间与对象开始时间之间的差。

ObjectDescriptorID 串的值、及相关开始和停止时间存储于图 39A 中所示的对象表中。该表具有五列，**ObjectDescriptorID** 3910、**OdId** 3920、开始时间 3930、停止时间 3940、和 **EsId** 3950。每一列中的各项由标识该表中的每一行的“**position**”值 3900 所指示。

可重用 **BIFS** 节点由指定“**DEF**”属性的值的 **XMT-A BIFS** 节点元素所定义。该属性的值是字母数字串。可重用 **BIFS** 节点 **DEF** 串的值存储于图 39B 中所示的表中。该表具有一列，**NodeString** 3966。该列中的各项由标识该表中的每一行的“**position**”值 3960 所指示。

可重用路由指定“**DEF**”属性的值的 **XMT-A** 路由元素所定义。该属性的值是字母数字串。可重用路由 **DEF** 串的值存储于图 39C 所示的表中。该表具有一列，**RouteString** 3976。该列中的各项由标识该表中每一行的“**position**”值 3970 所指示。

第四个表记录替换情景命令的时间值。该表具有一列，**ReplaceSceneTime** 3986。该列中的各项由标识该表中的每一行的“**position**”值 3980 所指示。

通过遍历图 40 中所示的 **XMT-A**“体”元素内的从属元素来构建这些表。该过程开始操作 4000，向索引“**i**”指派值“0”。在操作 4000 完成之后，控制流传递给操作 4010。

在操作 4010，将索引“**i**”的值与量“**numBodyChildren**”的值进行比较。量 **numBodyChildren** 的值指示由 **XMT-A** 体元素 120 所拥有的从属元素的数量。

在操作 4010 之后，如果索引“**i**”的值等于 **numBodyChildren** 的值，则该过程完成 4060，且处理继续创建 **odsm** 的编辑列表。

如果索引“**i**”的值不等于 **numBodyChildren** 的值，则控制传递给操作 4020，在其中使用标准 XML 手段获得从属于体元素 120 的第 **i** 个元素，并将所得到的从属元素标识为“**bpdychild**”。以字符串量“**childName**”来标识元素 **bodyChild** 的元素名。

在操作 4020 完成之后，控制传递给操作 4030，在其中将字符串

量 **childName** 的值与字符串“**par**”进行比较。如果 **childName** 的值为“**par**”，则在处理操作 4040，向量“**parTime**”指派值“0”，并使用当前 **bodyChild** 元素作为父元素来执行过程“处理 XMT-A **par** 元素（第 1 遍）”4040。过程“处理 XMT-A **par** 元素（第 1 遍）”在下面进行描述。

在操作 4040 完成之后，控制传递给操作 4050，在其中随后将索引“**i**”的值递增 1，并重复“**i**”的值与 **numBodyChildren** 4010 的值的比较。

7.1.5.1 处理 XMT-A **par** 元素（第 1 遍）

XMT-A“**par**”元素 140、200 可从属于 XMT-A 体元素 120 或另一个 XMT-A“**par**”元素 200。对每一个 XMT-A“**par**”元素 200 的处理如图 41 所示。

在操作 4100，将当前“**par**”元素 200 的“**begin**”属性的值与量“**parTime**”4040、4136 的当前值相加，并将结果指派给量“时间”。

在操作 4106，向索引“**i**”指派值“0”。该索引不同于在替他过程中使用的类似索引值。

在操作 4110，将索引“**i**”的值与量“**numParChildren**”的值进行比较。量 **numParChildren** 的值指示当前“父”元素所拥有的从属元素的数量。

在操作 4116，如果索引“**i**”的值等于 **numParChildren** 的值，则该过程完成，且如果该“父”元素从属于体元素 120 则处理继续操作 4050，或如果“父”元素从属于另一“**par**”元素 200 则处理继续操作 4126。

在操作 4120，如果索引“**i**”的值不等于 **numParChildren** 的值，则使用标准 XML 手段获得从属于当前父元素的第 **i** 个元素，并将所得到的从属元素标识为“**parChild**”元素。**ParChild** 元素的元素名由字符串“**childName**”所标识。

在操作 4130，将字符串量 **childName** 的值与字符串“**par**”进行比较。

在操作 4136, 如果 `childName` 的值为“par”, 将量“时间”的值指派给量“parTime”, 并使用当前 `parChild` 元素作为“父”元素递归执行过程“处理 XMT-A par 元素 (第 1 遍)”。在该过程完成之后, 处理继续步骤 4126, 该步骤将在下面进行描述。

在操作 4140, 将字符串量 `childName` 的值与字符串“Delete”进行比较。如果 `childName` 的值为“Delete”, 不进行任何处理, 且处理继续操作 4126。

在操作 4150, 将字符串量 `childName` 的值与字符串“Insert”进行比较。

在操作 4160, 将字符串量 `childName` 的值与字符串“Replace”进行比较。

在操作 4166, 如果 `childName` 的值为“Insert”或“Replace”, 则使用当前 `parChild` 元素作为“父”元素执行过程“处理 XMT-A 命令元素 (第 1 遍)”。该过程将在下面描述。在该过程完成之后, 处理继续操作 4126。

在操作 4170, 将字符串量 `childName` 的值与字符串“ObjectDescriptorUpdate”进行比较。

在操作 4176, 如果 `childName` 的值为“ObjectDescriptorUpdate”, 则使用当前 `parChild` 元素作为“父”元素来执行过程“处理 ODUpdate cmd-1”。该过程将在下面描述。在该过程完成之后, 处理继续操作 4126。

在操作 4180, 将字符串量 `childName` 的值与字符串“ObjectDescriptorRemove”进行比较。

在操作 4186, 如果 `childName` 的值为“ObjectDescriptorRemove”, 则使用当前 `parChild` 元素作为“父”元素来执行过程“处理 ODRemove cmd”。该过程在下面描述。

在操作 4126, 随后将索引“i”的值递增“1”, 并重复操作 4110 的“i”的值与 `numParChildren` 的值的比较。

7.1.5.2 处理 XMT-A 命令元素（第 1 遍）

过程“处理 XMT-A 命令元素（第 1 遍）”可或者作为过程“处理 XMT-A par 元素（第 1 遍）”的一部分 4166 或递归作为过程“处理 BIFS 命令元素（第 1 遍）”的一部分 4270 来执行。如图 42 所示，该过程开始于操作 4200，向索引“i”指派值“0”（不同于其他过程中使用的类似索引值）。

在操作 4206，将索引“i”的值与量“numCmdChildren”的值进行比较。量 numCmdChildren 指示当前“父”元素 200 所拥有的从属元素的数量。

在操作 4290，如果索引“i”的值等于 numCmdChildren 的值，该过程完成，且如果“父”元素从属于体元素 120 则处理继续操作 4050，或如果“父”元素从属于另一个“par”元素 200 则处理继续操作 4126。

在操作 4210，如果索引“i”的值不等于 numCmdChildren 的值，则使用标准 XML 手段获得从属于当前父元素的第 i 个元素，且所得到的从属元素标识为“cmdChild”元素。以字符串量“childName”标识 cmdChild 元素的元素名。

在操作 4216，将字符串量 childName 的值与字符串“ROUTE”进行比较。

在操作 4220，如果 childName 的值为“ROUTE”，则使用标准 XML 手段获得 cmdChild 元素的“DEF”属性的值。如果不存在“DEF”属性，则不进行任何操作，并处理继续操作 4280。

在操作 4226，如果已为 cmdChild 元素的“DEF”属性指定了一个值，则将“DEF”属性的值指派给字符串量“idString”。然后，将 IdString 的值与路由 ID 表 3976 的每一个当前项进行比较。如果 idString 的值与该表中的任何当前项匹配，则处理继续操作 4280。

在操作 4230，如果 idString 的值不与路由 ID 表 3976 中的任何一个当前项匹配，则在该表中创建新项，并将 idString 的当前值指派给该新项。然后，处理继续操作 4280。

在操作 4240，将字符串量 childName 的值与字符串“Scene”进行

比较。

在操作 4246, 如果 `childName` 的值为“Scene”, 则在替换情景时间表 3986 中创建新项, 并将量“time”的当前值指派给该新项。处理继续操作 4280。

在操作 4250, 使用标准 XML 手段获得 `cmdChild` 元素的“DEF”属性的值。如果不存在“DEF”属性, 处理继续操作 4270。

在操作 4256, 如果已为 `cmdChild` 元素的“DEF”属性指定了一个值, 则将“DEF”属性的值指派给字符串量“idString”。然后, 将 `idString` 的值与 `NodeID` 表 3966 的每一个当前项进行比较。如果 `idString` 的值与该表中的任何当前项匹配, 则处理继续操作 4270。

在操作 4260, 如果 `idString` 的值不与 `NodeID` 表 3966 中的任何当前项匹配, 则在该表中创建新项, 并将 `idString` 的当前值指派给该新项。

然后, 在操作 4270, 递归使用当前 `cmdChild` 元素作为父元素执行当前过程 (处理 BIFS 命令元素 (第 1 遍))。

随后, 在操作 4280, 将索引“i”的值递增“1”, 并重复操作 4206 的“i”的值与 `numCmdChildren` 的值的比较。

7.1.5.3 “处理 ODUpdate 命令-1”过程

使用标准 XML 手段获得从属于 (父) 对象描述符更新命令元素 220 的“OD”元素 230。然后, 使用标准 XML 手段获得从属于“OD”元素 230 的 `ObjectDescriptor` 元素 240。将 `ObjectDescriptor` 元素 240 的“`ObjectDescriptorID`”属性 (在图 2B 中缩写为“ODID”) 的值与对象表 (图 39A) 中 `ObjectDescriptorID` 列 3910 中的各项进行比较。如果发现匹配, 则将量“time”的当前值指派给开始时间列 3930 中的相应项。

如果“`ObjectDescriptorID`”属性的值不与对象表的 `ObjectDescriptorID` 列 3910 中的任何当前项匹配, 则向对象表中添加新项, 将“`objectDescriptorID`”属性的值指派给 `ObjectDescriptorID` 列

3910 中的该新项，将量“time”的当前值放置在开始时间列 3930 中的相应项中，并将值“-1.0”放置在停止时间列 3940 中的相应项中。将对象表中的项数指派给 OdId 列 3920 中的相应项。作为过程“处理 XMT-A 体元素（第 2 遍）”的一部分，向 EsId 列 3950 中的相应项指派一个值。

7.1.5.4 “处理 ODRmove cmdnd”过程

将（父）对象描述符移除命令元素 250 的“ObjectDescriptorID”属性（在图 2B 中缩写为“ODID”）的值与对象表 3910 中的 ObjectDescriptorID 列中的各项进行比较。如果发现有匹配，则将量“time”的当前值指派给停止时间列 3940 中的相应项。

如果“objectDescriptorID”属性的值不与对象表的 ObjectDescriptorID 列 3910 中的任何当前项匹配，则向对象表添加一个新项，将“ObjectDescriptorID”属性的值指派给 ObjectDescriptorID 列 3910 中的该新项，将量“time”的当前值放置在停止时间列 3940 中的相应项中，并将值“-1.0”放置在开始时间列 3930 中的相应项中。对象表中的项数被指派给 OdId 列 3920 中的相应项。作为过程“处理 XMT-A 体元素（第 2 遍）”的一部分，将一个值指派给 EsId 列 3950 中的相应项。

7.1.6 为 odsm 创建编辑列表

在对 XMT-A 体元素的第一遍之后，比较对象表（图 39A）中的开始时间项 3930 的值，以找到最小开始时间项（startTimeMin）。比较停止时间项 3940 的相应值，以确定最大停止时间项（stopTimeMax）。将 stopTimeMax 的值与 startTimeMin 的值之间的差指派给量“duration”。

如果最小开始时间值大于 0，则将编辑列表（“edts”）元素 2644 插入到与 odsm（对象描述符流）相关的 trak 元素 2600 中。这按下述来实现：

1. 使用标准 XML 手段获得 mp4-file 文档 2300 中的 moov 元素 2320。

2. 使用标准 XML 手段获得 moov 元素 2320 中的每一个 trak 元素 2350。

3. 将每一个 track 元素 2350 的 trackID 属性的值与量 trackIdForOdsM 的值进行比较。当 odsM 的 trak 元素 2350 被创建时指派该值。

4. 如果特定 track 元素 2600 的 trackID 属性的值与 trackIdForOdsM 的值匹配, 则执行下述步骤。

5. 使用标准 XML 手段创建新的 edts 元素 2644, 并将其插入到所选择的 trak 元素 2600 中。

6. 使用标准 XML 手段创建新的 elst 元素 (2648), 并将其插入到新的 edts 元素 2644 中。

7. 使用标准 XML 手段创建新的 segment 元素, 并将其插入到新的 elst 元素 2648 中。

向该“segment”元素的“开始时间”属性指派值“-1”。向该“segment”元素的“rate”属性指派值“1.0”。向该新“segment”元素的“duration”属性指派“moov”元素 2320 中的“timeScale”属性与开始时间的值的乘积。

8. 使用标准 XML 手段创建第二个新的 segment 元素, 并将其插入到新的 elst 元素 2648 中。

向该第二个“segment”元素的“开始时间”属性指派值“0”。向该第二个“segment”元素的“rate”属性指派值“1.0”。向该第二个“segment”元素的“duration”属性指派量“duration”的值与“moov”元素 2320 中的“timeScale”的值的乘积。

7.1.7 处理 XMT-A 体元素 (第 2 遍)

在该步骤中, 再次对 XMT-A“体”元素的从属元素进行遍历, 如图 40 所示的那样。使用图 43 中所示的过程“处理 XMT-A par 元素(第

2 遍)”对在此次遍历中发现的每一个从属“par”元素进行处理 4040。在该过程完成之后，处理继续步骤 8，“向 mp4bifs 文档插入命令帧”。

7.1.7.1 处理 XMT-A par 元素（第 2 遍）

在操作 4300，将当前“父”元素 200 的“begin”属性的值与量“parTime”4040、4346 的当前值相加，并将结果指派给量“time”。

在操作 4306，向索引“i”指派值“0”。该索引不同于在其他过程中使用的类似索引值。

在操作 4310，将索引“i”的值与量“numParChildren”的值进行比较。量 numParChildren 的值指示当前“父”元素所拥有的从属元素的数量。

在操作 4316，如果索引“i”的值等于 numParChildren 的值，则该过程完成，且如果该“父”元素从属于体元素 120 则处理继续 4050，或如果“父”元素从属于另一个“par”元素 200 则处理继续 4336。

在操作 4320，如果索引“i”的值不等于 numParChildren 的值，则使用标准 XML 手段获得从属于当前父元素的第 i 个元素，并将所得到的元素标识为“parChild”元素。ParChild 元素的元素名由字符串量“childName”标识。

在操作 4330，使用标准 XML 手段创建新的命令帧元素 2820、2830。

在操作 4340，将字符串量 childName 的值与字符串“par”进行比较。

在操作 4346，如果 childName 的值为“par”，则将量“time”的值指派给量“parTime”，并使用当前 parChild 元素作为“父”元素递归执行过程“处理 XMT-A par 元素（第 2 遍）”。在该过程完成之后，处理继续步骤 4336，该步骤将在下面描述。

在操作 4350，将字符串量 childName 的值与字符串“ObjectDescriptorUpdate”进行比较。

在操作 4356，如果 childName 的值为“ObjectDescriptorUpdate”，

则使用当前 **parChild** 元素作为“父”元素执行过程“处理 **ODUpdate** 命令-2”。该过程将在下面描述。在该过程完成之后，处理继续 4126。

在操作 4360，将字符串量 **childName** 的值与字符串“**Insert**”进行比较。

在操作 4366，如果 **childName** 的值为“**Insert**”，则使用当前 **parChild** 元素作为“父”元素执行过程“处理插入命令”。该过程将在下面描述。在该过程完成之后，处理继续步骤 4336，该步骤将在下面描述。

在操作 4370，将字符串量 **childName** 的值与字符串“**Delete**”进行比较。

在操作 4376，如果 **childName** 的值为“**Delete**”，则使用当前 **parChild** 元素作为“父”元素执行过程“处理删除命令”。该过程将在下面描述。在该过程完成之后，处理继续步骤 4336，该步骤将在下面描述。

在操作 4380，将字符串量 **childName** 的值与字符串“**Replace**”进行比较。

在操作 4386，如果 **childName** 的值为“**Replace**”，则使用当前 **parChild** 元素作为“父”元素执行过程“处理替换命令”。该过程将在下面描述。

在操作 4336，将在步骤 4330 创建的当前 **commandFrame** 元素 2830 插入到临时排序的 **commandFrame** 元素的列表中。这通过将每一个新的 **commandFrame** 元素插入到该列表中的一个位置处，该位置在该列表的具有大于新的 **commandFrame** 元素的 **time** 属性的值的 **time** 属性的第一个成员之前，或在该列表不具有大于新的 **commandFrame** 元素的 **time** 属性的值的 **time** 属性的当前成员时处于该列表的末端。在操作 4330 之后立即执行操作 4336，而不是在操作 4326 之前立即执行操作 4336 是可能的（如图 43 所示）。在每一种情况下，操作 4330 和 4336 可创建不包含命令的空 **commandFrame** 元素、具有与其他 **commandFrame** 元素相同的 **time** 属性的多个

commandFrame 元素。在操作 3136，消除空 **commandFrame** 元素，并组合具有相同 **time** 属性的多个命令帧。

随后，在操作 4326，将索引“i”的值递增“1”，并重复“i”的值与 **numParChildren** 4310 的比较。

7.1.7.2 处理 ODUpdate 命令-2

使用标准 XML 手段获得从属于（父）对象描述符更新命令元素 220 的“OD”元素 230。然后，使用标准 XML 手段获得从属于“OD”元素 230 的 **ObjectDescriptor** 元素 240。然后，使用标准 XML 手段获得从属于“**ObjectDescriptor**”元素 600 的“**Descr**”元素 610。如图 32 中所示的那样对“**Descr**”元素 610 进行处理，以获得从属“**esDescr**”元素 620。如图 33 中所示的那样对“**esDescr**”元素 620 进行处理，以获得从“**ES_Descriptor**”元素 630。然后，对于每一个以这种方式获得的“**ES_Descriptor**”元素 630 执行下面描述的过程“处理 **ES_Descriptor**”。

7.1.7.3 处理插入命令

图 44 中示出了用于处理“插入”命令元素的步骤。该过程中的父元素是 **XMT-A** 插入命令元素 300。该命令元素可以从属于 **XMT-A**“**par**”元素 200，或从属于条件节点元素 400 的“**buffer**”属性元素 410。如果插入命令元素从属于 **XMT-A**“**par**”元素 200，则 **mp4bifs**“目标”元素是在 4330 中创建的命令帧元素 2820、2830。如果插入命令元素从属于条件节点元素 400 的“**buffer**”属性元素 410，则 **mo4-bifs**“目标”元素是 **mp4-bifs** 条件节点元素。

最初，向两个布尔值 **bInsertNode** 和 **bInsertValue** 指派值“假”。

在操作 4400，将父元素的“**atNode**”属性的值指派给量“**NodeId**”。如果已对“**atNode**”属性指定了一个值，则该过程继续操作 4446。

在操作 4406，如果已对“**atNode**”属性指定了一个值，则将父元素的“**atField**”属性的值指派给量“**FieldName**”。

在操作 4416，如果已对“**atField**”属性指定了一个值，则将量

“FieldName”的值与字符串“children”进行比较。

在操作 4410，如果已对“atField”指定了一个值，或量“FieldName”的值为“children”，则向布尔量“bInsertNode”指派值“真”，且该过程继续操作 4446。

在操作 4420，如果已对“atField”属性指定了一个值，且量“FieldName”的值不是“children”，则使用标准 XML 手段创建新的 mp4-bifs 插入索引值命令元素（“newCommand”）。然后，使用标准 XML 手段将该 newCommand 元素添加到当前 mp4-bifs 目标元素。

在操作 4426，将父元素的“value”属性的值指派给量“value”。

在操作 4430，如果已对 XMT-A BIFS 命令元素的“value”属性指定了一个值，则向新 mp4bifs newCommand 元素指派一个值。在大多数情况下，指派给新 mp4bifs newCommand 元素的“value”属性的值等于 XMT-A BIFS 命令元素的“value”属性的值。在下面识别的某些情况（数据格式转换）下，指派给新 mp4bifs newCommand 元素的“value”属性的值是从 XMT-A BIFS 命令元素的“value”属性的值中导出的。

在这种情况下，该过程完成，且处理继续操作 4336 或对 XMT-A 条件节点元素 4890 的处理。

在操作 4436，如果已对“value”属性指派了一个值，则向布尔量“nInsertValue”指派值“真”。创建包括直接从属于当前父元素的所有元素的元素名列表的字符串量“childName”。

在操作 4440，将字符串量“childNames”的值指派给新 mp4-bifs 元素“newCommand”的“value”属性。

在操作 4446，向索引“i”指派值“0”，该索引值不同于在其他部分定义的其他索引值。

在操作 4450，将索引“i”的值与量“numCmdChildren”的值进行比较。量 numCmdChildren 的值指示当前“父”元素所拥有的从属元素的数量。

在操作 4456，如果索引“i”的值等于 numCmdChildren 的值，则

该过程完成，且处理继续操作 4336 或对 XMT-A 条件节点元素 4890 的处理。

在操作 4460，如果索引“i”的值不等于 numCmdChildren 的值，则使用标准 XML 手段获得从属于当前父元素的第 i 个元素，并将所得到的从属元素标识为“insertChild”元素。该 insertChild 元素的元素名由字符串量“childName”所标识。

在操作 4470，将字符串量 childName 的值与字符串“ROUTE”进行比较。

在操作 4476，如果字符串量 childName 的值为“ROUTE”，则执行过程“创建插入路由命令”。然后，使用标准 XML 手段将所得到的 newCommand 元素添加到当前 mp4-bifs 目标元素。然后，该过程继续操作 4466。

在操作 4480，如果字符串量 childName 的值不是“ROUTE”，则将布尔量 bInsertNode 的值与值“真”进行比较。

在操作 4486，如果布尔量 bInsertNode 的值为“真”，则执行过程“创建插入节点命令”。然后，使用标准 XML 手段将所得到的 newCommand 元素添加到当前 mp4-bifs 目标元素。处理继续操作 4496。

在操作 4490，如果布尔量 bInsertNode 的值不是“真”，则将布尔量 bInsertValue 的值与值“真”进行比较。

在操作 4496，如果布尔量 bInsertValue 的值为“真”，或布尔量 bInsertNode 的值为“真”，则使用当前 insertChild 元素作为父元素执行过程“处理 XMT-A BIFS 节点”。然后，使用标准 XML 手段将所得到的 mp4-bifs 节点元素添加到 newCommand 元素。处理继续操作 4466。

在操作 4498，如果布尔量 bInsertValue 的值不是“真”，则 XMT-A 文档无效，并报告出错。使用当前 insertChild 元素作为父元素执行过程“处理 XMT-A BIFS 节点”。然后，使用标准 XML 手段将所得到的 mp4-bifs 节点元素添加到当前 mp4-bifs 目标元素。处理继续操作 4466。

在操作 4466, 将索引“i”的值递增“1”, 并重复 numCmdChildren 的比较 4450。

7.1.7.4 “创建插入路由命令”过程

使用标准 XML 手段创建新的 mp4-bifs 插入路由命令元素 (“newCommand”)。

将 XMT-A 父元素的“fromNode”属性的值与 BIFS 节点 ID 表(图 39B)中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 fromNodeID, 并将结果递增“1”。然后, 将该结果指派给 newCommand 元素的“departureNode”属性。

将 XMT-A 父元素的“formField”属性的值指派给 newCommand 元素的“departureFieldName”属性。

将 XMT-A 父元素的“toNode”属性的值与 BIFS 节点 ID 表(39B)中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 fromNodeId, 并将结果递增“1”。然后, 将该结果指派给 newCommand 元素的“arrivalNode”属性。

将 XMT-A 父元素的“toField”属性的值指派给 newCommand 元素的“arrivalFieldName”属性。

如果已对 XMT-A 父元素的“DEF”属性指定了一个值, 则将该属性的值与 BIFS 路由 ID 表(图 39C)中的各项 3976 进行比较。将匹配项的“位置”3970 的值指派给整型量 routeID, 并将结果递增“1”。然后, 将该结果指派给 newCommand 元素的“routeId”属性。如果布尔量 bUseNames 的值为真, 则将“DEFS”属性的值指派给 newCommand 元素的“name”属性。BUseName 的值是在处理“替换情景”命令时建立的。

7.1.7.5 “创建插入节点命令”过程

使用标准 XML 手段创建新的 mp4-bifs 插入节点命令元素 (“newCommand”)。

将量“NodeId”的值与 BIFS 节点 ID 表（图 39B）中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 atNodeId，并将结果递增“1”。然后，将该结果指派给 newCommand 元素的“parentID”属性。

如果 XMT-A 父元素的“位置”属性的值为“BEGIN”，则向 newCommand 元素的“插入位置”属性指派值“2”。

如果 XMT-A 父元素的“位置”属性的值为“END”，则向 newCommand 元素的“插入位置”属性指派值“3”。

如果 XMT-A 父元素的“位置”属性的值不是“BEGIN”，也不是“END”，则向 newCommand 元素的“插入位置”属性指派值“0”，并将 XMT-A 父元素的“位置”属性的值指派给 newCommand 元素的“位置”属性。

7.1.7.6 处理删除命令

图 45 中示出了用于处理“删除”命令元素的步骤。该过程中的父元素为 XMT-A 删除命令元素 310。该命令元素可从属于 XMT-A“par”元素 200，或从属于条件节点元素 400 的“buffer”属性元素 410。如果删除命令元素从属于 XMT-A“par”元素 200，mp4bifs“目标”元素为在 4330 中创建的命令帧元素 2820、2830。如果删除命令元素从属于条件节点元素 400 的“buffer”属性元素 410，则 mp4-bifs“目标”元素为 mp4-bifs 条件节点元素。

在操作 4500，将父元素的“atRoute”属性的值指派给量“RouteId”。

在操作 4510，如果已对“atRoute”属性指定了一个值，则使用标准 XML 手段创建新的 mp4-bifs 删除路由命令元素（“newCommand”），并将量“RouteId”的值指派给 newCommand 元素的“routeId”属性。然后，使用标准 XML 手段将 newCommand 元素添加到当前 mp4-bifs 目标元素。

在操作 4520，将父元素的“atNode”属性的值指派给量“NodeId”。

在操作 4530, 如果已对“atNode”属性指定了一个值, 则该 XMT-A 文档无效。

在操作 4540, 如果已对“atNode”属性指定了一个值, 则将父元素的“atField”属性的值指派给量“FieldName”。如果已对“atField”属性指定了一个值, 则将量“FieldName”的值与字符串“children”进行比较。

在操作 4560, 如果已对“atField”指定了一个值, 且量“FieldName”的值不是“children”, 则使用标准 XML 手段创建新的 mp4-bifs 删除索引值命令元素 (newCommand)。将量 atField 的值指派给 newCommand 元素的“infieldName”属性。

如果 XMT-A 父元素的“位置”属性的值为“BEGIN”, 则向 newCommand 元素的“删除位置”属性指派值“2”。

如果 XMT-A 父元素的“位置”属性的值为“END”, 则向 newCommand 元素的“删除位置”属性指派值“3”。

如果 XMT-A 父元素的“位置”属性的值不为“BEGIN”, 也不为“END”, 则向 newCommand 元素的“删除位置”属性指派值“0”, 并将 XMT-A 父元素的“位置”属性的值指派给 newCommand 元素的“位置”属性。

然后, 使用标准 XML 手段将 newCommand 元素添加到当前 mp4-bifs 目标元素。

在操作 4580, 如果已对“atField”属性指定了一个值, 或量“FieldName”的值为“children”, 则使用标准 XML 手段创建新的 mp4-bifs 删除命令元素 (“newCommand”)。

将量“NodeId”的值与 BIFS 节点 ID 表 (图 39B) 中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 atNodeId, 并将结果递增“1”。然后, 将该结果指派给 newCommand 元素的“nodeId”属性。

然后, 使用标准 XML 手段将 newCommand 元素添加到当前 mp4-bifs 目标元素。

7.1.7.7 处理替换命令

图 46 中示出了用于处理 XMT-A“替换”元素的步骤。该过程中的父元素是 XMT-A 替换命令元素 320。该命令元素可从属于 XMT-A“par”元素 200，或从属于条件节点元素 400 的“buffer”属性元素 410。如果该替换命令元素从属于 XMT-A“par”元素 200，则 mp4bifs“目标”元素是在 4330 中创建的命令帧元素 2820、2830。如果该替换命令元素从属于条件节点元素 400 的“buffer”属性元素 410，则该 mp4-bifs“目标”元素是 mp4-bifs 条件节点元素。

最初，向两个布尔值 bReplaceNode 和 bReplaceValue 指派值“假”。

在操作 4600，将父元素的“atNode”属性的值指派给量“NodeId”。如果已对“atNode”属性指定了一个值，则该过程继续操作 4636。

在操作 4604，如果已对“atNode”属性指定了一个值，则将父元素的“atField”属性的值指派给量“FieldName”。

在操作 4612，如果已对“atField”属性指定了一个值，则使用标准 XML 手段创建新的 mp4-bifs 替换节点命令元素（“newCommand”）。将量“NodeId”的值与 BIFS 节点 ID 表（图 39B）中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 atNodeId，并将结果递增“1”。然后，将该结果指派给 newCommand 元素的“nodeId”属性。

使用标准 XML 手段将 newCommand 元素添加到 mp4-bifs 目标元素中。向布尔量“bReplaceNode”指派值“真”，且该过程继续操作 4636。

在操作 4616，如果已对“atField”属性指定了一个值，则使用标准 XML 手段创建新的 mp4-bifs 替换字段命令元素（“newCommand”）。将量“NodeId”的值与 BIFS 节点 ID 表（图 39B）中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 atNodeId，并将结果递增“1”。然后，将该结果指派给 newCommand 元素的“nodeId”属性。

将量“FieldName”的值指派给 newCommand 元素的

“inFieldName”属性。然后，使用标准 XML 手段将 newCommand 元素添加到当前 mp4-bifs 目标元素中。

在操作 4629，将父元素的“value”属性的值指派给量“value”。

在操作 4624，如果已对 XMT-A BIFS 命令元素的“value”属性指定了一个值，则向新的 mp4bifs newCommand 元素的“value”属性指派一个值。在大多数情况下，指派给新的 mp4bifs newCommand 元素的“value”属性的值等于 XMT-A BIFS 命令元素的“value”属性的值。在下面识别的某些情况（数据格式转换）下，指派给新的 mp4bifs newCommand 元素的“value”属性的值是从 XMT-A BIFS 命令元素的“value”属性的值中导出的。在这种情况下，该过程完成，且处理继续操作 4336 或对 XMT-A 条件节点元素 4890 的处理。

在操作 4628，如果已对“value”属性指定了一个值，则向布尔量“bReplaceField”指派值“真”。创建包括直接从属于当前父元素的所有元素的元素名列表的字符串量“childNames”。

在操作 4632，将字符串量“childNames”的值指派给该新的 mp4-bifs 元素“newCommand”的“value”属性。

在操作 4636，向索引“i”指派值“0”，其不同于其他部分定义的其他索引值。

在操作 4640，将索引“i”的值与量“numCmdChildren”的值进行比较。量 numCmdChildren 的值指示当前“父”元素所拥有的从属元素的数量。

在操作 4644，如果索引“i”的值等于 numCmdChildren 的值，则该过程完成，且处理继续操作 4336 或对 XMT-A 条件节点元素操作 4890 的处理。

在操作 4648，如果索引“i”的值不等于 numCmdChildren 的值，则使用标准 XML 手段获得从属于当前父元素的第 i 个元素，并将所得到的从属元素标识为“replaceChild”元素。该 replaceChild 元素的元素名由字符串量“childName”所标识。

在操作 4652，将字符串量 childName 的值与字符串“Scene”进行

比较。在操作 4656，如果字符串量 `childName` 的值为“Scene”，则执行过程“创建替换情景命令”。然后，使用标准 XML 手段将所得到的 `newCommand` 元素添加到当前 `mp4-bifs` 目标元素中。然后，该过程继续操作 4696。

在操作 4660，如果字符串量 `childName` 的值不是“Scene”，则将字符串量 `childName` 的值与字符串“ROUTE”进行比较。在操作 4664，如果字符串量 `childName` 的值为“ROUTE”，则执行过程“创建替换路由命令”。然后，使用标准 XML 手段将所得到的 `newCommand` 元素添加到当前 `mp4-bifs` 目标元素中。然后，该过程继续操作 4696。

在操作 4668，如果字符串量 `childName` 的值不是“ROUTE”，则将布尔量 `bReplaceNode` 的值与值“真”进行比较。

在操作 4672，如果布尔量 `bReplaceNode` 的值为“真”，则使用当前 `replaceChild` 元素作为父元素执行过程“处理 XMT-A BIFS 节点”。然后，使用标准 XML 手段将所得到的 `mp4-bifs` 节点元素添加到当前 `mp4-bifs` 目标元素中。处理继续操作 4496。

在操作 4680，如果布尔量 `bReplaceNode` 的值不为“真”，则将布尔量 `bReplaceField` 的值与值“真”进行比较。

在操作 4684，如果布尔量 `bReplaceField` 的值为“真”，则使用当前 `replaceChild` 元素作为父元素执行过程“处理 XMT-A BIFS 节点”。然后，使用标准 XML 手段将所得到的 `mp4-bifs` 节点元素添加到 `newCommand` 元素中。处理继续操作 4496。

在操作 4690，如果布尔量 `bReplaceNode` 的值不为“真”，则 XMT-A 文档无效。使用当前 `replaceChild` 元素作为父元素执行过程“处理 XMT-A BIFS 节点”。然后，使用标准 XML 手段将所得到的 `mp4-bifs` 节点元素添加到当前 `mp4-bifs` 目标元素中。处理继续操作 4496。

在操作 4696，将索引“i”的值递增“1”，并重复操作 4640 的与 `numCmdChildren` 的比较。

7.1.7.8 “创建替换路由命令”过程

使用标准 XML 手段创建新的 mp4-bifs 替换路由元素 (“newCommand”)。

将 XMT-A 父元素的“fromNode”属性的值与 BIFS 节点 ID 表(参见图 39B)中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量“fromNodeId”，并将结果递增“1”。然后，将该结果指派给 newCommand 元素的“departureNode”属性。

将 XMT-A 父元素的“fromField”属性的值指派给 newCommand 元素的“departureFieldName”属性。

将 XMT-A 父元素的“toNode”属性的值与 BIFS 节点 ID 表(参见图 39B)中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 fromNodeId，并将结果递增“1”。然后，将该结果指派给 newCommand 元素的“arrivalNode”属性。

将 XMT-A 父元素的“toField”属性的值指派给 newCommand 元素的“arrivalFieldName”属性。

将 XMT-A 父元素的“atRoute”属性的值与 BIFS 路由 ID 表(见图 39C)中的各项 3976 进行比较。将匹配项的“位置”3970 的值指派给整型量 routeId，并将结果递增“1”。然后，将该结果指派给 newCommand 元素的“routeId”属性。如果已对 XMT-A 父元素的 atRoute 属性指定了一个值，则该 XMT-A 文档有效。

7.1.7.9 “创建替换情景命令”过程

图 47 中示出了用于创建 mp4-bifs“替换情景”命令元素的步骤。该过程中的 XMT-A 父元素是 XMT-A 情景命令元素 320。该命令元素通常从属于 XMT-A“替换”元素 200。

在操作 4700，使用标准 XML 手段创建新的 mp4-bifs“替换情景”命令元素 2930 (newCommand)。使用标准 XML 手段将该新命令元素添加到当前 mp4-bifs 目标元素中。该 mp4-bifs 目标元素必须是 mp4-bifs 命令帧元素 2830 或 mp4-bifs 条件节点元素。向布尔量

“bHaveRoutes”指派值“假”。

在操作 4710, 将 XMT-A“情景”元素的“useNames”属性的值指派给布尔量“USENAMES”。

在操作 4716, 如果已对 XMT-A“情景”元素的“useNames”属性指定了一个值, 则将布尔量“USENAMES”的值与值“真”进行比较。

在操作 4720, 如果已对 XMT-A“情景”元素的“useNames”属性指定了一个值, 或布尔量“USENAMES”的值不为“真”, 则向布尔量“bUseNames”指派值“假”。

在操作 4726, 如果布尔量“USENAMES”的值为“真”, 则向布尔量“bUseNames”指派值“真”。

在操作 4730, 向索引“i”指派值“0”, 其不同于其他部分定义的其他索引值。

在操作 4740, 将索引“i”的值与量“numSceneChildren”的值进行比较。量 numSceneChildren 的值指示该 XMT-A 情景元素所拥有的从属元素的数量。

在操作 4746, 如果索引“i”的值等于 numSceneChildren 的值, 则该过程完成, 且处理继续操作 4656。

在操作 4750, 如果索引“i”的值不等于 numSceneChildren 的值, 则使用标准 XML 手段获得从属于 XMT-A 情景元素的第 i 个元素, 并将所得到的从属元素标识为“sceneChild”元素。SceneChild 元素的元素名由字符串量“childName”所标识。

在操作 4760, 将字符串量 childName 的值与字符串“ROUTE”进行比较。

在操作 4766, 如果字符串量 childName 的值为“真”, 则将布尔量 bHaveRoutes 的值与值“真”进行比较。

在操作 4770, 如果布尔量 bHaveRoutes 的值不为“真”, 则使用标准 XML 手段创建新的 mp4-bifs“Routes”元素。使用标准 XML 手段将所得到的 mp4-bifs“Routes”元素添加到 newCommand 元素中。向布尔量“bHaveRoutes”指派值“真”。

在操作 4776, 使用标准 XML 手段创建新的 mp4-bifs“Route”元素。使用标准 XML 手段将所得到的 mp4-bifs“Route”元素添加到 mp4-bifs“Routes”元素中。

将 XMT-A 父元素的“fromNode”属性的值与 BIFS 节点 ID 表(见图 39B)中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 fromNodeId, 并将该结果递增“1”。然后, 将该结果指派给 mp4-bifs Route 元素的“fromNode”属性。

将 XMT-A 父元素的“fromField”属性的值指派给 mp4-bifs Route 元素的“fromFieldName”属性。

将 XMT-A 父元素的“toNode”属性的值与 BIFS 节点 ID 表(见图 39B)中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 fromNodeId, 并将结果递增“1”。然后, 将该结果指派给 mp4-bifs Route 元素的“toNode”属性。

将 XMT-A 父元素的“toField”属性的值指派给 mp4-bifs Route 元素的“toFieldName”属性。

如果已对 XMT-A 父元素的“DEF”属性指定了一个值, 则将该属性的值与 BIFS 路由 ID 表(见图 39C)中的各项 3976 进行比较。将匹配项的“位置”3970 的值指派给整型量 routeId, 并将结果递增“1”。然后, 将该结果指派给 mp4-bifs Route 元素的“routeId”属性。如果布尔量 bUseName 的值为真, 则将“DEFS”属性的值指派给 mp4-bifs Route 元素的“name”属性。

在操作 4780, 如果字符串量 childName 的值不为“ROUTE”, 则使用当前 sceneChild 元素作为父元素执行过程“处理 XMT-A BIFS 节点”。使用标准 XML 手段将所得到的 mp4-bifs 节点元素添加到 newCommand 元素中。处理继续操作 4496。

在操作 4790, 将索引“i”的值递增“1”, 并重复操作 4740 的与 numSceneChildren 的比较。

7.1.7.10 “处理 XMTA BIFS 节点”过程

在 MPEG-4 系统规范中定义了 100 多种类型的 BIFS 节点。每一个 MPEG-4 BIFS 节点具有特定节点名称和一组指定属性字段。每一个指定属性字段具有特定的数据类型，诸如布尔型、整型、浮点型、字符串型、“node”或“buffer”。对于每一种类型的 MPEG-4 BIFS 节点，为 XMTA 文档和 mp4bifs 文档定义了相应的同名节点元素。为 mp4bifs 文档定义的每一个节点元素拥有一组具有匹配相应 MPEG-4 BIFS 节点的那些属性字段的名称的属性。

如图 30 所示，每一个具有数据类型“node”或“buffer”的 MPEG-4 BIFS 属性字段还可由 mp4bifs 节点元素的一个或多个从属元素代表，且 mp4bifs 节点元素的相应属性包括与该属性字段相关的从属元素的元素名称的列表。这些从属元素可以是节点元素或命令元素。以这种方式，每一个 mp4bifs 节点元素的结构模拟相应 MPEG-4 BIFS 节点的结构。

除了为每一个 XMT-A 节点元素定义的属性仅包括不具有“node”或“buffer”的数据类型的属性之外，为 XMT-A 文档定义的节点元素与那些为 mp4bifs 文档定义的节点元素类似。对于具有“node”或“buffer”的数据类型的 MPEG-4 BIFS 节点的每一个属性，XMTA 规范定义了不具有属性的同名从属属性元素，且相应属性字段由从属于这些属性元素的节点元素或命令元素代表。

如图 48 所示，对于 XMTA BIFS 节点元素的转换过程开始于 4800，将 XMT-A 节点元素的“USE”属性的值指派给字符串量“nodeRef”。如果已对该 XMT-A 节点元素的“USE”属性指定了一个值，则在操作 4806 使用标准 XML 手段创建新的 mp4bifs 重用节点元素。使用标准 XML 手段将该新的重用节点元素插入到当前 mp4bifs 目标元素中。

在操作 4810，将字符串量“nodeRef”的值与 BIFS 节点 ID 表（见图 39B）中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 nodeId，并将结果递增“1”。然后，将该结果指派给 newCommand 元素的“nodeRef”属性。在操作 4816，对该 XMT-A 节

点元素的处理完成，处理继续对拥有该 XMT-A 节点元素的 XMT-A BIFS 命令元素或父 XMT-A BIFS 节点元素进行处理。

在操作 4820，如果已对该 XMT-A 节点元素的“USE”属性指定了一个值，则使用标准 XML 手段创建新的 mp4bifs NodeName 元素，其中“NodeName”代表当前 XMT-A BIFS 节点元素的名称。使用标准 XML 手段将该新的“NodeName”元素插入到当前 mp4bifs 目标元素中。例如，如果当前 XMT-A BIFS 节点元素的元素名为“Geometry”，则创建新的 mp4bifs“Geometry”元素，并将其插入到当前 mp4bifs 目标元素中。

如果已对 XMT-A BIFS 节点元素的“DEF”属性指定了一个值，则将“DEF”属性的值与 BIFS 节点 ID 表（见图 39B）中的各项 3966 进行比较。将匹配项的“位置”3960 的值指派给整型量 nodeId，并将结果递增“1”。然后，将该结果指派给 mp4bifs “NodeName”元素的“nodeId”属性。如果布尔量“bUseNames”为真，则将 XMT-A BIFS 节点元素的“DEF”属性的值指派给 mp4bifs NodeName 元素的“name”属性。

使用 XMT-A BIFS 节点元素的所有其他属性的值向 mp4bifs NodeName 元素的同名属性指派各值。在大多数情况下，指派给 mp4bifs NodeName 元素的每一个属性的值等于 XMT-A BIFS 节点元素的相应属性的值。在下面识别的某些情况（数据格式转换）下，指派给 mp4bifs NodeName 元素的属性的值是从 XMT-A BIFS 节点元素的相应属性的值中导出的。

在操作 4826，向索引“i”指派值“0”，其不同于其他部分定义的其他索引值。

在操作 4830，将索引“i”的值与量“numNodeChildren”的值进行比较。量 numNodeChildren 的值指示当前 XMT-A BIFS 节点元素所拥有的从属元素的数量。NumNodeChildren 的非 0 值可能仅用于代表具有“Node”或“Command Buffer”字段数据类型的数据字段的 MPEG-4 BIFS 节点的 XMT-A BIFS 节点元素。

在操作 4836, 如果索引“i”的值等于 numNodeChildren 的值, 则该过程完成, 且处理继续对拥有该 XMT-A 节点元素的 XMT-A BIFS 命令元素或父 XMT-A BIFS 节点元素的处理。

在操作 4840, 如果索引“i”的值不等于 numNodeChildren 的值, 则使用标准 XML 手段获得从属于当前父元素的第 i 个元素, 并将所得到的从属元素标识为“nodeChild”元素。NodeChild 元素的元素名由字符串量“childName”所标识。量“childName”的值将与具有“Node”或“Command Buffer”字段数据类型的 MPEG-4 BIFS 节点数据字段的字段名相匹配。

在操作 4846, 使用标准 XML 手段获得从属于 nodeChild 元素的所有元素的元素名。将这些元素名的值以空格隔开, 连接成字符串量“NameList”。将所得到的字符串量“NameList”的值指派给当前 mp4bifs NodeName 元素的 childName 属性。例如, 如果 childName 的值为“children”, 则将从属于 XMT-A“children”元素的 XMT-A 元素的元素名列表指派给当前 mp4bifs NodeName 元素的“children”属性。

在操作 4850, 向索引“j”指派值“0”, 其不同于其他部分定义的其他索引值。

在操作 4856, 将索引“j”的值与量“numNodeChildChildren”的值进行比较。量 numNodeChildChildren 的值指示当前“nodeChild”元素所拥有的从属元素的数量。

在操作 4860, 如果索引“j”的值等于 numNodeChildChildren 的值, 则将索引“i”的值递增“1”, 并重复操作 4830 的与 numNodeChildren 的比较。

在操作 4866, 如果索引“j”的值不等于 numNodeChildChildren 的值, 则使用标准 XML 手段获得从属于当前 nodeId 元素的第 j 个元素, 并将所得到的从属元素标识为“attributeChild”元素。AttributeChild 元素的元素名由字符串量“attributeChildName”所标识。

在操作 4866, 将字符串量 childName 的值与字符串“buffer”进行

比较。

在操作 4870, 如果字符串量 `childName` 的值为“buffer”, 则执行过程“处理 XMT-A 命令”。然后, 使用标准 XML 手段将所得到的 `newCommand` 元素添加到当前 `mp4-bifs NodeName` 元素中。过程“处理 XMT-A 命令”与图 43 中所示的过程“处理 XMT-A par 元素 (第 2 遍)”的操作 4360 到 4386 相同, 使用 `attributeChildName` 的值作为 `childName` 的值。这是一个递归的过程, 因为当前过程通常从属于过程“处理 XMT-A par 元素 (第 2 遍)”。然后, 该过程继续操作 4890。

在操作 4880, 如果字符串量 `childName` 的值不是“buffer”, 则递归执行过程“处理 XMT-A BIFS 节点”。然后, 使用标准 XML 手段将所得到的 `NodeName` 元素添加到当前 `mp4-bifs NodeName` 元素中。

在操作 4890, 将索引“j”的值递增“1”, 并重复操作 4856 的与 `numNodeChildChildren` 的比较。

7.1.7.11 数据格式转换

将数据格式转换应用于 XMTA BIFS 节点的下列属性字段属性: 这些转换还应用于 XMT-A 插入命令元素操作 4430 和 XMT-A 替换命令元素操作 4624 的“value”属性的值。在 XMT-A 插入和替换命令的情况下, 数据类型由相应 `atField` 属性的值所确定。

1. 由六个数字的十六进制串“#RRGGBB”表示具有数据类型“color”的字段属性的每一个 XMTA 属性值。其被转换为三部分十进制表示“rrr ggg bbb”, 其中“rrr”是被 256 除的十六进制值 0xRR 的十进制表示, “ggg”是被 256 除的十六进制值 0xGG 的十进制表示, 而“bbb”是被 256 除的十六进制值 0xBB 的十进制表示。

2. 将具有数据类型“string”的字段属性的每一个 XMTA 属性值从为 XMTA 定义的引用串格式转换为 `mp4bifs` 使用的可选格式。该转换包括“引用”符号 (\) 的移除 (除非其前面有反斜杠符号 (\))、以其后有两个数字的十六进制码的百分号 (%) 替换空格和字符串内的其他“特殊”符号、以及以空格分隔多个字符串。上述的“特殊”符号

包括空格、引用号、百分号（%）、and 符号（&）、大于号（>）、具有小于 32 的数字值的符号、及具有大于 127 的数字值的符号。然后，使用空格分隔包括两个或更多字符串的属性字段内的各个字符串。该字符串属性的转换不是本发明所必需的，在本发明的可选实施例中可以将其省略。

3. 如果具有数据类型“url”的字段属性的 XMTA 属性值以“od://”或“odid://”开始，则指派给相应 mp4bifs 属性的值由“Odid:”给出，其后跟随对象表（见图 39A）中的项 3900 的索引，该项 3900 具有与 XMTA url 属性值（“od://”或“odid://”之后）的剩余部分匹配的 ObjectDescriptorID 3910。

7.1.8 将命令帧插入到 mp4bifs 文档中

在完成对 XMTA“体”元素 120 的第二遍操作 3130 之后，将命令帧元素 2830 的临时排序的列表的内容插入到 mp4bifs 文档 2800 中。丢弃任何空命令帧元素，并将具有相同“time”属性值的多个命令帧元素合并到单个命令帧元素中。

然后，根据最后的命令帧元素 2830 的时间值，更新 mp4file 文档的“moov”元素的“duration”属性的值。指派给该属性的值由从最后的命令帧元素获得的以秒计的值与“moov”元素 2320 的 timeScale 属性的乘积所确定。以类似方式更新 Sdsm 数据的“trak”元素 2350 和 2600 的“duration”属性、和从属于该“trak”元素 2600 的“mdia”元素 2604 的“duration”属性中的每一个。

7.1.9 将 OD 命令插入到 odsM 的 mdat 元素中

如果发现 XMTA 文档包含任何媒体对象，则使用在对 XMTA“体”元素的第一遍操作 3120 中创建的对象表（见图 39A）构建 odsM（对象描述符流）的 XML 描述。如果该表不具有任何项，则该 odsM 不存在，并跳过该步骤。如果该对象表具有至少一个项目，则使用该表创建分类对象表，如图 39E 所示。

分类对象表中的每一项（行）3990 包括对应于该对象表中 **ObjectDescriptorId** 项 3920 的 **OdId** 值 3992、时间值 3994、和布尔标志（开始）3996。

分类对象表包括对象表中每一项 3900 的两项 3990。**OdId** 列 3992 中的每一项的值是在对象表中相应项 3920 中发现的值的副本。时间列 3994 中的项的值是在对象表中的开始时间列 3830 或停止时间列 3940 中的相应项中发现的值的副本。如果在分类对象表的时间列 3994 中的项是从对象表的开始时间列 3930 中的相应项获得的，则向分类对象表的开始列 3996 中的相应项指派值“真”。否则，向分类对象表的开始列 3996 中的该相应项指派值“假”。

分类对象表中的各项以渐增时间值 3994 的顺序存储。在分类对象表创建之后，如图 49 所示，创建 **odsm** 的 XML 表示。

在操作 4900，向整型量“**numSample**”、“**odsmSzie**”和“**sampleSize**”指派值“0”。向浮点型量“**prevTime**”指派一个负值。

在操作 4906，使用标准 XML 手段在 **odsm** 的“**mdat**”元素 2310 和 2400 中定位“**odsmChunk**”元素 2470。使用标准 XML 手段在先前为 **odsm** 创建的“**trak**”元素 2350 和 2600 内定位“**stts**”元素 2660、“**stsz**”元素 2668、和“**stsc**”元素 2656。使用标准 XML 手段定位从属于该“**stsc**”元素 2656 的“**sampleToChunk**”元素。这些元素全部已在先前对 **XMTA**“头”元素 3116 进行处理时创建。

在操作 4910，向索引“**i**”指派值“0”，其不同于其他部分定义的其他索引值。

在操作 4916，将索引“**i**”的值与量“**numEntries**”的值进行比较。量 **numEntries** 的值指示分类对象标中的行数。

在操作 4940，如果索引“**i**”的值不等于 **numEntries** 的值，则将分类对象表中时间列 3994 中的第 **i** 项的值与量 **prevTime** 的当前值进行比较。

在操作 4946，如果分类对象表中的时间列 3994 中的第 **i** 项的值大于量 **prevTime** 的当前值，则使用标准 XML 手段创建新的 **mp4file**

odsmSample 元素。否则，处理继续操作 4970。

然后，使用标准 XML 手段将新的 **odsmSample** 元素插入到在操作 4906 获得的 **odsmChunk** 元素中。将量 **odsmSzie** 的当前值指派给新 **odsmSample** 元素“offset”属性，并将分类对象表中的当前项（“i”）的“时间”列 3994 的值指派给新 **odsmSample** 元素的“时间”属性。

在操作 4950，将索引“i”的值与“0”进行比较。

在操作 4956，如果索引“i”的值大于 0，则使用标准 XML 手段创建新的 **mp4file timeToSample** 元素。否则，处理继续操作 4966。

然后，使用标准 XML 手段将该新 **timeToSample** 元素插入到在操作 4906 获得的 **stts** 元素中。将分类对象表中的当前项的时间值 3994 与量“prevTime”的值之间的差指派给 **timeToSample** 元素的“duration”属性。向新的“timeToSample”元素的“numSample”属性指派值“1”。

使用标准 XML 手段创建新的 **mp4file sampleSize** 元素。然后，使用标准 XML 手段将新的 **sampleSize** 元素插入到在操作 4906 获得的 **stsz** 元素中。将量 **sanpleSize** 的值指派给新“sampleSize”元素的“size”属性。

在操作 4960，将量 **odsmSize** 的值递增量 **sampleSize** 的值，向量 **sampleSize** 的值指派值“0”，并将量 **numSample** 的值递增“1”。

在操作 4966，将分类对象表中的时间列 3994 中的第 i 项的值指派给量“prevTime”。

在操作 4970，将分类对象表中的开始列 3996 中的第 i 项的值与值“真”进行比较。

在操作 4980，如果分类对象表中的开始列 3996 中的第 i 项的值具有值“真”，则使用标准 XML 手段创建新的 **mp4file ObjectDescriptorUpdate** 元素 2540。然后，使用标准 XML 手段将新的 **ObjectDescriptorUpdate** 元素 2540 插入到在操作 4946 创建的 **odsmSample** 元素 2510 中。

使用标准 XML 手段创建新的 **mp4file ObjectDescriptor** 元素

2550。然后，使用标准 XML 手段将新的 ObjectDescriptor 元素 2550 插入到新的 ObjectDescriptorUpdate 元素 2540 中。将与分类对象表中的当前项相关的量“OdId”3992 的值指派给新的 ObjectDescriptor 元素 2950 的“OdId”属性。

使用标准 XML 手段创建新的 mp4file EsIdRef 元素 2560。然后，使用标准 XML 手段将新的 EsIdRef 元素 2560 插入到新的 ObjectDescriptor 元素 2550 中。将与匹配分类对象表中当前项的 OdId 值 3993 的“OdId”值 3920 相关的对象表（见图 39A）中的操作 3950 中的“EsId”项的值指派给“EsIdRef”元素 2560 的“EsId”属性。

在操作 4986，将量 sampleSize 的值递增“10”。

在操作 4990，如果分类对象表中的开始列 3996 中的第 i 项的值不具有值“真”，则使用标准 XML 手段创建新的 mp4file ObjectDescriptorRemove 元素 2570。然后，使用标准 XML 手段将新的 ObjectDescriptorRemove 元素 2570 插入到在操作 4946 中创建的 odsmSample 2510 中。将与分类对象表中的当前项相关的量“OdId”3992 的值指派给新的 ObjectDescriptorRemove 元素 2950 的“OdId”属性。

在操作 4996，将量 sampleSize 的值递增“4”。

在操作 4936，将索引“i”的值递增“1”，并重复索引“i”与值 numEntries 的比较。

在操作 4920，如果索引“i”的值等于 numEntries 的值，则将量“odsmSize”的值递增 sampleSize 的值，并将量 numEntries 的值递增“1”。

在操作 4926，使用标准 XML 手段创建新的 mp4file timeToSample 元素。然后，使用标准 XML 手段将新的 timeToSample 元素插入到在操作 4906 中获得的 stts 元素中。将分类对象表中的当前项的时间值 3994 与量“prevTime”的值之间的差指派给新的 timeToSample 元素的“duration”属性。向新的“timeToSample”元素的“numSamples”属性指派值“1”。

使用标准 XML 手段创建新的 mp4file sampleSize 元素。然后，使用标准 XML 手段将新的 sampleSize 元素插入到在操作 4906 中获得的 stsz 元素中。将量 sampleSize 的值指派给新的“sampleSize”元素的“size”属性。

在操作 4930，将量“numSamples”的值指派给“sampleToChunk”元素。量 odsmSize 的值指派给“odsmChunk”元素的“size”属性。

7.1.10 更新 mp4-bifs 和 mp4-file 文档的 bifsConfig

确定表示 BIFS 节点 Id 表(见图 39B)中的项数所需的最小位数，并将其指派给量“numNodeIdBits”。这是最小数“n”，使得 2 增加到其“n”次乘方后大于该表中的项数。将量 numNodeIdBits 的值指派给步骤 2 中创建的“bifsConfig”元素 (2810) 的“nodeIdBits”属性。还将该值指派给包含在步骤 4 中创建的 sdsm(情景描述流)的“trak”元素 2350 和 2600 中的“BIFS_DecoderConfig”元素 2720 的“nodeIdBits”属性。

以相同方式，确定表示 BIFS 路由 ID 表(见图 39C)中项数所需的最小位数，并将其指派给量“numRouteIdBits”。将量 numRouteIdBits 的值指派给在步骤 2 中创建的“bifsConfig”元素 2810 的 routeIdBits 属性。还将该值指派给包含在步骤 4 中创建的 sdsm(情景描述流)的“trak”元素 2350 和 2600 中的“BIFS_DecoderConfig”元素 2720 的 routeIdBits 属性。

该步骤完成 mp4-file 和 mp4-bifs 文档的创建。该创建 mp4 二进制文件的过程继续“3.b 根据中间 XML 文档创建 mp4 二进制文件”。

7.1.10.1 处理 ES 描述符

如图 34 中所示的那样对每一个“ES_Descriptor”元素进行处理。该过程用于处理包含在 XMT-A 文档 100 的体元素 120 内的 ES_Descriptor 元素 630 以及包含在 XMT-A 文档 100 的头元素 110 内的 ES_Descriptor 元素 180 和 190。

每一个“ES_Descriptor”元素拥有称作“ES_ID”和属性，且将该属

性的值指派给字符串量“ES_DescriptorId”。

该过程“处理 ES_Descriptor”开始于操作 3400 的过程“处理 decConfigDescr 元素”。该过程包括下列四个步骤：

1. 使用标准 XML 手段获得从属于 ES_Descriptor 元素 640 的 decConfigDescr 元素 646。

2. 使用标准 XML 手段获得从属于 decConfigDescr 元素 646 的 DecoderConfigDescriptor 元素 650。

3. 使用 DecoderConfigDescriptor 元素 650 的“流类型”属性的值建立用于该 ES_Descriptor 元素所描述的数据流的流类型属性的数字值。“流类型”属性的值可包括数字值或 MPEG-4 系统规范的表中定义的一组字母数字串中的一个。这些所定义的串包括“ObjectDescriptor”、“SceneDescriptor”、“Visual”、“Audio”等。如果“流类型”属性的值与这些串中的一个匹配，则根据 MPEG-4 表中的相关项向流类型指派一个数字值。例如，如果“流类型”属性的值为“ObjectDescriptor”，则向 iStreamType 指派值 1。否则，“流类型”的值必须代表一个数字值，并将该数字值指派给该流的流类型属性。

4. 使用 DecoderConfigDescriptor 元素的“objectTypeIndication”属性建立用于该 ES_Descriptor 元素所描述的数据流的“objectType”属性的数字值。“objectTypeIndication”属性的值可包括一个数字值或 MPEG-4 系统规范的表中定义的一组字母数字串中的一个。这些所定义的串包括“MPEG4Systems1”、“MPEG4Visual”、“MPEG4Audio”、“Unspecified”等。如果“objectTypeIndication”属性的值与这些串中的一个匹配，则根据 MPEG-4 表中的相关项向 iObjectType 指派一个数字值。例如，如果“objectTypeIndication”属性的值为“Unspecified”，则向 iObjectType 指派值 255。否则，“objectTypeIndication”属性的值必须表示一个数字值，并将该数字值指派给该流的 objectType 属性。

在过程“处理 decConfigDescr 元素”（3400）之后，过程“处理 ES_Descriptor”继续操作 3410 中的过程“处理 slConfigDescr 元素”。

该过程包括下列三个步骤:

1. 使用标准 XML 手段获得从属于“ES_Descriptor”元素 640 的“slConfigDescr”元素 660。
2. 然后使用标准 XML 手段获得从属于“slConfigDescr”元素 660 的“SLConfigDescriptor”元素 666。
3. 使用“SLConfigDescriptor”666 的“timeStampResolution”属性的值向该流的 timeScale 属性指派一个数字值。如果未对“timeStampResolution”属性指定一个值, 则向 timeScale 指派一个默认值。该默认值对于除 MPEG-4 视频(iStreamType = 4, 和 iObjectType = 32)之外的所有流都是 1000, 在 MPEG-4 视频情况下, 默认 timeScale 值是 30。

在操作 3410 的过程“处理 slConfigDescr 元素”之后, 过程“处理 ES_Descriptor”继续操作 3420 中的过程“处理 StreamSource 元素”。

在包含在 XMT-A 体元素 120 中的 ES_Descriptor 630 的情况下, 过程“处理 ES_Descriptor”包括下列两个步骤:

1. 使用标准 XML 手段获得从属于“ES_Descriptor”元素的“StreamSource”元素。
2. 将该“StreamSource”元素的“url”属性的值指派给称作“mediaFileName”的量。

在包含在 XMT-A 头元素 110 的 ES_Descriptor 180 和 190 的情况下, 将不会存在 StreamSource 元素, 且将量“sdsmFileName”的值指派给量“mediaFileName”。

在操作 3420 中的过程“处理 StreamSource 元素”之后, 过程“处理 ES_Descriptor”继续操作 3430 中的过程“为指定流创建 mdat 元素”。如图 35 所示, 过程“为指定流创建 mdat 元素”包括下列步骤:

1. 操作 3500: 使用标准 XML 手段创建新的“mdat”元素 2310, 并将其插入到先前创建的“moov”元素 2320 之前的 mp4file 文档 2300 中。
2. 操作 3506: 将量“nextTrackId”的当前值指派给新 mdat 元素

2320 的“mdatId”属性。向该元素的“size”属性指派 0 值（“0”）。

3a.操作 3510: 将过程“处理 decConfigDescr 元素”操作 3400 所建立的流类型属性与值“1”进行比较。

4a.如果流类型属性的值为“1”，则在操作 3516 创建新的“odsm”元素 2420 和 2460，并将其插入到新的“mdat”元素 2310 和 2400 中，在操作 3520 将量“nextTrackId”的当前值指派给该新“odsm”元素 2420 的“trackID”属性，在操作 3526 创建新的“odsmChunk”元素 2470 并将其插入到新的“odsm”元素 2460 中，在操作 3530 向新的“odsmChunk”元素 2470 的“offset”属性指派值 0。

3b.操作 3540: 如果流类型属性的值不为“1”，则将流类型属性的值与值“3”进行比较。

4b.如果流类型属性的值为“3”，则在操作 3546，创建新的“sdsm”元素 2410 和 2440，并将其插入到“mdat”元素 2310 和 2400 中。在操作 3550，将量“nextTrackId”的当前值指派给该新的 sdsm 元素 2410 的“trackID”属性，并将量“mediaFileName”的值指派给新的 sdsm 元素 2410 的“xmlFile”属性。在操作 3556，创建新的“chunk”元素 2450，并将其插入到新的“sdsm”元素 2440 中。在操作 3560，向新的“chunk”元素 2450 的“offset”属性指派值 0。

4c.操作 3566: 如果流类型的值不是“1”，也不是“3”，则创建新的“mediaFile”元素 2430 和 2480，并将其插入到新的“mdat”元素 2310 和 2400 中。在操作 3570，将量“nextTrackId”的当前值指派给新的“mediaFile”元素 2430 的“trackID”属性。在操作 3576，创建新的“chunk”元素 2490，并将其插入到新的“mediaFile”元素 2480 中。在操作 3580，向新的“chunk”元素 2480 的“offset”属性指派值 0。

向“offset”属性指派值 0 的处理即操作 3530、3560 和 3580 完成过程“为指定流创建 mdat 元素”3430。在该过程 3430 之后，过程“处理 ES_Descriptor”继续过程“为指定流创建 trak 元素”3440。该过程下面在“创建 trak 元素”标题下描述。该过程 3440 之后，过程“处理 ES_Descriptor”3340 继续测试“指定流是 sdsm 或 odsm？”3450。

如果当前流为 **odsm**（流类型的值为 1）或 **sds**m（流类型的值为 3），则在操作 3460，创建新的“**EsIdInc**”元素 2380，并将其添加到 **mp4file** 文档 2300 中的“**mp4fi**ods”元素 2360。然后，将量“**nextTrackID**”的值指派给新的“**EsIdInc**”元素 2380 的“**trackID**”属性 2390。

否则（量“流类型”的值不是“1”或“3”），则在操作 3470，将量“**nextTrackID**”的值添加到 **odsm** 的“**trak**”元素 2600 中的“**tref**”元素 2636 中的“**mpod**”元素 2640 的“**trackID**”元素的值中。还将量“**nextTrackID**”的值指派给在对该 **XMT-A**“体”元素的第一遍中创建的对象表的“**OdId**”列 3920 中的一项。该项对应于其中的“**ObjectDescriptorID**”项 3910 与包含该“**ES_Descriptor**”元素 636 的“**ObjectDescriptor**”元素 600 的“**objectDescriptorId**”属性 606 匹配的行。将量“**nextEsId**”的值指派给该表的同名同一行中的 **EsId** 项 3950。然后，将量“**nextRsId**”的值递增 1。

然后，无论在哪种情况下，在操作 3480，将量 **nextTrackID** 的值递增 1，并将量 **nextTrackID** 的新值指派给“**moov**”元素 2320 的“**nextTrackID**”属性。

这完成了图 34 中示出的对“**ES_Descriptor**”元素的处理。对于在 **XMT-A** 文档 100 的“头”元素 110 中的“**InitialObjectDescriptor**”元素 130 内的“**Descr**”元素 160 内发现的每一个“**ES_Descriptor**”元素 180 和 190 执行该过程。还对在 **XMT-A** 文档 100 的“体”元素 120 内发现的“**ObjectDescriptor**”元素 600 内的“**Descr**”元素 610 内发现的每一个“**ES_Descriptor**”元素 630 执行该过程。

在该过程（处理 **ES_Descriptor**）完成之后，创建 **mp4file** 文档 2250 的过程继续图 33 中示出的过程“处理 **esDescr** 元素”中的操作 3350 递增索引“**i**”的值的步骤。

7.1.10.2 创建 trak 元素

如图 36A 所示，过程“为指定流创建 **trak** 元素”包括下列 11 个步

骤:

1. 在操作 3600, 使用标准 XML 手段创建新的“trak”元素 2350 和 2600, 并将其插入到 mp4file 文档 2300 中的“moov”元素 2320 中。

向新的 trak 元素 2600 的下列属性指派值: 向“flags”属性指派值“1”。将等于自从 1904 年 1 月 1 日以来的秒数的值指派给“creationTime”和“modifyTime”属性。将量 nextTrackId 的值指派给 trackID 属性。向“trackHeight”属性指派值“240”。向“trackWidth”属性指派值“320”。

如果流类型属性的值为“1”或“3”, 则向 duration 属性指派值“0”。这些仅是初始值, 其将由后面确定的校正值所替代。否则, 使用所附的 ObjectDescriptor 元素 600 的 objectDescriptorID 获得在对 XMT-A 文档 100 的体元素 120 的第一遍操作 3120 期间构建的表中的相应媒体持续时间值。将该媒体持续时间值(以秒)与从“SLConfigDescriptor”元素 666 中获得的时窗值相乘, 并四舍五入为整数值。

如果流类型属性的值为“1”(对象描述符流), 则将 trackID 属性的值指派给量 trackIdForOdsM。如果流类型属性的值为“3”(情景描述流), 则将 trackID 属性的值指派给量 trackIdForSdsM。

2. 在操作 3606, 使用标准 XML 手段创建新的“mdia”元素 2604, 并将其插入到在步骤 1 中创建的新“trak”元素 2600 中。

向该新“mdia”元素 2604 的下列属性指派值: 将等于自从 1904 年 1 月 1 日以来的秒数的值指派给“creationTime”和“modifyTime”属性。这是用于父 trak 元素 2600 的相应属性的相同值。将从“SLConfigDescriptor”元素 666 中获得的时窗值指派给“timeScale”属性。将指派给父 trak 元素 2600 的持续时间值指派给持续时间属性。

3. 在操作 3610, 使用标准 XML 手段创建新的“hdlr”元素 2608, 并将其插入到在步骤 2 中创建的新“mdia”元素 2604 中。

向“hdlr”元素的下列属性指派值: “handlerType”和“name”。指派给“handlerType”属性的值取决于流类型。如果流类型等于 1 (osdm)、3 (sdsM)、4 (视频流)、或 5 (音频流), 则将值“odsm”、

“sdsm”、“soun”或“vide”指派给“handlerType”属性。否则，将值“none”指派给“handlerType”属性。指派给“name”属性的值是由所附 XMT-A ES_Descriptor 元素 180、190 或 630 的 ES_ID 属性确定的字符串“Es_DescriptorId”的副本。这种对于“name”属性的选择是不必要的，但这种选择使得在该 mp4 文档和随后的文件中保存和传播 ES_ID 属性字符串的值成为可能。

4. 在操作 3616，使用标准 XML 手段创建新的“minf”元素 2612，并将其插入到在步骤 2 中创建的新“mdia”元素 2604 中。

5. 在操作 3620，使用标准 XML 手段创建新的“dinf”元素 2616，并将其插入到在步骤 4 中创建的新“minf”元素 2612 中。

6. 在操作 3626，使用标准 XML 手段创建新的“dref”元素 2620，并将其插入到在步骤 5 中创建的新“dinf”元素 2616 中。

7. 在操作 3630，使用标准 XML 手段创建新的“urlData”元素 2624，并将其插入到在步骤 6 中创建的新“dref”元素 2620 中。向“urlData”元素 2624 的“flags”属性指派值“1”。

8. 然后，在操作 3636，使用标准 XML 手段创建新的“stbl”元素 2628 和 2652，并将其插入到在步骤 4 中创建的“minf”元素 2612 中。如下面在“创建初始样本表元素”之下描述的那样，创建组成的样本表元素。

9. 在操作 3640，使用标准 XML 手段创建新的媒体头元素 2632，并将其插入到在步骤 4 中创建的“minf”元素 2612 中。该媒体头元素的元素名取决于该流的流类型属性：

如果流类型属性为 1（odsm）或 3（sdsm），则该媒体头元素为不具有属性的“nmhd”元素。

如果流类型属性为 4（视频流），则该媒体头元素为具有值“0”的属性“transferMode”的“vmhd”元素。

如果流类型属性为 5（音频流），则该媒体头元素为具有值“0”的属性“balance”的“smhd”元素。

否则，该媒体头元素为具有值“0”的属性“transferMode”和值“0”

的属性“balance”的“gmhd”元素。

10. 在操作 3646, 将流类型属性的值与值 4 和 5 进行比较, 并将量 `startTime` 的值与 0 进行比较。

在音频或视频流的情况下, 该操作在过程“处理 XMT-A 体元素 (第 2 遍)”3130 期间执行。在这种情况下, 从在过程“处理 XMT-A 体元素 (第 1 遍)”3120 中创建的对象表 (见图 39A) 中获得量 “`startTime`”的值。该值由某一行的开始时间列中的项所确定, 其中该行中 `ObjectDescriptorID` 列的项与包含该“ES_Descriptor”元素的“ObjectDescriptor”元素的“`objectDescriptorId`”属性相匹配。

在 `odsm` 和 `sds` 的情况下, 在过程“处理 XMT-A 头元素”3116 期间执行该操作。还未对那些流创建用于建立量开始时间的值的对象表。因此, 作为单独步骤“创建 `odsm` 的编辑列表”3126 的一部分, 执行对 `odsm` 流的开始时间值的相应测试。`Sds` 通常开始于时间 0。

如果流类型的值为 4 (视频流) 或 5 (音频流), 且该流的量 “`startTime`”的值不为 0, 则在操作 3650, 使用标准 XML 手段创建新的“`edts`” (编辑列表) 元素 2644, 并将其插入到当前“`trak`”元素 2400。使用标准 XML 手段创建新的“`elst`”元素 2648, 并将其插入到新的“`edts`”元素 2644。然后, 创建两个“`segment`”元素, 并将其插入到“`elst`”元素 2648 中。

向每一个“`segment`”元素指派称作“`startTime`”、“`duration`”和“`rate`”的属性。向第一个 `segment` 元素的“`startTime`”属性指派值“-1”。向第二个 `segment` 元素的“`startTime`”属性指派值“0”。向这两个 `segment` 元素的“`rate`”属性指派值“1.0”。向第一个 `segment` 的“`duration`”属性的值指派一个值, 该值由该流的 `startTime` 值和封装的 `moov` 元素的“`timeScale`”属性的值的乘积所确定。向第二个 `segment` 的“`duration`”属性的值指派一个值, 该值由该流的持续时间值和封装 `moov` 元素的“`timeScale`”属性的值的乘积确定。该流的持续时间值由从对象表 (见图 39A) 获得的“`stopTime`”值和“`startTime`”值之间的差所确定, 其中该对象表是在对 XMT-A“体”元素的第一遍操作中创建

的。这些值由某一行的相应列中的项所确定，其中该行中的 **ObjectDescriptorID** 列的项与包含该“ES_Descriptor”元素的“ObjectDescriptor”元素的“objectDescriptorId”属性相匹配。

11. 在操作 3656，将流类型属性的值与“1”进行比较。如果流类型为 1 (odsm)，则在操作 3660，使用标准 XML 手段创建新的“tref”元素 2636，并将其插入到在步骤 1 中创建的“trak”元素 2600。创建新的“mpod”元素 2640，并将其插入到新的“tref”元素 2636 中。向“mpod”元素 2640 的“trackID”属性指派值“-1”。这是要由以后获得的数据所替代的临时值。

步骤 3656 完成过程“为指定流创建 trak 元素”3440。在该过程之后，过程“处理 ES_Descriptor”3340 继续测试“指定流是 sdsm 或 odsm?”3450。

7.1.10.3 创建初始样本表元素

最终 mp4 二进制文件 2230 中的每一个样本表包含依赖 sdsm、odsm 和媒体数据文件的二进制形式的信息。确定这些表中的值所需的信息此时是不可获得的。因此，如图 36B 中所示，创建这些表的初始表示，以指示当创建实际 mp4 二进制文件 2230 时最终值将被放置的位置。

在操作 3666，使用标准 XML 手段创建新的“stsc”元素 2656，并将其插入到当前 trak 元素 2600 的“stbl”元素 2628 和 2652。使用标准 XML 手段创建新的“sampleToChunk”元素，并将其插入到新的“stsc”元素 2656 中。向新的“sampleToChunk”元素的“sampleDesc”属性指派值“1”。向新的“sampleToChunk”元素的“firstChunk”属性指派值“1”。如果该流的流类型属性为 1 或 3 (odsm、sdsm)，则向新的“sampleToChunk”元素的“numSamples”属性指派值“0”。如果该流的 objectType 属性为 108 (JPEG 图像)，则向新的“sampleToChunk”元素的“numSamples”属性指派值“1”。否则，向新的“sampleToChunk”元素的“numSamples”属性指派值“-1”。

在操作 3670, 使用标准 XML 手段创建新的“stts”元素 2660, 并将其插入到当前 trak 元素 2600 的“stbl”元素 2628 和 2652 中。如果当前流类型属性不为 1 (odsm), 也不为 3 (sdsm), 则使用标准 XML 手段创建新的“timeToSample”元素, 并将其插入到新的“stts”元素 2660 中。将该“trak”元素中指定的持续时间值指派给“timeToSample”元素的“duration”属性。如果该流的对象类型属性为 108 (JPEG 图像), 则向该“timeToSample”元素的“numSamples”属性指派值“1”。否则, 向该新的“timeToSample”元素的“numSamples”属性指派值“-1”

在操作 3676, 使用标准 XML 手段创建新的“stco”元素 2664, 并将其插入到当前 trak 元素 2600 的“stbl”元素 2628 和 2652 中。使用标准 XML 手段创建新的“chunkOffset”元素, 并将其插入到新的“stco”元素 2664 中。将 nextTrackId 的当前值指派给该“chunkOffset”元素的“mdatId”属性。向该“chunkOffset”元素的“mdatOffset”属性指派值“0”。向该“chunkOffset”元素的“offset”属性指派值“8”。

在操作 3680, 使用标准 XML 手段创建新的“stsz”元素 2668, 并将其插入到当前 trak 元素 2600 的“stbl”元素 2628 和 2652 中。如果该流的流类型属性不为 1 (odsm), 也不为 3 (sdsm), 则向该新“stsz”元素 2668 的“numSamples”属性指派一个值。如果该流的对象类型属性为 108 (JPEG 图像), 则向新的“stsz”元素 2668 的“numSamples”属性指派值“1”。否则, 向新的“stsz”元素 2668 的“numSamples”属性指派值“-1”。

在操作 3686, 如果流类型属性为 1 (odsm) 或 3 (sdsm), 则使用标准 XML 手段创建新的“stss”元素 (2672), 并将其插入到当前 trak 元素 2600 的“stbl”元素 2628 和 2652 中。如果流类型为 1, 则向该新“stss”元素 2672 的“numEntries”属性指派值“1”, 并创建新的“syncSample”元素, 并将其插入到新“stss”元素 2672 中。然后, 向该“syncSample”元素的“sampleNumber”属性指派值“0”。如果流类型属性为 3, 则向新的“stss”元素 2672 的“numEntries”属性指派值“0”。

如果流类型属性为 4, 且对象类型属性为 32 (MPEG-4 视频),

则使用标准 XML 手段创建新的“stss”元素 2672, 并将其插入到当前 trak 元素 2600 的“stbl”元素 2628 和 2652 中, 并向新的“stss”元素 2672 的“numEntries”属性指派值“-1”。

在操作 3690, 使用标准 XML 手段创建新的“stsd”元素 2676, 并将其插入到当前 trak 元素 2600 的“stbl”元素 2628 和 2652 中。创建该新“stsd”元素 2676 内的从属元素, 如图 37 中所示的那样。

在操作 3700, 使用标准 XML 手段创建新的“esds”元素 2684。

在操作 3706, 将流类型属性的值与“1”和“3”进行比较。

在操作 3710, 如果流类型属性的值为“1”或“3”, 则使用标准 XML 手段创建新的“mp4s”元素 2680, 并将其插入到当前“stsd”元素 2676 中。将该新“esds”元素插入到新的“mp4s”元素 2680 中, 并向该新“mp4s”元素 2680 的“dataRefIndex”属性指派值“1”。

在操作 3716, 如果流类型属性的值不为“1”或“3”, 则将流类型属性与“4”进行比较。

在操作 3720, 如果流类型的值为 4, 则使用标准 XML 手段创建新的“mp4v”元素 2680, 并将其插入到当前“stsd”元素 2676 中。将新的“esds”元素 2684 插入到新的“mp4v”元素 2680 中, 并向新“mp4v”元素 2680 的“colorTable”、“dataRefIndex”、“frameCount”、“horizontalRes”、“verticalRes”、“pixelDepth”、“height”和“width”分别指派值“-1”、“1”、“1”、“72.0”、“72.0”、“24”、“240”和“320”。

在操作 3726, 如果流类型的属性值不为“4”, 则将流类型的属性与“5”比较, 如果流类型的属性值不为“5”, 完成了该过程(操作 3690)。

在操作 3730, 如果流类型的值为 5, 则使用标准 XML 手段创建新的“mp4a”元素 2680, 并将其插入到当前“stsd”元素 2676 中。将新的“esds”元素 2684 插入到新的“mp4a”元素 2680 中, 并向新“mp4a”元素 2680 的“dataRefIndex”、“numchannels”及“sampleSize”分别指派值“1”、“-1”和“1”。

可容易地处理另外的流类型情况, 但这些情况(1、3、4、5)仅是当前实现所需的那样情况。如果流类型属性的值为1、3、4或5, 则

对stsd元素2676不执行其他处理。

在操作3736, 如果流类型属性的值为“1”、“3”、“4”或“5”, 则使用标准XML手段创建新的“Es_Descr”元素2688, 并将其插入到当前“esds”元素2684中。向新“ES_Descr”元素2688的“ES_ID”属性指派值“0”。向新“ES_Descr”元素2688的“优先级”属性指派值“0”。

在操作3740, 使用标准XML手段创建新的“DecoderConfigDescriptor”(D-C-D)元素2710, 并将其插入到当前“ES_Descr”元素2676中。从该流的XMT-A“DecoderConfigDescriptor”650中获得“bufferSizeDB”、“avgBitrate”和“maxBitrate”属性的值, 并将这些属性的值指派给新“DecoderConfigDescriptor”元素2710的“bufferSize”、“avgBitrate”和“maxBitrate”属性。将当前流的流类型、和对象类型属性的值指派给新“DecoderConfigDescriptor”元素2710的“streamType”和“objectType”属性。

在操作3746, 使用标准XML手段创建新的“SLCofigDescriptor”(SLC-D)元素2760, 并将其插入到当前“ES_Descr”元素2676中。向新的“SLCofigDescriptor”元素2760的“predefined”属性指派值“2”。

根据流类型和对象类型属性的值, 可将解码器详细信息元素插入到“DecoderConfigDescriptor”元素2710中。如果流类型属性的值为1(odsm), 则无需解码器详细信息元素。

在操作3750, 将流类型属性的值与“3”进行比较。

在操作3756, 如果流类型属性的值为3(sdsm), 则执行过程“处理BIFS配置”。该过程将在下面描述。

在操作3760, 将对象类型属性的值与“32”进行比较。

在操作3766, 如果对象类型属性的值为32(MPEG-4视频), 则使用标准XML手段创建新的“VisualConfig”元素2740, 并将其插入到当前“DecoderConfigDescriptor”元素2710中。

在操作3770, 将对象类型属性的值与“64”进行比较。

在操作3776, 如果对象类型属性的值为64(MPEG-4音频), 则

使用标准XML手段创建新的“AudiioConfig”元素，并将其插入到当前“DecoderConfigDescriptor”元素2710中。

在操作3780，将对象类型属性的值与“108”进行比较。

在操作3786，如果对象类型属性的值为108（JPEG图像），则使用标准XML手段创建“JPEG_DecoderCofig”元素2730中，并将其插入到当前“DecoderConfigDescriptor”元素2710中。

其他的流类型和对象类型情况可以容易地处理，但是这些情况仅是当前实现所需的那样情况。

除了“BIFS_DecoderConfig”元素2720之外，上面说明的解码器详细信息元素2730、2740、和2750仅是存根或占位符。如果流类型属性的值为3（sdsm），则执行过程“处理BIFS配置”，其将在下面描述。否则，对ES_Descriptor元素640的处理继续过程“创建trak元素”3440的步骤9（操作3640）。

7.1.10.4 处理BIFS配置

图38中示出了过程“处理BIFS配置”。

在操作3800，使用标准XML手段创建新的“BIFS_DecoderConfig”元素2720，并将其插入到“DecoderConfigDescriptor”元素2710中。

在操作3810，使用标准XML手段获得mp4bifs文档2800中的“bifsConfig”元素2810。

在操作3820，使用标准XML手段获得从属于sdsm的“decSpecificInfo”元素656和680。然后，使用标准XML手段获得从属于该“decSpecificInfo”元素680的“BIFSConfig”元素686。

在操作3830，向“BIFS_DecoderConfig”元素2720的“nodeIdBits”属性和“bifsConfig”元素2810的相应属性指派值“0”。向“BIFS_DecoderConfig”元素2720的“routeIdBits”属性和“bifsConfig”元素2810的相应属性指派值“0”。

在操作3840，将对象类型属性的当前值与“2”进行比较。

在操作3846，如果对象类型属性的当前值为“2”，则向BIFS_DecoderConfig元素2720的“protoIdBits”属性和bifsConfig元素2810的相应属性指派值“0”，将BIFSConfig元素686的“use3DmeshCoding”和“usePredictiveMFField”属性所确定的值指派给BIFS_DecoderConfig元素2720和bifsCofig元素2810的同名属性。

在操作3850，使用标准XML手段获得从属于BIFSConfig元素（686）的命令流元素690。

在操作3856，如果BIFSConfig元素686不包含从属命令流元素690，则使用标准XML手段获得从属于BIFSConfig元素686的animMask元素。如果BIFSConfig元素不拥有从属animMask元素，则XMT-A文档无效，并在操作3860报告出错。

在操作3866，如果BIFSConfig元素686拥有从属animMask元素，则向BIFS_DecoderConfig元素2720的命令流属性指派值“假”。

在操作3870，如果BIFSConfig元素686拥有从属命令流元素690，则向BIFS_DecoderConfig元素2720的命令流属性指派值“真”。将命令流元素690的pixelMetric属性的值指派给BIFS_DecoderConfig元素2720的pixelMetric属性。如果已对命令流元素690的pixelMetric属性指定了一个值，则向BIFS_DecoderConfig元素2720的pixelMetric属性指派默认值“假”。

然后，在操作3880，使用标准XML手段获得从属于命令流元素690的“size”元素696。

在操作3886，如果命令流元素690不拥有从属size元素696，则向BIFS_DecoderConfig元素2720的“pixelHeight”和“pixelWidth”属性指派值“0”。

在操作3890，如果命令流元素690拥有从属size元素696，则将“size”元素696的“pixelHeight”和“pixelWidth”属性的值指派给BIFS_DecoderConfig元素2720的“pixelHeight”和“pixelWidth”属性。

在这些步骤完成之后，对ES_Descriptor元素640的处理继续过程“创建trak元素”3440的步骤9（操作3640）。

7.2 根据中间XML文档创建mp4二进制文件

在创建中间XML文档2250和2260之后，使用该中间XML文档2250和2260，以及任何相关的媒体数据文件2220创建表示原始XMT-A文档2210中指定的信息的新mp4二进制文件2230。该新mp4文件称作“输出mp4文件”或“mp4文件”。用于创建该新mp4文件的手段包括下列六个步骤：

1. 建立输入文档和输出目的文件；
2. 创建工作数组；
3. 处理“mdat”元素2310；
4. 处理“moov”元素2320；
5. 处理可选用户数据元素2330；以及
6. 更新odsm缓冲区大小。

这些步骤中的每一个将在下面描述。

7.2.1 建立输入文档和输出目的地

这些步骤中的第一个包括获得对表示上面创建的mp4file文档2250和mp4bifs文档2260的XML数据结构的引用。该步骤还包括接收为输出mp4二进制文件2230指定文件名的数据结构。如果所指定的文件名对应于已有文件，则删除此文件。然后，使用所指定的文件名创建新的空输出文件。

在创建空输出文件之后，使用标准XML手段获得mp4file文档的顶级元素。此时还可获得mp4bifs文档的顶级元素，但这到后面才需要。

该新的输出文件（“mp4 file”）将包括称作“mp4原子”和“mp4对象结构”的数据结构的分层集合。在当前的实现中，每一个mp4原子包括32位“size”值、32位“原子ID”、和一组属性值。Mp4原子还可包括一个或多个从属mp4原子或mp4对象结构。size值指出包括大小和原子ID的全部mp4原子中的字节数。Mp4对象结构包括1字节对象结构标签、不定大小的size值、一组属性值、和一组零个或多个从属mp4对象

结构。在这种情况下，size值指出除对象结构标签和size值之外的对象结构中的字节数。

图50中示出了用于创建每一个原子的一般过程。图51中示出了用于创建对象结构的相应过程。这些过程需要有控制输出mp4文件的“文件位置”的能力。“文件位置”定义为从该文件的开始到下一个字节要被写入处的字节数。因为控制文件位置的需要，必须将该新文件打开为“随机存储”或“读/写”类型的文件。

7.2.1.1 创建mp4原子的过程

创建mp4原子的过程包括下列步骤：

1. 在操作5000，向输出文件的当前文件位置指派量“sizePos”。量“sizePos”的值对于每一个mp4原子或对象结构是唯一的。
2. 在操作5010，将32位具有零值的整数写入输出文件。
3. 在操作5020，将32位原子ID值写入输出文件。例如，在“mdat”原子的情况下，将表示字符“m”、“d”、“a”和“t”的ascii值的四种类型写入输出文件。
4. 在操作5030，对当前mp4原子所表示的mp4file元素的属性进行解释。每一个mp4原子所拥有的特定属性集合由原子ID来确定，如MP4文件格式的MPEG-4规范中所指示的那样。为该mp4file文档中未指定的属性提供缺省值。
5. 在操作5040，将当前mp4原子的属性的值写入输出文件。在MP4 file文档的MPEG-4规范中指出了用于表示每一个属性值的位数。
6. 在操作5050，如果当前mp4file元素拥有任何从属元素，则对每一个这样的从属元素进行处理。如果该从属元素对应于mp4file原子元素，则递归重复当前的过程。如果从属元素对应于mp4file对象元素，则执行图51中示出的过程。
7. 在操作5060，向当前文件为止指派量“endPos”。
8. 在操作5070，将量“endPos”的值与量“sizePos”的值之间的差指派给量“size”。

9. 在操作5080, 将输出文件的文件位置改为量“sizePos”的值所指定的位置。

10. 在操作5090, 将表示量“size”的值的32位整数写入输出文件中。

11. 在操作5095, 将输出文件的文件位置改为量“endPos”的值所指定的位置。

7.2.1.2 创建mp4对象结构的过程

创建mp4对象结构的过程包括以下步骤:

1. 在操作5100, 将一字节对象结构标签写入输出文件。该对象结构标签的值由该mp4对象结构所表示的mp4file元素的元素名称和MPEG-4规范中提供的表所确定。

2. 在操作5110, 向输出文件的当前文件位置指派量“sizePos”。量“sizePos”的值对于每一个mp4原子或对象结构是唯一的。

3. 在操作5120, 根据表示mp4对象结构所需的字节数的估算或上界, 向量“numSizeBytes”指派一个值。如果表示mp4对象结构所需的字节数小于128, 则向量“numSizeBytes”指派值“1”。在大多数情况下, 这是足够的。

4. 在操作5130, 将一字节值的序列写入输出文件。这些一字节值的数量由量“numSizeBytes”的值所指出。这些一字节量的值是无意义的, 因为它们随后将被覆写。值0可用于这些字节中的每一个。

5. 在操作5135, 对当前mp4对象元素所表示的mp4file元素的属性进行解释。每一个mp4对象元素所拥有的特定属性组由对象标签所确定, 如MPEG-4系统规范中所指示的那样。为该mp4file文档中未指定的属性提供默认值。

6. 在操作5140, 将当前mp4对象结构的属性的值写入输出文件。在MPEG-4系统规范中指出了用于表示每一个属性值的位数。

7. 在操作5150, 如果当前mp4file元素拥有任何从属元素, 则根据图51中示出的过程对每一个这样的从属元素进行处理(递归)。

8. 在操作5160, 向当前文件位置指派量“endPos”。
9. 在操作5165, 将量“endPos”的值与量“sizePos”的值之间的差指派给量“size”。
10. 在操作5170, 从量“size”的值中减去量“numSizeBytes”的值。
11. 在操作5180, 将输出文件的文件位置改为量“sizePos”的值所指定的位置。
12. 在操作5190, 将表示量“size”的值的一字节值序列写入输出文件。这些一字节值的数量由量“numSizeBytes”的值指出。这些一字节值的每一个的低七位由量“size”的值的相应七位部分所确定。这些一字节值的每一个的高七位的值除了最后的一字节值之外均为“1”。该序列中的最后的一字节值的高位的值为“0”。
13. 在操作5195, 将输出文件的文件位置改为量“endPos”的值所指定的位置处。

7.2.2 创建工作数组

上面所标识的第二步包括根据mp4file文档2250和2300中表示的“trak”元素2350的数量、块元素2450和2490的数量、及odsmSample元素2510的数量, 创建若干工作数组。

下列手段用于确定量“MaxNumTracks”:

使用标准XML手段识别从属于mp4file文档2300的顶级元素的所有元素2310和2320。这些从属元素中的一个“moov”元素2320。该“moov”元素2320的“nextTrackID”属性的值提供从属于“moov”元素2320的“trak”元素2350的数量的上界。如果该mp4file文档是按上面指示的那样创建的, 则“nextTrackID”属性的值指出从属于“moov”元素2320的“trak”元素2350的数量。将“nextTrackID”属性的值指派给量“MaxNumTracks”。

使用量“MaxNumTracks”的值指定下列九个列表中的每一列表中的项数来创建下列九个列表:

1. **MediaSample**
2. **MediaDataFile** (“File”对象的数组)
3. **MediaHeaderSize**
4. **MediaHeader** (位置值的数组)
5. **EsDescrSize**
6. **TrackIdForTrack**
7. **StreamTypeForTrack**
8. **ObjectTypeForTrack**
9. **TrackIdfor OdId**

这些项中除了MediaDataFile项和MediaHeader项之外的每一列表都是整数数组。

在创建该组的九列表之后，向量TrackNum、MaxNumChunks和MaxNumOdsmSample指派值0。

下列手段用于确定列表TrackIdForTrack、StreamTypeForTrack和ObjectTypeForTrack中的各项：

使用标准XML手段识别所有从属于“moov”元素2320的元素。对于每一个这样的“trak”类型的从属元素2350，将“trackID”属性的值指派给TrackIdForTrack列表中的项TrackNum。使用标准XML手段识别从属于该“trak”元素的“DecoderConfigDescriptor”元素2710（通过九级）。将该元素2710的“流类型”属性的值指派给列表StreamTypeForTrack中的项TrackNum。将该元素2710的“对象类型”属性的值指派给列表ObjectTypeForTrack中的项TrackNum。然后，将量TrackNum的值增加1。

使用下述手段确定量“MaxNumChunks”和“MaxNumOdsmSample”的值。

使用标准XML手段识别从属于mp4file文档2300的顶级元素的所有“mdat”元素2310。使用标准XML手段识别从属于每一个“mdat”元素2310和2400的所有元素。所得到的从属元素可包括“mediaFile”元素2430、“sdsm”元素2410、和“odsm”元素2420。使用标准XML手段识

别从属于元素2410、2420和2430中的每一个的每一个“块”元素2450和2490和“odsm块”元素2470，而元素2410、2420和2430从属于每一个“mdat”元素2400。

对于从属于每一个“mdat”元素2400所附属的每一个元素2410、2420和2430的每一个“块”元素2450和2490和每一个“odsm块”元素2470，将量MaxNumChunk的值增加1。

使用标准XML手段识别从属于每一个“odsm块”元素2470和2500的每一个“odsm样本”元素2510。对于从属于每一个“odsmChunk”元素2500的每一个“odsm样本”元素2510，将量MaxNumOdsmSample增加1。

使用量“MaxNumChunks”的值指定下面4个列表的每一个中的项数，来创建这些列表：

1. MdatIdForChunk
2. TrackIdForChunk
3. OffsetForChunk
4. MediaDataSize

这些列表的每一个是整数数组。在这些列表创建之后，向量NumChunks指派值0。

如果量“MaxNumOdsmSample”的值大于0，则使用量“MaxNumOdsmSample”的值指定下列两个列表的每一个中的项数来创建这些列表：

1. OdsmSampleSize
2. OdsmSampleTime

这些列表的每一个都是整数数组。在这些列表创建之后，向量NumOdsmSample指派值0。

7.2.3 处理“mdat”元素

创建输出mp4文件2230中的第三步包括对每一个包括在mp4file文档2300中的mdat元素2310进行处理。

使用标准XML手段识别从属于如图23A中所示mp4file文档2300的顶级元素的每一个mdat元素2310。然后，使用图52中示出的手段对这些“mdat”元素2310中的每一个进行处理。图52中示出的过程是图50中示出的过程的一个例子。

在操作5200，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5212，将具有值0的32位整数写入该输出mp4文件724。在操作5224，将表示字符“m”、“d”、“a”和“t”的ASCII值的四字节写入输出mp4文件730。将该mdat元素的“mdatId”属性的值指派给量“mdatId”。无属性值写入该输出mp4文件。

在操作5236，向索引“i”指派值0。在操作5242，将索引“i”的值与量“numMdatChildren”的值进行比较，其中量“numMdatChildren”表示当前mdat元素所拥有的从属元素的数量。在操作5248，如果索引“i”的值等于量“numMdatChildren”的值，则将mdat原子724的大小更新，如图50的最后五部分中所表示的那样（操作5060到5095）。

在操作5254，如果索引“i”的值不等于量“numMdatChildren”的值，则将标准XML手段用于获得每一个从属于当前mdat元素的XML元素。由“mdatChild”来表示从属于当前mdat元素的第i个XML元素，并将元素mdatChild的元素名表示为“childName”。

在操作5260，将XML元素mdatChild的名称与字符串“mediaFile”进行比较。在操作5266，如果XML元素mdatChild的名称与字符串“mediaFile”匹配，则执行过程“插入媒体文件数据”。在执行过程“插入媒体文件数据”之后，将索引“i”的值增加1（5296），并重复索引“i”的值与量“numMdatChildren”的值的比较操作5242。

在操作5272，如果XML元素mdatChild的名称与字符串“mediaFile”不匹配，则将XML元素mdatChild的名称与字符串“odsm”进行比较。在操作5278，如果XML元素mdatChild的名称与字符串“odsm”匹配，则执行过程“插入Odsm数据”。在执行过程“插入Odsm数据”之后，在操作5296将索引“i”的值增加1，并重复索引“i”的值与量“numMdatChildren”的值的比较操作5242。

在操作5284，如果XML元素mdatChild的名称与字符串“odsm”不匹配，则将XML元素mdatChild的名称与字符串“sdsm”进行比较。在操作5290，如果XML元素mdatChild的名称与字符串“sdsm”匹配，则执行过程“插入sdsm数据”。在执行过程“插入sdsm数据”之后，在操作5296将索引“i”的值增加1，并重复索引“i”的值与量“numMdatChildren”的值的比较操作5242。

如果XML元素mdatChild的名称与字符串“sdsm”不匹配，则在操作5296将索引“i”的值增加1，并重复索引“i”的值与量“numMdatChildren”的值的比较操作5242。

7.2.3.1 插入媒体文件数据

图53中示出的过程“插入媒体文件数据”5266用于处理从属于“mdat”元素2400的“媒体文件”元素2430。在操作5300，将“媒体文件”元素2430的“trackId”属性的值指派给量“trackId”。在操作5306，将“媒体文件”元素2430的“名称”属性的值指派给量“mediaFileName”。

在操作5312，量“trackNum”的值由TrackIdForTrack列表中匹配量“trackId”的值的项的索引确定。在操作5318，将列表Track的流类型和列表Track的对象列表中相应项（具有索引trackNum）的值指派给量“streamType”和“objectType”。

在操作5324，为mediaFileName量的值所标识的媒体数据文件创建新的“File”对象。在操作5330，将该对象保存为媒体数据文件列表中具有量trackNum的值所确定的索引的项。在操作5336，获得定义为包括该媒体数据文件的字节数的该媒体数据文件的大小，作为该新File对象的长度属性。将该大小值指派给量“mediaFileSize”。在操作5342，将量“MediaHeaderSize”的值初始化为0。

在操作5348，向索引“i”指派值0。在操作5354，将索引“i”的值与量“numMediaFileChildren”的值进行比较，其中量“numMediaFileChildren”的值由从属于当前mediaFile元素2430的XML元素的数量所确定。

在操作5360, 如果索引“i”的值等于量“numMediaFileChildren”的值, 则计数媒体数据文件中的样本数。用于计数媒体数据文件中的样本的手段依赖于“streamType”和“objectType”的值, 及每一个特定类型的媒体数据文件的详细文件结构规范。这些手段不是本发明所特有的, 且这里未呈现。在操作5366, 在对媒体数据文件中的样本数进行计数之后, 将所得到的样本技术保存为MediaSample列表中具有量trackNum的值所确定的索引的项。

在操作5372, 如果量“i”的值不等于量“numMediaFileChildren”的值, 则使用标准XML手段获得从属于当前mediaFile元素2480的每一个XML元素。从属于当前mediaFile元素的第i个XML元素由“mediaFileChild”所表示, 且元素mediaFileChild的元素名由“childName”所表示。

在操作5384, 将XML元素mediaFileChild的名称与字符串“chunk”进行比较。

在操作5390, 如果XML元素mdatChild的名称与字符串“chunk”匹配, 则执行过程“插入媒体数据块”。在执行过程“插入媒体数据块”之后, 在操作5396将索引“i”的值增加1, 并重复索引“i”的值与量“numMediaFileChildren”的值的比较操作5354。

如果XML元素mediaFileChild的名称与字符串“chunk”不匹配, 则在操作5396将索引“i”的值增加1, 并重复索引“i”的值与量“numMediaFileChildren”的值的比较操作5354。

7.2.3.2 插入媒体数据块

过程“插入媒体数据块”5390主要包括将媒体数据文件2220的内容添加到输出mp4文件2230。由量“streamType”和“objectType”的值所确定的某种类型的媒体数据可以初始“头”数据部分开始。这些包括“MPEG-4视频”(流类型=4且对象类型=32)。识别特定类型的媒体数据的头数据部分的适当手段依赖于每一类型的媒体数据文件的详细规范。这些文件规范在本发明的范围之外, 且这里未涵盖。参见

ISO/IEC文档14496-2 (1999, 2000年修改)“信息技术-音频-视频对象的编码-第2部分: 视频”中对于MPEG-4视频流的描述。

在将媒体数据从媒体数据文件复制到mp4二进制文件之前, 执行下列操作:

1. 将在操作 5230 中确定的量“mdatId”的值指派给列表“MdatIdForChunk”中的项“NumChunks”。

2. 将在操作 5300 中确定的量“trackId”的值指派给列表“TrackIdForChunk”中的项“NumChunks”。

3. 将输出 mp4 文件中的当前文件位置的值指派给列表“OffsetForChunk”中的项“NumChunks”。

4. 将在操作 5336 中确定的量“mediaFileSize”的值指派给列表“MediaDataSize”中的项“NumChunks”。

5. 向列表“MediaHeaderSize”中的项“trackNum”指派值0。

6. 如果量“streamType”和“objectType”的值所指出的媒体文件类型包括初始头数据部分, 则将包括此头部分的字节数指派给列表“MediaHeaderSize”中的项“trackNum”。创建该大小的字节数组, 并将媒体头部分中的数据从该媒体数据文件复制到该数组。将该字节数组的位置的值指派给列表“MediaHeader”中的项“trackNum”。

7. 将媒体数据的剩余部分从(输入)媒体数据文件2220复制到输出mp4二进制文件2230和730。

如果需要, 在此阶段, 可对数据应用数据格式转换。例如, 可将MPEG-2音频数据(流类型=5且对象类型=64)修改为满足MPEG-4音频流的需要。这些修改依赖于MPEG-2高级音频编码(AAC)数据[ISO-IEC文档13818-7 (1997)“信息技术-移动图像和相关音频信息的通用编码-第7部分: 高级音频编码”]的详细规范。这些规范和相

8. 将量“numChunks”的值增加1。

7.2.3.3 插入odsm数据

使用过程“插入OdsM数据”处理从属于“mdat”元素2400的“odsm”元素2420。该过程将在输出mp4文件中为每一个从属于当前odsm元素2460的“odsmChunk”元素2470产生一个新的块736。

将“odsm”元素2420的“trackId”属性的值指派给量“trackId”。使用标准XML手段获得每一个从属于“odsm”元素2420和2460的“odsmChunk”元素2470。

对于每一个从属于当前“odsm”元素2460的“odsmChunk”元素2470执行下列操作：

1. 将在操作5230中确定的量“mdatId”的值指派给列表“MdatIdForChunk”中的项“NumChunks”。
2. 将量“trackId”的值指派给列表“TrackIdForChunk”中的项“NumChunks”。
3. 将输出mp4文件的当前文件位置的值指派给列表“offsetForChunk”中的项“NumChunks”。
4. 将值“-1”指派给列表“MediaDataSize”中的项“NumChunks”。
5. 将量“numChunks”的值增加1。
6. 使用标准XML手段获得每一个从属于“odsmChunks”元素2500的“odsmSample”元素2510。

对于每一个在步骤6中识别的“odsmSample”元素2510，将当前mp4文件位置指派给量“sampleStart”，将“size”属性的值指派给量“sampleSize”，并将“time”属性的值指派给量“sampleTime”。将量“sampleTime”的值指派给列表“OdsMSampleTime”中的项numOdsMSamples。将“sampleSize”的值作为所得到的二进制odsm样本的估算对待。这将由最终文件位置和“sampleStart”之间的差所确定的准确值来替代。

使用标准XML手段获得每一个从属于“osdmSample”元素2520的XML元素2530。这些从属元素被期望具有名为“ObjectDescrUpdate”2540或“ObjectDescrRemove”2570的元素。对这

些情况中的每一种按下面所指示的进行处理。

在完成对所有从属于“odsmSample”元素2520的XML元素2530的处理之后，将所得到的输出mp4文件的文件文件与量“sampleStart”的值之间的差指派给量“sampleSize”（替换从相应属性值中获得的估算）。将该值指派给列表“OdsSampleSize”中的项“numOdsSample”。然后，将量“numOdsSample”的值增加1。

7.2.3.4 ObjectDescrUpdate元素

对于每一个从属于“odsmSample”元素2520的“ObjectDescrUpdate”元素2540，图51中示出的过程用于在输出mp4文件中创建ObjectDescrUpdate对象结构2000。在操作5100，将结构标签“ObjectDescrUpdateTag”（值=1）作为8位整数写入输出mp4文件中。在操作5110，将输出mp4文件的当前文件位置指派给量“sizePos”，并将“sizePos”的值指派给量“filePos1”。在操作5120，向量“numSizeBytes”指派值“1”。在操作5130，将值0作为初始大小值2020写入输出mp4文件。

“ObjectDescrUpdate”元素2540不具有属性，因此在操作5135和5140不进行任何处理。

使用标准XML手段获得从属于“ObjectDescrUpdate”元素2540的每一个XML元素2550。这些从属元素被期望具有名为“ObjectDescriptor”2550的元素。在如下面在操作5150中所述的对每一个从属“ObjectDescriptor”元素2550进行处理之后，如图51中所指示（操作5160到5195）的那样对ObjectDescrUpdate结构2020的大小进行更新。

对于每一个从属于“ObjectDescrUpdate”元素2540的“ObjectDescrUpdate”元素2550，使用图51中示出的过程在输出mp4文件中创建ObjectDescriptor对象结构2030和2100。在操作5100，将结构标签“MP4_OD_TAG”（值=17）2108作为8位整数写入输出mp4文件。在操作5110，将输出mp4文件的当前文件位置指派给量“sizePos”，并

将“sizePos”的值指派给量“filePos2”。在操作5120，向量“numSizeBytes”指派值“1”。在操作5130，将值0作为初始大小值2116写入输出mp4文件。

将“ObjectDescriptor”元素2550的“OdId”属性的值指派给量“OdId”。将量“OdId”的数字值乘以64（左移6位），并将结果与值“31”相加，以确定量“OdId”的更改值。值“31”代表ObjectDescriptor对象结构2100内的保留字段2140。

如果指定了“ObjectDescriptor”元素2550的“url”属性，则将值“32”与更改的“OdId”值相加。将所得到的值作为16位整数写入mp4文件。然后，将表示“url”属性的值中的字符数的一个字节写入该mp4文件。然后，将“url”属性的值作为字符序列写入该mp4文件。

如果未指定“ObjectDescriptor”元素2550的“url”属性，则将更改的“OdId”值作为16位整数2124、2132和2140写入mp4文件。

然后，使用标准XML手段获得每一个从属于“ObjectDescriptor”元素2550的XML元素2560。这些从属元素被期望具有名为“EsIdRef”2560的元素。

对于每一个从属于当前“ObjectDescriptor”元素2550的“EsIdRef”元素2560的，使用图51中示出的过程在输出mp4文件中创建EsIdRef对象结构2148和2160。在操作5100，将结构标签“EsIdRefTag”（值=15）2170作为8位整数写入输出mp4文件。在操作5110，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5120，向量“numSizeBytes”指派值“1”。在操作5130，将值0作为初始大小值2180写入输出mp4文件。

在操作5135，将“EsIdRef”元素2560的“EsId”属性的值指派给量“EsId”。然后，在操作5140，将量“EsId”的数字值作为16位整数2190写入输出mp4文件。EsIdRef元素2560不具有从属元素5150。

在对“EdId”值2190处理之后，如图51中所指示的那样对EsIdRef对象结构的大小2180进行更新（操作5160到5195）。

在操作2550对“ObjectDescriptor”元素进行处理之后，将filePos2

的值指派给量“sizePos”，并如图51中所示的那样对MP4_OD对象结构2100的大小2116进行更新（操作5160到5195）。

在操作2540对“ObjectDescriptor”元素进行处理之后，将filePos1的值指派给量“sizePos”，且如图51中所示的那样对ObjectDescrUpdate对象结构2000的大小2020进行更新（操作5160到5195）。

7.2.3.5 ObjectDescrRemove元素

对于每一个从属于当前“odsmSample”元素2520的“ObjectDescrRemove”元素2570，使用图51中所示的过程在输出mp4文件中创建ObjectDescrRemove对象结构2040。在操作5100，将结构标签“ObjectDescrRemoveTag”（值=2）2050作为8位整数写入输出mp4文件。在操作5110，将输出mp4文件的当前文件位置指派给量“sizePos”，并将“sizePos”的值指派给量“filePos1”。在操作5120，向量“numSizeBytes”指派值“1”。在操作5130，将值0作为初始大小值2060写入输出mp4文件。

将“ObjectDescrRemove”元素2570的“OdId”属性的值指派给量“OdIdList”。

量“OdIdList”包括表示一个或多个以“白空间”（空格或其他非数字字符）格开的整数的字符串。将“OdIdList”内的每一个数字字符序列解释为整数，并将所得到的值作为10位objectDescriptorId值2070写入mp4文件。写入mp4文件中的连续objectDescriptorId值2070不是字节对齐的。如果由objectDescriptorId值序列2070所占用的全部位数（n位）不是8的倍数，则将nPad 0位2080写入mp4文件，其中nPad的值由nBits模型8给出。

在处理“OdIdList”量之后，如图51中所示的那样对ObjectDescrRemove对象结构2040进行更新（操作5160到5195）。

7.2.3.6 插入sdsm数据

过程“插入sdsm数据”用于对从属于“mdat”元素2400的“sdsm”元

素2410进行处理。将“sdsM”元素2410和2440的“trackId”属性的值指派给量“trackId”。可能存在可选属性“xml File”。该属性可用于指出表示mp4bifs文档2800的输入XML文件的名称。可选地，mp4bifs文档2800可从另一个过程（诸如上述用于根据XMT-A文档创建mp4file和mp4bifs文档的过程）的结果中获得。然后，使用标准XML手段获得mp4bifs文档2800的顶级元素。

如图 28A 中所示，mp4bifs 文档 2800 包括具有单个从属“bifsConfig”元素 2810 和一个或多个从属“commandFrame”元素 2820 的顶级元素。每一个“commandFrame”元素 2820 表示情景描述流（sdsM）的“样本”。在准备对 mp4bifs 文档 2800 进行解释中，通过对从属于 mp4bifs 顶级元素 2800 的“commandFrame”元素 2820 的数量进行计数，来确定 sdsM 样本的数量。将所得到的值指派给量“MaxNumSdsMSample”，并创建每一个具有 MaxNumSdsMSample 项的两个列表。这些表中的一个“SdsMSampleTime”是整数值的列表。第二个列表“SdsMSample”是浮点值优选为双精度浮点值（每项 64 位）的列表。向量“NumSample”指派值 0。

使用标准 XML 手段获得从属于“sdsM”元素 2440 的每一个“块”元素 2450。对于每一个“sdsM”元素 2440，最多期望具有一个从属“块”元素 2440。对“块”元素 2440 执行下列操作：

1. 将在操作 5230 确定的量“mdatId”的值指派给列表“MdatIdForChunk”中的项“NumChunks”。
2. 将量“trackId”的值指派给列表“TrackIdChunk”中的项“NumChunks”。
3. 将当前文件位置的值指派给列表“OffsetForChunk”中的项“NumChunks”。
4. 向列表“MediaDataSize”中的项“NumChunks”指派值“-1”。
5. 将量“numChunks”的值增加 1。

然后，按如下所述对 mp4bifs 文档 2800 进行解释。在该文档被解释时，将数据值写入输出 mp4 文件 700，并将各值输入到列表

“SdsmSampleSize”和“SdsmSampleTime”。在面向对象的实现中，这通过创建新的 SdsmEncoder 对象、并调用对于该对象的方法“encodeSdsm”来实现。该方法将返回完整列表“SdsmSampleSize”和“SdsmSampleTime”以及将表示sdsm的二进制编码的数据添加到输出mp4文件700中。

使用标准XML手段获得从属于mp4bifs顶级元素2800的“bifsConig”元素2810。将该元素的“routeIdBits”属性的值指派给量“RouteIdBits”。将该元素的“nodeIdBits”属性的值指派给量“NodeIdBits”。将数字2提高到乘方“NodeIdBits”次乘方后的值（或左移NodeIdBits位的“1”）指派给量MaxUpdateableNodes。创建两个新的整数列表“UpdateableNodeId”和“UpdateableNodeNumber”。这些列表的每一个中的项数由“MaxUpdateableNodes”的值确定。向量“NumUpdateableNodes”指派值0。向布尔量“bUseNames”指派值“假”。

然后，使用标准XML手段获得从属于mp4bifs顶级元素2800的每一个“commandFrame”元素2820。下面的手段用于对每一个这样的“commandFrame”元素2820进行处理：

1. 将输出mp4文件的当前文件位置的值指派给量“FilePointerAtStart”。

2. 将“commandFrame”元素2820的“time”属性的值指派给量“Time”。将量“Time”的值指派给列表“SdsmSampleTime”中的项“NumSample”。

3. 使用标准XML手段获得从属于“commandFrame”元素2830的每一个bifsCommand元素2840。每一个这样的从属元素处理如下。

4. 将输出mp4文件的当前文件位置的值指派给量“FilePointerAtEnd”，并将量“FilePointerAtEnd”的值与量“FilePointerAtStart”的值之间的差指派给列表“SdsmSampleSize”中的项“NumSamples”。

5. 将量“NumSamples”的值增加1。

如图28B中所示，每一个“commandFrame”元素2830包含一个或

多个从属bifsCommand元素2840。每一个bifsCommand元素2910表示11个可能以sdsm数据编码的可能BIFS命令中的一个。这些包括三个插入命令（“插入节点”、“插入路由”和“插入索引值”）、三个删除命令（“删除节点”、“删除路由”和“删除索引值”）、和五个替换命令（“替换节点”、“替换路由”、“替换索引值”、“替换字段”和“替换情景”）。如图29A中所示，一个BIFS命令元素2910可能具有表示BIFS节点2920的从属元素。如图29B中所示，表示替换情景命令的bifsCommand元素2930还包含单个从速Routes元素2950，其包含一个或多个Route元素2960。

在生成从属于特定“commandFrame”元素2830的bifsCommand元素2840的二进制表示之前，对每一个从属bifsCommand元素2910进行“扫描”，以识别对其已为“NodeId”属性3010指定了一个值的所有从属Node元素2920和3000。该“扫描”操作通过是使用标准XML手段获得每一个从属于当前“commandFrame”元素2830的bifsCommand元素2840来实现。该操作仅应用于6个可能包括一个或多个从属BIFS节点元素2920和2940的bifsCommand元素（“插入节点”、“插入索引值”、“替换节点”、“替换索引值”、“替换字段”和“替换情景”）2910。

用于实现该“扫描”操作的过程除了对输出mp4文件什么都不做之外，与下面描述的用于随后BIFS命令解释的过程那些过程相同，且除了“nodeId”属性、和具有字段数据类型“node”或“command buffer”的属性之外忽略所有属性。对于每一个为其指定了“NodeId”属性的节点，将“NodeId”属性的值指派给列表“UpdateableNodeId”中的项“numUpdateableNodes”。将该节点的“节点数”属性的值指派给“UpdateableNodeNumber”列表中的相应项。特定节点的“节点数”属性由节点名表中与该节点元素的元素名匹配的项的索引所确定。节点名表定义于MPEG-4系统规范文档中。然后，将量“NumUpdateableNode”的值增加1。

在对从属bifsCommand元素2840扫描之后，再次使用标准XML手段获得每一个从属于当前“commandFrame”元素2830的XML元素

2840。如图11B中所示，每一个BIFS命令1120之后有“continue”位1130和1140。在对从属于除了第一个命令之外的“commandFrame”元素2830的每一个bifsCommand元素2840进行处理之前，将具有值“1”的单个位写入输出mp4文件，以指定“continue = 1”1130。然后，按下面所述对每一个从属bifsCommand元素2840进行处理。在所有从属元素均已处理之后，将具有值“0”的单个位写入输出mp4文件，已指出“continue = 0”（命令帧结束）1140。如果用于以二进制形式表示该命令帧的总位数不是8的倍数，则以0填充最后字节1150，以使总位数达到8的倍数。

然后，使用下面的手段将二进制BIFS命令数据结构添加到每一个从属于当前commandFrame元素2830的bifsCommand元素2840的输出mp4文件。

如果bifsCommand元素2840表示三个插入命令中的一个，则将两位插入码（二进制值 = 00）1206写入输出mp4文件。如果BIFS命令元素2840表示三个删除命令中的一个，则将两位删除码（二进制值 = 01）1220写入输出mp4文件。如果BIFS命令元素2840表示四个替换命令（除替换情景之外）中的一个，则将两位替换码（二进制值 = 10）1240写入输出mp4文件。如果BIFS命令元素2840表示替换情景命令，则将两位情景替换码（二进制值 = 11）1280写入输出mp4文件。

7.2.3.7 节点插入BIFS命令

在“插入节点”bifsCommand元素2840和2910的情况下，将如图13A中所示的节点插入BIFS命令1300添加到输出mp4文件。将“节点”的两位参数类型码（二进制值 = 00）1308写入输出mp4文件的插入码1304之后。将插入节点元素的“parented”属性的值指派给量“nodeID”，并将该量1312的整数值写入mp4文件。用于对量“nodeID”的值进行编码的位数由量“nodeIdBits”的值指定。插入节点元素的“parentID”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeId列表中的相应项指出从属节点元素的父节点的“节

点数”属性的值。

将插入节点元素的“插入位置”属性的值指派给量“insertionPosition”，并将表示该量1316的整数值的两位写入mp4文件。如果量“insertionPosition”的值为0，则将插入节点元素的“位置”属性的值指派给量“position”，并将表示该量1320的整数值达到8位写入mp4文件。

每一个插入节点元素包含从属节点元素2920。将该节点元素1324的二进制SFNode表示添加到表示插入位置1316和1320的数据之后的输出mp4文件中。SFNode结构的格式显示于图17中。用于创建该SFNode结构的过程在下面描述。

7.2.3.8 索引值插入BIFS命令

在“插入索引值”bifsCommand元素2840和2910的情况下，将如图13B中所示的索引值插入BIFS命令1328添加到输出mp4文件。将“索引值”的两位参数类型码（二进制值 = 10）1336写入mp4文件的插入码1332之后。将插入索引值元素的“nodeId”属性的值指派给量“nodeID”，并将该量1340的整数值写入mp4文件。用于对量“nodeID”的值进行编码的位数由量“nodeIdBits”的值指定。

插入索引值元素的“nodeId”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出要由与该BIFS命令相关的字段值修改的BIFS节点的“节点号”属性的值。

该BIFS命令的“字段索引”属性的值由该节点号的字段名列表中具有匹配插入索引值元素的“inFieldName”属性的值匹配的值的项的索引确定。字段名列表定义于MPEG-4系统规范中。该字段的inFiledID属性的值由节点号的值、字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由节点号的值、inFiledID的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1344将量inFiledID的整数值写入mp4文件。

将插入索引值元素的“插入位置”属性的值指派给量

“insertionPosition”，并将表示该量1316的整数值的两位写入mp4文件1348。如果量“insertionPosition”的值为0，则将插入索引值元素的“位置”属性的值指派给量“position”，并将表示该量的整数值的16位写入mp4文件1352。

每一个插入索引值元素包括“value”属性。对该value属性的解释依赖于插入索引值元素的inFieldName属性所标识的属性字段的“字段数据类型”属性。该字段数据类型属性由节点号的值、字段索引的值、和MPEG-4系统规范中定义的一组表所确定。如果字段数据类型属性为“SFNode”，则“value”属性的值指出从属节点元素的名称。否则，“value”属性的值直接指出“字段值”的值。在每一种情况下，下面在“SFField结构”之下描述的手段用于解释该value属性、创建由value属性指出的字段值的二进制表示、并将结果添加到输出mp4文件1356中。

7.2.3.9 路由插入BIFS命令

在“插入路由”bifsCommand元素2840和2910的情况下，将如图13C中所示的路由插入BIFS命令1360添加到输出mp4文件中。将“路由”的两位参数类型码（二进制值 = 11）1368写入mp4文件的插入码1364之后。如果已对插入路由元素的“routeId”属性指定了一个值，则将具有值“0”的单个位作为“isUpdateable”值1372写入输出mp4文件。否则，将值“1”作为“isUpdateable”值1372写入输出mp4文件，其后有由该插入路由元素的“routeId”属性所指出的值1376。用于表示该整数值的位数由量RouteIdBits的值所指出。

将插入路由元素的“出发节点”属性的值指派给量“departureNodeID”，并将该值写入mp4文件1380。用于表示量“departureNodeID”的整数值的位数由量“NodeIdBits”的值指定。

插入路由元素的“出发节点”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出“出发节点”的“节点号”属性的值。

该出发节点的“字段索引”属性的值由该节点号的字段名列表中

具有匹配插入路由元素的“出发字段名”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的出发字段ID属性的值由出发节点的节点号的值、出发节点的字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由出发节点的节点号的值、出发字段ID的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1384将量出发字段ID的值写入mp4文件。

将插入路由元素的“到达节点”属性的值指派给量“arrivalNodeID”，并将该值写入mp4文件1388。将用于表示量“arrivalNodeID”的整数值的位数由量“NodeIdBits”的值所指出。

插入路由元素的“到达节点”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeId列表中的相应项指出“到达节点”的“节点号”属性的值。

到达节点的“字段索引”属性的值由该节点号的字段名列表中具有与该插入路由元素的“到达字段名”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的到达字段ID属性的值由到达节点的节点号的值、到达节点的字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由到达节点的节点号的值、到达字段ID的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1392将量到达字段ID的整数值写入mp4文件。

7.2.3.10 节点删除命令

在“删除节点”bifsCommand元素2840和2910的情况下，将如图14A中所示的节点删除BIFS命令1400添加到输出mp4文件中。在该情况下，将“节点”的两位参数类型码（二进制值 = 00）1412写入mp4文件的删除码1406之后。将“nodeId”属性的值指派给量“nodeID”，并将该值的整数表示写入mp4文件1418。用于表示量“nodeID”的整数值的位数由量“NodeIdBits”的值指出。

7.2.3.11 索引值删除BIFS命令

在“删除索引值”bifsCommand元素2840和2910的情况下，将如图14B中所示的索引值删除BIFS命令1424添加到输出mp4文件。将“索引值”的两位参数类型码（二进制值 = 10）1436写入mp4文件的删除码1430之后。将“nodeId”属性的值指派给量“nodeID”，并将该值写入mp4文件1442。用于表示量“nodeID”的整数值的位数由量“nodeIDBits”的值指定。

删除索引值元素的“nodeId”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出与该BIFS命令相关的BIFS节点的“节点号”属性的值。

该BIFS命令的“字段索引”属性的值由该节点号的字段名列表中具有匹配删除索引值元素的“inFieldName”属性的值匹配的值的项的索引确定。字段名列表定义于MPEG-4系统规范中。该字段的inFiledID属性的值由节点号的值、字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由节点号的值、inFiledID的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1448将量inFiledID的整数值写入mp4文件。

将删除索引值元素的“删除位置”属性的值指派给量“deletionPosition”，并将表示该量的整数值的两位写入mp4文件1454。如果量“deletionPosition”的值为0，则将删除索引值元素的“位置”属性的值指派给量“position”，并将表示该量的整数值的16位写入mp4文件1460。

7.2.3.12 路由删除BIFS命令

在“删除路由”bifsCommand元素2840和2910的情况下，将如图14C中所示的路由删除BIFS命令1466添加到输出mp4文件中。将“路由”的两位参数类型码（二进制值 = 11）1478写入mp4文件的删除码1472之后。将删除路由元素的“routeId”属性的值指派给量“routeID”，

并将该值的整数表示写入mp4文件1484。用于表示量“routeID”的整数值的位数由量“RouteIdBits”的值指出。

7.2.3.13 节点替换BIFS命令

在“替换节点”bifsCommand元素2840和2910的情况下，将如图15A中所示的节点替换BIFS命令1500添加到输出mp4文件中。将“节点”的两位参数类型码（二进制值 = 00）1508写入mp4文件的替换码1504之后。将替换节点元素的“nodeId”属性的值指派给量“nodeID”，并将该值的整数表示写入mp4文件1510。用于表示量“nodeID”的整数值的位数由量“NodeIdBits”的值指出。

每一个替换节点元素包含从属节点元素2920。将该节点元素1514的二进制SFNode表示添加到输出mp4文件的表示nodeID值1510的数据之后。图17中示出了SFNode结构的格式。用于创建该SFNode结构的过程在下面描述。

7.2.3.14 字段替换BIFS命令

在“替换字段”bifsCommand元素2840和2910的情况下，将如图15B中所示的字段替换BIFS命令1520添加到输出mp4文件中。将“字段”的两位参数类型码（二进制值 = 01）1528写入mp4文件的替换码1524之后。将替换字段元素的“nodeId”属性的值指派给量“nodeID”，并将该值写入mp4文件1530。用于表示量“nodeID”的整数值的位数由量“NodeIdBits”的值指出。

替换字段元素的“nodeId”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出要由与该BIFS命令相关的字段值所更改的BIFS节点的“节点号”属性的值。

该BIFS命令的“字段索引”属性的值由该节点号的字段名列表中具有与该替换字段元素的“inFieldName”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的inFieldID

属性的值由节点号的值、字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由节点号的值、inFiledID的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1534将量inFieldID的值写入mp4文件中。

每一个替换字段元素包括“value”属性。对该值属性的解释依赖于由该替换字段元素的inFieldName属性所标识的属性字段的“字段数据类型”属性。字段数据类型属性由节点号的值、字段索引、和MPEG-4系统规范中定义的一组表所确定。如果字段数据类型属性为“SFNode”，则“value”属性的值指出从属节点元素的名称。否则，“value”属性的值直接指出“字段值”的值。在每一种情况下，下面在“SFField结构”之下描述的手段用于解释value属性、创建由value属性所指出的字段值的二进制表示、并将结果添加到输出mp4文件1538中。

7.2.3.15 索引值替换BIFS命令

在“替换索引值”bifsCommand元素2840和2910的情况下，将如图15C中所示的索引值替换BIFS命令1540添加到输出mp4文件中。将“索引值”的两位参数类型码（二进制值 = 10）1548写入mp4文件的替换码1544之后。将“nodeId”属性的值指派给量“nodeID”，并将该值写入mp4文件1550。用于表示量“nodeID”的整数值的位数由量“NodeIDBits”的值指出。

替换索引值元素的“nodeId”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出要由与该BIFS命令相关的字段值所更改的BIFS节点的“节点号”属性的值。

该BIFS命令的“字段索引”属性的值由该节点号的字段名列表中具有与该替换索引值元素的“inFieldName”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的inFieldID属性的值由节点号的值、字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由节点号的值、inFiledID

的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1554将量inFieldID的值写入mp4文件中。

将插入索引值元素的“替换位置”属性的值指派给量“replacementPosition”，并将表示该量的值的两位写入mp4文件1558。如果量“replacementPosition”的值为0，则将替换索引值元素的“位置”属性的值指派给量“position”，并将表示该量的整数值的16位写入mp4文件1560。

每一个替换索引值元素包括“value”属性。对该值属性的解释依赖于由该替换索引值元素的inFieldName属性所标识的属性字段的“字段数据类型”属性。字段数据类型属性由节点号的值、字段索引、和MPEG-4系统规范中定义的一组表所确定。如果字段数据类型属性为“SFNode”，则“value”属性的值指出从属节点元素的名称。否则，“value”属性的值直接指出“字段值”的值。在每一种情况下，下面在“SFField结构”之下描述的手段用于解释value属性、创建由value属性所指出的字段值的二进制表示、并将结果添加到输出mp4文件1564中。

7.2.3.16 路由替换BIFS命令

在“替换路由”bifsCommand元素2840和2910的情况下，将如图15D中所示的路由替换BIFS命令1570添加到输出mp4文件中。将“路由”的两位参数类型码（二进制值 = 11）1578写入mp4文件的替换码1574之后。将该替换路由元素的“routeId”属性的值指派给量“routeID”，并将该值的整数表示写入mp4文件1580。用于表示量“routeID”的整数值的位数由量“RouteIdBits”的值指出。

将替换路由元素的“出发节点”属性的值指派给量“departureNodeID”，并将该值写入mp4文件1584。用于表示量“departureNodeID”的整数值的位数由量“NodeIdBits”的值指定。

该替换路由元素的“出发节点”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出“出发节点”的“节点号”属性的值。

该出发节点的“字段索引”属性的值由该节点号的字段名列表中具有匹配该替换路由元素的“出发字段名”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的出发字段ID属性的值由出发节点的节点号的值、出发节点的字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由出发节点的节点号的值、出发字段ID的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1588将量出发字段ID的值写入mp4文件。

将该替换路由元素的“到达节点”属性的值指派给量“arrivalNodeID”，并将该值写入mp4文件1590。将用于表示量“arrivalNodeID”的整数值的位数由量“NodeIdBits”的值所指出。

该插入路由元素的“到达节点”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeId列表中的相应项指出“到达节点”的“节点号”属性的值。

到达节点的“字段索引”属性的值由该节点号的字段名列表中具有与该替换路由元素的“到达字段名”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的到达字段ID属性的值由到达节点的节点号的值、到达节点的字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由到达节点的节点号的值、到达字段ID的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1594将量到达字段ID的整数值写入mp4文件。

7.2.3.17 情景替换BIFS命令

在“替换情景”bifsCommand元素2930的情况下，将情景替换BIFS命令1290添加到输出mp4文件。如图12D中所示，情景替换BIFS命令1290包括两位情景替换码（二进制值=11）1280，其后有BIFSScene数据结构1290和1600。图16中示出了BIFSScene结构1600的成分。

在写入两位情景替换码1280之后，将6位0值（“保留”）1610写入

输出mp4文件。

替换情景元素的“USENAMES”属性的值用于确定布尔量bUseNames的值。如果“USENAMES”属性的值为“真”，则向“bUseNames”指派值“真”，并将单个“1”位写入mp4文件1620。否则，向bUsenames指派值“假”，并将单个“0”位写入mp4文件1620。接着，将单个“0”位写入mp4文件，以指示在该mp4文件1630中不会存在“protoList”。

ProtoList位1630之后有“SFNode”结构。这除了只允许在MPEG-4系统规范中定义的节点的子集的成员之外，与下面描述的“SFNode”结构相同。SFTopNode结构的描述由从属于替换情景命令元素2930的mp4bifs节点元素2940所指出。

除了所需的从属节点元素之外，mp4bifs替换情景元素2930还具有单个从属“Routes”元素2950。如果替换情景元素2930不具有从属“Routes”元素2950，则将单个“0”位作为“hasRoutes”位1650写入mp4文件，从而指出BIFS情景替换命令1270的结束。如果替换情景命令元素2930具有从属“Routes”元素2950，则将单个“1”位作为“hasRoutes”位1650写入mp4文件，其后有图18中描述的Routes结构1660。

Routes结构1660可具有两种形式中的一种，图18A中示出的列表形式1800或图18B中示出的向量形式1830。这些由第一位的值来区别，该第一位对于列表形式为“1”1805，对于向量形式为“0”1835。在本发明的一个实施例中，常常选取列表形式。因此，如果“hasRoutes”位被设为“1”，则下一位1805也被设为“1”。列表形式对向量形式的选取是不重要的，本发明可同样应用Routes结构的向量形式1830。

Mp4bifs“Routes”元素2950可具有一个或多个从属“Route”元素2960。对于每一个从属于“Routes”元素2950的“Route”元素2960，将单个“1”位1805和1815写入输出mp4文件，其后有Route元素2960的二进制描述1810和1860。将单个“0”位1820写入输出mp4文件的最后一个从属于“Routes”元素2950的“Route”元素2960的描述之后。在第一个从属“Route”元素2960的二进制描述1810之前的“1”位1805指出二进制

Routes数据结构2960的“列表形式”1800正使用。随后的“1”位1815指出“moreRoutes = 真”。最后的“0”位1820指出“moreRoutes = 假”并指示Routes结构1800和2960的结束。

每一个Route元素的二进制描述1860的结构示于图18C中。用于创建每一个Route元素的二进制描述1860的手段在下面“Route结构”之下描述。

7.2.3.18 Route结构

对于每一个从属于Route元素2950（从属于替换情景元素2930）的Route元素2960，将二进制Route结构1860添加到输出mp4文件。如果已对Route元素2960的“routeId”属性指定了一个值，则将具有值“0”的单个位作为“isUpdateable”值1865写入输出mp4文件。否则，将值“1”作为“isUpdateable”值1865写入输出mp4文件，其后有Route元素2960的“routeId”属性所指出的值1870。用于表示routeId属性的整数值的位数由量RouteIdBits的指出。

如果已对Route元素2960的“routeId”属性指定了一个值，则相应的替换情景元素2930的USENAMES属性的值为“真”，然后将Route元素2960的“name”属性的值作为已空结尾的字符串1875添加到输出mp4文件。Route元素2960的“name”属性的值是相应XMT-A Route元素390的DEF属性的副本。

将Route元素的“toNode”属性的值指派给量“outNodeID”，并将该值写入mp4文件1880。用于表示量“outNodeID”的整数值的位数由量“NodeIdBits”的值指出。

Route元素的“outNodeID”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出“出发节点”的“节点号”属性的值。

该出发节点的“字段索引”属性的值由该相应节点号的字段名列表中具有与该Route元素的“toFieldName”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的

outFieldRef属性的值由出发节点的节点号的值、出发节点的字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由出发节点的节点号的值、**outFiledRef**属性的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1885将量**outFieldRef**的值写入mp4文件中。

将Route元素2960的“fromNode”属性的值指派给量“inNodeID”，并将该值写入mp4文件1890。用于表示量“inNodeID”的整数值由量“NodeIdBits”的值指出。

Route元素2960的“inNodeID”属性的值必须与UpdateableNodeId列表中的某一项匹配。UpdateableNodeNumber列表中的相应项指出“到达节点”的“节点号”属性的值。

该到达节点的“字段索引”属性的值由该节点号的字段名列表中具有与该Route元素的“fromFieldName”属性的值匹配的值的项的索引所确定。字段名列表定义于MPEG-4系统规范中。该字段的**inFieldRef**属性的值由到达节点的节点号的值、到达节点的字段索引的值、和MPEG-4系统规范中定义的一组表所确定。量“numBits”的值由到达节点的节点号的值、**inFiledRef**属性的值、和MPEG-4系统规范中定义的表所确定。然后，使用numBits位1895将量**inFieldRef**的整数值写入mp4文件中。

7.2.3.19 SFNode结构

mp4bifs节点元素3000、3040、和3080可呈现为从属于插入节点元素、替换节点元素、替换情景元素或另一个mp4bifs节点元素3000的元素。每一个mp4bifs节点元素3000、3040和3080具有图30A、30B或30C中示出的结构。存在100多种类型的mp4bifs节点元素。这些中的每一个对应于MPEG-4系统规范中定义的一个BIFS节点。每一个BIFS节点具有指定的节点名（字符串）和属性字段的有序集合。每一个属性字段具有指定名称（字符串）和其他特性。每一类型的BIFS节点由同名mp4bifs节点元素表示，BIFS节点的每一个属性字段由相应

mp4bifs节点元素的同名属性表示。

对于大多数BIFS节点属性字段数据类型(包括布尔、整数、浮点、颜色和字符串),相关数据值由mp4bifs节点元素的相应属性的值表示。存在两个例外:字段数据类型“node”和“buffer”。在字段数据类型“node”和“buffer”的情况下,相关数据值由从属mp4bifs元素3030和3070表示,且相应属性包含相关从属元素的名称有序列表。这些名称有序列表可在不只一个属性具有node或buffer字段数据类型的情况下用于确定与mp4bifs节点元素的每一个属性相关的特定从属元素。

除了对于每一类型的BIFS节点的属性字段之外,每一个BIFS节点具有通用属性的集合。这些包括重用状态、可更新状态和掩蔽访问状态。图7中示出了所得到的属性字段和通用属性的组合。

下面的手段用于创建由mp4bifs节点元素表示的二进制SFNode结构。

第一步是检查 mp4bifs 节点元素的可选“nodeRef”属性。如果已对该属性指定了一个值,则将该属性的值指派给量“nodeIDRef”。在此情况下,将该节点分类为“重用”节点,且所得到的 BIFS 节点具有图 17A 中示出的结构。在此情况下,将具有值“1”1704 的单个位写入 mp4 文件。该位指出条件“isReused = 真”。该位之后有量“nodeIDref”1708 的值。用于表示量“nodeIDref”的值的位数由量“NodeIDBits”的值给出。在此情况下,不允许从属元素或其他属性。

如果已对 mp4bifs 节点元素的“nodeRef”属性指定了一个值,则将单个“0”位 1712 和 1730 写入 mp4 文件。该位指出条件“isReused = 假”。所得到的 BIFS SFNode 具有图 17B(掩蔽节点) 1710 或图 17C(列表节点) 1730 中示出的结构。本发明的当前实施例通常选取掩蔽节点形式 1710。选择掩蔽节点而不是列表节点是不重要的,本发明可同样使用 SFNode 的列表节点形式 1730 来实现。

接着,在操作 3000,将 mp4bifs 节点元素的元素名(NodeName)与 MPEG-4 系统规范中定义的 BIFS 节点名表中的项进行比较,以确定相应 BIFS 节点的“节点号”的值。该“节点号”的值、BIFS 节点的“节

点数据类型”、MPEG-4 系统规范中定义的一组表用于确定该 BIFS 节点的“本地节点类型”的值和用于标识该值的位数。然后，使用指定位数 1714 将该“本地节点类型”写入 mp4 文件。

如果一个 mp4bifs 节点元素从属于替换情景 bifsCommand 元素，则将相应 BIFS 节点的“节点数据类型”定义为“SFWorldNode”。否则，BIFS 节点的“节点数据类型”由“父节点”的节点号、父节点的相关属性字段的字段索引、和 MPEG-4 系统规范中定义的一组表所确定。如果一个 mp4bifs 节点元素从属于另一个 mp4bifs 节点元素，则“父节点”由其从属于的 mp4bifs 节点元素所确定。如果一个 mp4bifs 节点元素从属于一个 mp4bifs bifsCommand 元素，则“父节点”由其从属于的 mp4bifs bifsCommand 元素的“NodeId”或“parentId”属性所确定。

接着，在操作 3010，检查该 mp4bifs 节点元素的可选“NodeId”属性的存在。如果已对该节点元素的“NodeId”属性指定了一个值，则将单个“0”位写入 mp4 文件，以指示条件“isUpdateable = 假”1716。否则（即已对该节点元素的 NodeId 属性指定了一个值），则在操作 3010 将 NodeId 属性的值指派给两“nodeID”，将该节点分类为“可更新”节点，并将单个“1”位写入 mp4 文件 1716。这之后有两“nodeID”1718 的值。用于表示量“nodeID”的整数值的位数由量“NodeIdBits”的值指出。如果布尔量“bUseNames”的值为“真”，则在操作 3016，将该节点元素的“name”属性的值逐个字符地复制到 mp4 文件，其后有空字节（8 位 0）1720。

接着，将单个“1”位写入 mp4 文件，以指示已选择的 BIFS 节点 1722 的“掩蔽节点”形式。这之后有掩蔽位序列 1726 和可能的属性字段值 1728。每一个掩蔽位 1726 对应于由该 mp4bifs 节点元素的元素名指定的 BIFS 节点的类型“无掩蔽”属性字段。该无掩蔽属性字段由 MPEG-4 系统规范中的表所定义。指定 BIFS 节点的属性字段的有序集合的每一个成员被认为处于该序列中。对于每一个对应于无掩蔽属性字段中的一个、且在 mp4bifs 节点元素中已为其指定了一个属性值的属性字段，将具有值“1”的掩蔽位写入 mp4 文件。其后有相关属

性值的二进制表示。用于表示每一个这样的属性值的手段在下面描述。对于每一个对应于无掩蔽属性字段中的一个、且在 **mp4bifs** 节点元素中已为其指定了一个值的属性字段，将具有值“0”的掩蔽位写入 **mp4** 文件。

除了指定属性字段名和指定属性字段数据类型之外，每一类型的 **BIFS** 节点的每一个属性字段具有指定特性，其确定该属性字段是包括指定数据类型的单个值 (**SFField** 属性) 还是指定数据类型的多个值 (**MFField** 属性字段)。每一个 **MFField** 属性字段包括 0 或多个 **SFField** 结构，如图 17D (1760) 和图 17E (1780) 中所示。在本发明的当前实施例中，通常选择图 17D (1760) 中所示的列表形式。对于列表形式还是向量形式的选取是不重要的，且本发明可同样使用 **MFField** 结构 1780 的向量形式来实现。

7.2.3.20 SFField 结构

每一个 **SFField** 数据结构表示指定数据类型。所支持的数据类型包括“简单数据类型”，诸如布尔型、整型、浮点型、字符串型、和颜色，及“复杂数据类型”。复杂数据类型包括“**node**”和“**buffer**”数据类型。每一个具有简单数据类型的属性字段的二进制表示由 **mp4bifs** 节点元素的同名属性的值所确定。例如，“布尔型”属性字段由单个位表示。“整型”属性字段由 32 位整数值表示等。用于表示每一类型的属性字段的位数定义于 **MPEG-4** 系统规范中。

在“**node**”类型的属性字段的情况下，该属性字段的值由从属 **mp4bifs** 节点元素表示。与该属性字段相关的属性的值包括从属 **mp4bifs** 节点元素的元素名列表。通过递归应用上面对于 **mp4bifs** 节点元素 (**SFNode** 结构) 所描述的过程来创建每一个这样的从属节点元素的二进制表示。

在“**buffer**”类型的属性字段的情况下，从属元素包括 **mp4bifs bifsCommand** 元素。与该属性字段相关的属性的值包括从属 **mp4bifs bifsCommand** 元素的元素名列表。递归应用上面对于 **mp4bifs**

commandFrame 元素所描述的相同过程创建每一个这样的从属 **bifsCommand** 元素的二进制表示。

7.2.4 处理“moov”元素

创建输出 mp4 文件 2230 中的第四步骤包括对包含在 mp4file 文档 2300 中的单个 moov 元素 2320 进行处理。

使用标准 XML 手段获得从属于图 23A 中示出的 mp4file 文档 2300 的顶级元素的单个“moov”元素 2320。图 50 中示出的过程用于在输出 mp4 文件中创建具有原子 ID“moov”712 和 754 的原子。在操作 5000，将输出 mp4 文件的当前文件位置指派给量“sizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值 760。在操作 5020，将原子 ID“moov”766 写入输出 mp4 文件。将“sizePos”的值指派给量“moovSizePos”。

为 mp4bifs“moov”元素定义下面的属性：版本、标志、创建时间、修改时间、时窗、持续时间、和 nextTrackID。将为这些属性中的每一个指定的值指派给同名量（“属性值”）。将这些属性值中的每一个直到创建了“mvhd”原子 772 时才写入输出 mp4 文件。

图 50 中示出的过程用于在输出 mp4 文件中创建具有原子 ID“mvhd”的原子 772。在操作 5000，将输出 mp4 文件的当前文件指派给量“sizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值。在操作 5020，将原子 ID“mvhd”写入输出 mp4 文件。将“sizePos”的值指派给量“mvhdSizePos”。

在操作 5040，将从“moov”元素的属性获得的下列属性值写入输出 mp4 文件：

1. 版本（8位整数）
2. 标志（24位整数）
3. 创建时间（32位整数）
4. 修改时间（32位整数）
5. 时窗（32位整数）

6. 持续时间 (32位整数)
7. 保留 (76字节)
8. NextTrackID (32位整数)

“保留”数据值是除了具有值“1”的字节1、4、17和33和具有值“4”的字节48之外的全0。

Mvhd原子772不具有从属原子5050。

在写入了mvhd原子772的属性字段之后，将mp4文件中的mvhd原子772的原子大小的值更新为图50中所指示的那样（操作5060到5095）。

在完成了mvhd原子772之后，将量“trackNum”的值设为0。然后，使用标准XML手段获得从属于“moov”元素2320的每一个mp4file元素。如图23中所示，这些从属元素可包括单个“mp4fiods”元素2330、可选“udta”（用户数据）元素2340、和一个或多个“trak”元素2350。对这些从属元素中的每一个按下面操作5050所述的那样进行处理。

在完成了对所有从属于“moov”元素2320的mp4file元素的处理之后，将量“moovSizePos”的值指派给“sizePos”，并将moov原子754的原子大小760的值更新为图50中所示的那样（操作5060到5095）。

7.2.4.1 处理mp4fiods元素

使用标准 XML 手段获得从属于图 23A 中示出的“moov”元素 2320 的“mp4fiods”元素 2330。图 50 中示出的过程用于在输出 mp4 文件中创建具有原子 ID“iods”778 和 800 的原子。在操作 5000，将输出 mp4 文件的当前文件位置指派给量“sizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值 804。在操作 5020，将原子 ID“iods”808 写入输出 mp4 文件。将“sizePos”的值指派给量“iodsSizePos”。

为“mp4iods”元素定义下面的属性：版本、objectDescriptorID 2370、url、includeInlineProfilesFlag、ODProfileLevelIndication、sceneProfileLevelIndication 、 audioProfileLevelIndication 、

visualProfileLevelIndication、**graphicsProfileLevelIndication**。将为这些属性中的每一个指定的值指派给同名量（属性值）。

将量“**version**”的值作为32位整数812和816写入mp4文件。该值表示iods原子的“版本”812和“标志”816。

图 51 中示出的过程用于在输出 mp4 文件中创建 **Mp4fInitObjectDescr** 对象结构 824。在操作 5100，将结构标签“**MP4_IOD_Tag**”（值 = 16）828作为8位整数写入输出mp4文件。在操作5110，将输出mp4文件的当前文件指派给量“**sizePos**”，并将“**sizePos**”的值指派给量“**mp4fiodPos**”。在操作 5120，将值“1”指派给量“**numSizeBytes**”。在操作5130，将值零作为初始大小值832写入输出mp4文件。

将量“**objectDescriptorID**”的值作为10位整数836写入mp4文件。如果已对“**url**”属性指定了一个值，则将单个“1”写入mp4文件840。否则，将单个“0”位写入mp4文件840。如果量“**includeInlineProfilesFlag**”的值为真，则将单个“1”位写入mp4文件844。否则，将单个“0”位写入mp4文件844。将值“15”作为4位整数（二进制值 = 1111）848写入mp4文件。

如果已对“**url**”属性指定了一个值，则将量“**url**”的值作为“Pascal 串”（用于其后有特定数量的字符字节的字符数的一个字节）写入mp4文件。否则，在操作 5140，将量 **ODProfileLevelIndication** 852、**sceneProfileLevelIndication** 856、**audioProfileLevelIndication** 960、**visualProfileLevelIndication** 864、和 **graphicsProfileLevelIndication** 868的值其每一个作为8位整数写入mp4文件。

然后，在操作5050，使用标准XML手段获得从属于“**mp4fiods**”元素的每一个元素。如图23B中所示，“**mp4fiods**”元素2360被期望具有一个或多个从属“**EsIdInc**”元素2390，且每一个“**EsIdInc**”元素2390具有“**trackID**”属性。

对于每一个从属于“**mp4fiods**”元素2360的“**EsIdInc**”元素2390，将包括下列属性的 **ES_ID_Inc** 对象结构876添加到输出mp4文件：

1. 表示“ES_ID_IncTag”的值（值 = 14）的一个字节880，
2. 表示值“4”（“numBytes”）的一个字节884，和
3. 表示作为32位整数（“ES_ID”）的“trackID”属性的值四个字节888。

在操作5050完成对所有从属于“mp4fiods”元素2360的“EsIdInc”元素2390的处理之后，将MP4_IOD大小832的值更新为如图51中所示的那样（操作5160到5195）。然后，将量“iodsSizePos”的值指派给量“sizePos”，并将iods原子800的原子大小804更新为如图50中所示（操作5060到5095）。

7.2.4.2 处理每一个trak元素

使用标准 XML 手段获得从属于图 23A 中示出的“moov”元素2320的每一个“trak”元素2350。图50中示出的过程用于在输出mp4文件中创建具有原子ID“trak”790和900的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值903。在操作5020，将原子ID“trak”906写入输出mp4文件。将“sizePos”的值指派给量“trakSizePos”。

为“trak”元素定义下面的属性：版本、标志、创建时间、修改时间、trackID和持续时间。将为这些属性中的每一个指定的值指派给同名量（属性值）。将这些属性值中的每一个直到创建了“tkhd”原子910时才写入输出mp4文件。

图50中示出的过程用于在输出mp4文件中创建具有原子ID“tkhd”910的原子。在操作5000，将输出mp4文件的当前文件指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“tkhd”写入输出mp4文件。将“sizePos”的值指派给量“tkhdSizePos”。

在操作5040，将下列量的值写入mp4文件：

1. 版本（8位整数）

2. 标志 (24位整数)
3. 创建时间 (32位整数)
4. 修改时间 (32位整数)
5. TrackID (32位整数)
6. 保留1 (32位, 0), 然后将文件位置保存为“durationPos”,
7. 持续时间 (32位整数)
8. 保留2 (56字节),
9. 保留3 (32位, 值 = 0x01400000)
10. 保留4 (32位, 值 = 0x00f00000)。

“保留2”数据值是除了具有值“1”的字节17和33及具有值“4”的字节48之外的全0。

tkhd原子910不具有从属原子。在操作5040写入了tkhd原子910的属性字段之后, 将mp4二进制文件中的tkhd原子910的原子大小的值更新为图50中所指示的那样 (操作5060到5195)。

在完成了tkhd原子910之后, 使用标准XML手段获得从属于“trak”元素2600的每一个mp4file元素。如图26中所示, 这些从属元素可包括单个“mdia”元素2604、可能的“tref”(trak引用)元素2636、和可能的“edts”元素2644。对这些从属元素中的每一个按下面操作5050所述的那样进行处理。

在完成了对所有从属于“trak”元素2600的mp4file元素的处理之后, 将量“trakSizePos”的值指派给“sizePos”, 并将trak原子900的原子大小903的值更新为图50中所示的那样 (操作5060到5095)。将量“trackNum”的值增加1。

7.2.4.3 处理mdia元素

使用标准XML手段获得从属于图26A中示出的“trak”元素2600的单个“mdia”元素2604。图50中示出的过程用于在输出mp4文件中创建具有原子ID“mdia”912的原子。在操作5000, 将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010, 将值零写入输出

mp4 文件，替换原子大小值。在操作 5020，将原子 ID“mdia”写入输出 mp4 文件。将“sizePos”的值指派给量“mdiaSizePos”。

为“mdia”元素定义下面的属性：版本、标志、创建时间、修改时间、时窗、持续时间、语言和质量。将为这些属性中的每一个指定的值指派给同名量（属性值）。将这些属性值中的每一个直到创建了“mdhd”原子915时才写入输出mp4文件。

图 50 中示出的过程用于在输出 mp4 文件中创建具有原子 ID“mdhd”915的原子。在操作5000，将输出mp4文件的当前文件指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5040，将下列量的值写入mp4文件：

1. 版本（8位整数）
2. 标志（24位整数）
3. 创建时间（32位整数）
4. 修改时间（32位整数）
5. 时窗（32位整数）
6. 持续时间（32位整数）
7. 语言（16位整数）
8. 质量（16位整数）

mdhd原子915不具有从属原子。在操作5040写入了mdhd原子915的属性字段之后，将mp4文件中的mdhd原子915的原子大小的值更新为图50中所指示的那样（操作5060到5095）。

然后，检查媒体数据2220，以确定每一个“样本”（也称作“访问单元”）的大小和持续时间。将在列表StreamTypeForTrack中的量trackNum的值所指定的项的值指派给量streamType。将在列表ObjectTypeForTrack中的量trackNum的值所指定的项的值指派给量objectType。

如果量streamType的值为“4”，则创建三个列表，称作sampleSize、sampleTime和syncSample。每一个列表中的项数由列表MediaSample中的量trackNum的值所指定的项的值所确定。然后，以

依赖于媒体的方式检查媒体数据文件中的数据，以确定每一个样本的大小和持续时间。将结果指派给列表sampleSize和sampleTime中的项。将确定为“随机访问样本”的每一个样本的索引指派给列表“syncSample”中的项。

如果量streamType的值为“5”，则创建两个列表，称为sampleSize和sampleTime。每一个列表中的项数由列表MediaSample中量trackNum的值所指定的项的值所确定。然后，以依赖于媒体的方式检查媒体数据文件中的数据，以确定每一个样本的大小和持续时间。将结果指派给列表sampleSize和sampleTime中的项。

在其他情况（流类型不为“4”或“5”）下，整个媒体数据流定义为一个样本。

然后，使用标准XML手段获得从属于“mdia”元素2604的每一个元素。如图26A中所示，这些从属元素可包括单个“hdlr”（处理器）元素2608、单个“minf”（媒体信息）元素2612、单个“sthl”（样本表）元素2628、和媒体头（“*mhd”）元素2632。对这些从属元素中的每一个按下面操作5050所述的那样进行处理。

在完成了对所有从属于“mdia”元素2604的mp4file元素的处理之后，将量“mdiaSizePos”的值指派给“sizePos”，并将mdia原子912的原子大小的值更新为图50中所示的那样（操作5060到5095）。

7.2.4.4 处理hdlr元素

使用标准XML手段获得从属于图26A中所示的“media”元素2604的单个“hdlr”元素2608。使用图50中所示的过程在输出mp4文件中创建具有原子ID“hdlr”918的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件，替代原子大小值。在操作5020，将原子ID“hdlr”写入输出mp4文件。

为“hdlr”元素2608定义下列属性：版本、标志、处理器类型和名称。将这些属性中的每一个指定的值指派给同名量。接着在操作5040，将下列量的值写入mp4文件：

- 1、版本（8 位整数），
- 2、标志（24 位整数），
- 3、处理器类型（32 位整数），
- 4、保留（12位，都为0）
- 5、名称（以空结尾的字符串）

hdlr元素2608没有从属元素5050。在已写入hdlr原子918的属性字段之后，将hdlr原子918的原子大小值更新为图50中显示的那样（操作5060至5095）。

7.2.4.5 处理minf元素

使用标准XML手段获得从属于“media”元素2604的单个“minf”元素2612，如图26A中所示。使用图50中所示的过程在输出mp4文件中创建具有原子ID“minf”918的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件，替代原子大小值。在操作5020，将原子ID“minf”写入输出mp4文件。将“sizePos”的值指派给量“minfSizePos”。

“minf”元素2612不具有属性和没有用作minf原子920的属性字段（见操作5030至5040）。

接着，在操作5050，使用标准XML手段获得从属于“minf”元素2612的所有元素。如图26A中所示，这些从属元素可包括“dinf”元素2616、“stbl”元素2628和媒体头（*mhd）元素2632。用于处理“stbl”元素2628和媒体头（*mhd）元素2632的手段在下面描述。

在“dinf”元素2628的情况下，使用图50所示的过程在输出mp4文件中创建具有原子ID“dinf”924的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件，替代原子大小值。在操作5020，将原子ID“dinf”写入输出mp4文件。将“sizePos”值指派给量“dinfSizePos”。

“dinf”元素2616不具有属性，且没有用作dinf原子924的属性字段（见操作5030）。

接着，在操作5040，使用标准XML手段获得从属于“dinf”元素2616的所有元素。如图26A中所示，这些从属元素可包括“dref”元素2620。

如果“dinf”元素2616具有从属“dref”元素2620，则使用图50中所示的过程在输出mp4文件中创建具有原子ID“dref”927的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010将值0写入输出mp4文件替代原子大小值。在操作5020将原子ID“dref”写入输出mp4文件。将“sizePos”值指派给量“drefSizePos”。“dref”元素不具有属性，且dref原子不具有属性字段。

接着，使用标准XML手段获得从属于“dref”元素2620的所有元素。如图26A中所示，这些从属元素可包括“urlData”元素2624。

如果“dref”元素2620具有从属“urlData”元素2624，则使用图50中所示的过程在输出mp4文件中创建具有原子ID“url”930的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件，替代原子大小值。在操作5020，将四字符原子ID“url”（u-r-l-space）写入输出mp4文件。

在操作5030，将“urlData”元素的“版本”、“标志”和“位置”属性的值指派给同名量：

在操作5040，将下列值写入输出mp4文件：

- 1、表示量“version”的值的8位整数；
- 2、表示量“flags”的值的24位整数；
- 3、表示量“location”的值的以空结尾的字符串。

“urlData”元素2624没有从属元素（见操作5050）。在将这些属性值写入输出mp4文件后，将“url”原子930的原子大小值更新为图50中显示的（操作5060至5095）。

接着将量“drefSizePos”的值指派给“sizePos”且将dref原子927的原子大小值更新为图50中显示的（操作5060至5095）。

接着将量“dinfSizePos”的值指派给“sizePos”且将dinf原子924的原子大小值更新为图50显示的那样（操作5060至5095）。

在完成对从属于“minf”元素2612的所有mp4文件元素的处理后，将量“minfSizePos”的值指派给量“sizePos”，且将minf原子920的原子大小值更新为图50显示的那样（操作5060至5095）。

7.2.4.6 处理stbl元素

使用标准XML手段获得从属于如图26A中所示的“minf”元素2612的单个“stbl”元素2628。使用图50中所示的过程在输出mp4文件中创建具有原子ID“stbl”933、950的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件，替代原子大小954的值。在操作5020，将原子ID“stbl”957写入输出mp4文件。将“sizePos”值指派给量“stblSizePos”。

“stbl”元素2628不具有属性，且stbl原子不具有属性字段（见操作5030至5040）。

接着，使用标准XML手段获得从属于“stbl”元素2652的所有元素。如图26B中所示，这些从属元素可包括“stsc”元素2656、“stts”元素2660、“stco”元素2664、“sts在”元素2668、“stss”元素2672和“stsc”元素2676。这些元素的每一个除了对于其单个实例是可选的“stss”元素2672以外的都是所需的。以下在操作5050描述用于处理这些元素中的每一个的手段。

在完成对从属于“stbl”元素2652的所有mp4文件元素的处理后，将量“stblSizePos”的值指派给“sizePos”，且将stbl原子950的原子大小954的值更新为图50显示的那样（操作5060至5095）。

7.2.4.7 处理stsc元素

使用标准XML手段获得从属于如图26B中所示的“stbl”元素2652的单个“stcs”元素2656。使用图50中所示的过程在输出mp4文件中创建具有原子ID“stcs”960的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件，替代原子大小值。在操作5020，将原子ID“stcs”写入输出mp4文件。

“stcs”元素2656具有“版本”和“标志”属性。在操作5030，将这些属性中的每一个值指派给同名量。接着在操作5040，将量“version”的值作为8位整数写入mp4文件及将量“version”的值作为24位整数写入mp4文件。

将mp4文件的文件位置指派给量“numEntriesPos”，将32位0值写入mp4文件，且将0值指派给量“numEntries”。

接着在操作5050，使用标准XML手段获得从属于“stcs”元素2656的所有元素。这些从属元素将由一个或多个“样本-块”元素组成。每个“样本-块”元素具有属性“firstChunk”、“numsample”和“sampleDesc”。

对于从属于“stsc”元素2656的每个“样本-块”元素执行下列操作：

- (1) 将“firstChunk”属性的值作为32位整数写入mp4文件。
- (2) 将“numsample”属性的值作为32位整数写入mp4文件。
- (3) 将“sampleDesc”属性的值作为32位整数写入mp4文件。
- (4) 量“numEntries”的值增加1。

在完成对从属于“stcs”元素2656的所有元素的处理后，在操作5060，将mp4文件的文件位置指派给量“endPos”。将mp4文件的文件位置改变为由量“numEntriesPos”指定的值，且将量“numEntries”的值作为32位整数写入mp4文件。接着将stsc原子960的原子大小值更新为图50中显示的那样（操作5070至5095）。

7.2.4.8 处理stsc元素

使用标准XML手段获得从属于如图26B中所示的“stbl”元素2652的单个“stts”元素2660。使用图50中所示的过程在输出mp4文件中创建具有原子ID“stts”963的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件，替代原子大小值。在操作5020，将原子ID“stts”写入输出mp4文件。

“stts”元素2656具有“版本”和“标志”属性。在操作5030，将这些属性中的每一个值指派给同名量。接着在操作5040，将量“version”的值作为8位整数写入mp4文件及将量“version”的值作为24位整数写入

mp4文件。

将mp4文件的文件位置指派给量“numEntriesPos”，将32位0值写入mp4文件，且将0值指派给量“numEntries”。

将列表StreamTypeForTrack中项“trackNum”的值指派给量“streamType”。将列表ObjectTypeForTrack中项“trackNum”的值指派给量“objectType”。量“trackNum”的值作为“处理trak元素”部分过程被确定。

如果量“streamType”的值是“1”，将在odsm数据被输入到odsm的mdat原子中时创建的odsm样本时间值（OdsampleTime）表用于确定每一个odsm样本的持续时间值。每个odsm样本的持续时间由该列表中连续项值之间的差所确定。这些值在跟踪时间单元中指定。

如果量“streamType”的值是“3”，将在sdsm数据被输入到sdsm的mdat原子中时创建的sdsm样本时间值（SdsampleTime）表用于确定每一个sdsm样本的持续时间值。每个sdsm样本的持续时间由该列表中连续项值之间的差所确定。结果持续时间值在短时间内乘以包含该“stts”元素的“track”元素的“timescale”属性值以在跟踪时间单元中确定持续时间值。

如果量“streamType”的值是“4”或量“streamType”的值是“5”且量“objectType”的值是“64”或“107”，将在相应媒体数据被输入到mdat原子中时创建的媒体样本时间值列表（SampleTime）用于确定每一个媒体样本的持续时间值。每个媒体样本的持续时间由该列表中相应项而确定。这些值在根据时间单元中指定。

前述三种中的每种情况，将具有相同持续时间的连续样本数目指派到量“numSample”。每个时刻样本的持续时间值不同于前述样本持续时间值，将“numSample”的值作为32位整数写入mp4文件，前述样本持续时间值作为32位整数写入mp4文件，将值1指派给“numSample”值，且量“numEntries”的值增加1。

另外，使用标准XML手段获得从属于如图26B中所示的“stts”元素2660的所有元素。这些从属元素将包括一个或多个“timeToSample”

元素。每个“timeToSample”元素具有属性“numSample”和“duration”。为从属于“stts”元素2660的每个“timeToSample”元素重复下列操作：

- (1) 将“numSample”属性的值作为32位整数写入mp4文件。
- (2) 将“duration”属性的值作为32位整数写入mp4文件。
- (3) 将量“numEntries”的值增加1。

在完成对从属于“stts”元素2660所有元素的处理后，在操作5060将mp4文件的文件位置指派给量“endPos”。将mp4文件的文件位置改变为由量“numEntriesPos”指定的值，且将量“numEntries”的值作为32位整数写入mp4文件。接着将stts原子963的原子大小值更新为图50中显示的那样（操作5070至5095）。

7.2.4.9 处理stco元素

使用标准XML手段获得从属于如图26B中所示的“stbl”元素2652的单个“stco”元素2664。使用图50中所示的过程在输出mp4文件中创建具有原子ID“stco”966的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值0写入输出mp4文件替代原子大小值。在操作5020将原子ID“stco”写入输出mp4文件。

“stco”元素2664具有“版本”和“标志”属性。在操作5030，将这些属性中的每一个值指派给同名量。接着在操作5040，将量“version”的值作为8位整数写入mp4文件及将量“version”的值作为24位整数写入mp4文件。

将mp4文件的文件位置指派给量“numEntriesPos”，将32位0值写入mp4文件，且将0值指派给量“numEntries”。

在操作5050，使用标准XML手段获得从属于“stco”元素2664的所有元素。这些从属元素将由一个或多个“chunkOffset”元素。将下列两个属性为“chunkOffset”元素：“mdatId”和“mdatOffset”。

为从属于“stco”元素2664的每个“chunkOffset”元素重复下列操作：

(1) 将定义为“mdatId”和“mdatOffset”属性的值指派给同名量。

(2) 通过下列三个条件确定量“chunk”的值:

- a、列表mdatIdForChunk中相应项匹配量mdatId的值,
- b、列表mdatIdForChunk中相应项匹配量trackId的值, 及
- c、“numEntries”的值匹配满足前述两条件的多个项。

(3) 将列表 offsetForChunk 中项 “chunk” 的值指派到量 “chunkOffset”。

(4) 将量“chunkOffset”的值作为32位整数写入mp4文件。

(5) 项“numEntries”的值增加1。

在完成对从属于“stco”元素2664所有元素的处理后, 在操作5060将mp4文件的文件位置指派给量“endPos”。将mp4文件的文件位置改变为由量“numEntriesPos”指定的值, 且将量“numEntries”的值作为32位整数写入mp4文件。接着将stco原子966的原子大小值更新为图50中显示的那样(操作5070至5095)。

7.2.4.10 处理stsz元素

使用标准XML手段获得从属于如图26B中所示的“stbl”元素2652的单个“stsz”元素2668。使用图50中所示的过程在输出mp4文件中创建具有原子ID“stsz”970的原子。在操作5000, 将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010, 将值0写入输出mp4文件替代原子大小值。在操作5020, 将原子ID“stsz”写入输出mp4文件。

“stsz”元素2668具有“版本”、“标志”、“sizeForAll”和“numSample”属性。在操作5030将这些属性中的每一个值指派给同名量。接着在操作5040, 将量“version”的值作为8位整数写入mp4文件及将量“version”的值作为24位整数写入mp4文件。

将列表StreamTypeForTrack中项“TrackNum”的值指派给量“streamType”。将列表ObjectTypeForTrack中项“TrackNum”的值指派给量“objectType”。量“trackNum”的值作为“处理trak元素”的部分

过程被确定。

如果量“streamType”的值是“1”且量“numOdsmSample”的值是“1”，那么将值“1”指派给量“numSample”且将列表OdsmSampleSize中的第一项的值指派给量“sizeForAll”。

如果量“streamType”的值是“3”且列表SdsmSampleSize中的第二项是否定的，那么将值“1”指派给量“numSample”且将列表SdsmSampleSize中的第一项的值指派给量“sizeForAll”。列表SdsmSampleSize中的否定项显示该列表结束。

如果量“streamType”的值是“5”，量“objectType”的值是“193”，且量“sizeForAll”的值小于1，那么将值“24”指派到量“numSample”且将列表SdsmSampleSize中的第一项的值指派给量“sizeForAll”，对象类型“193”定义的音频流中每个样本的大小是24位。

将mp4文件的文件位置指派给量“sizeForAllPos”，且将量“sizeForAllPos”的值作为32位整数写入mp4文件。将mp4文件的文件位置指派到量“numEntriesPos”，且将量“numSample”的值作为32位整数写入mp4文件。将值0指派给量“numEntries”。

“stsz”元素2668可具有从属“SampleSize”元素，但忽视这些，因为包含进这些元素的数据值不必与当前媒体数据流一致。替代地，当已（为odsm和sdsm）创建了流时，或当创建mdia原子（音频和视频流）时，已重新计算了所有的样本大小值。已将结果样本大小值输入到列表OdsmSampleSize、SdsmSampleSize或SampleSize中的一个。如果媒体数据具有统一的样本大小，由量“sizeForAll”指定每个样本的值。

如果量“streamtype”的值是“1”（odsm）且量“numOdsmSample”的值大于“1”，则将量“numOdsmSample”的值指派到量“numEntries”。量“numOdsmSample”的值显示了列表OdsmSampleSize中项的数量。将列表OdsmSampleSize中的每个项作为32位整数写入mp4文件。如果量“numOdsmSample”的值和量“numSample”的值不同，则将mp4文件的文件位置指派到量

“mp4FilePos”，将文件位置改变为由量“numEntriesPos”指定的值，将量“numOdsmsSample”的值作为32位整数写入mp4文件，且将文件位置恢复为量“mp4FilePos”指定的值。

如果量“streamtype”的值是“3”（sdsms）且列表SdsmsSampleSize中第二项不为负，则将列表SdsmsSampleSize中每个项的值作为32位整数写入mp4文件，直到找到负项。对于列表SdsmsSampleSize中每个非负项量“numEntries”的值增加1。该列表中最后的负值不写入mp4文件。如果量“numEntries”的值与量“numSample”不同，将mp4文件的文件位置指派到量mp4FilePos，将文件位置改变为量“numEntriesPos”指定的值，将量“numEntries”的值作为32位整数写入mp4文件，且将文件位置恢复为量“mp4FilePos”指定的值。

如果量“streamtype”的值不是“1”和“3”，则量“sizeForAll”的值不为正，且量“numSample”的值为负，量“numMediaSamples”的值被指派到量“numSamples”。这种情况包括大多数视频媒体数据和具有变化样本尺寸的音频媒体数据。量“numMediaSamples”指定了创建的列表“sampleSize”中的项数，然后创建mdia原子。在列表sampleSize中的每一项的值被作为32位整数写到mp4文件中。mp4文件的文件位置被指派到量“mp4FilePos”文件位置被改变到量“numEntriesPos”指定的值，量“numMediaSamples”的值被作为32位整数写入到mp4文件，且文件位置被恢复为量“mp4FilePos”指定的值。

如果量“streamType”的值不是“1”，也不是“3”，则量“sizeForAll”的值为负，且量“numSamples”的值为0，通过将该track的媒体数据的大小被量“sizeForAll”的值除，而确定量“numSamples”的值。这种情况包括某些具有均匀样本大小的音频媒体数据。将Mp4文件的文件位置指派给量“mp4FilePos”，将文件位置改为由量“numEntriesPos”指定的值，将量“numSamples”的值作为32位整数写入mp4文件，并将文件位置恢复到由量“mp4FilePos”所指定的值。

如果量“streamType”的值不是“1”，也不是“3”，则量“sizeForAll”的值为0，且量“numSamples”的值为正，通过将该track的媒体数据的

大小被量“numSamples”的值除，而确定量“sizeForAll”的值。这种情况包括某一具有均匀样本大小和已知样本计数（通常为“1”）的媒体数据，诸如表示单个图像的媒体数据。将Mp4文件的文件位置指派给量“mp4FilePos”，将文件位置改为由量“sizeForAllPos”指定的值，将量“sizeForAll”的值作为32位整数写入mp4文件，并将文件位置恢复到由量“mp4FilePos”所指定的值。

在所有的情况下，通过将stsz原子970的原子大小的值更新为图50中所示出的那样而完成对“stsz”元素2668的处理（操作5060到5095）。

7.2.4.11 处理stss元素

“stss”元素2672（如果存在的话）包含“同步样本表”。该表对于某一类型的媒体数据流（诸如MPEG-4视频流）是所需的。“stbl”元素2652仅当需要时才包含从属“stss”元素2672。使用标准XML手段确定从属“stss”元素2672是否存在于“stbl”元素2652内。

如果从属“stss”元素2672存在，使用标准XML手段获得该元素。图50中示出的过程用于在输出mp4文件中创建具有原子ID“stss”974的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“stss”写入输出mp4文件。

“stss”元素2672具有属性“版本”、“标志”和“numEntries”。在操作5030，将这些属性的每一个的值指派给同名量。在操作5040，将量“version”的值作为8位整数写入mp4文件，将量“标志”的值作为24位整数写入mp4文件。

将mp4文件的文件位置指派给量“numEntriesPos”，并将32位0值写入mp4文件。

然后，使用图54中示出的手段在输出mp4文件中构建同步样本表。根据该过程，在操作5400，将列表“MediaSample”中的项“trakNum”指派给量“iMediaSamples”的值。该量指示表示与当前trak原子790和900相关的媒体数据流的样本（或“访问单元”）数。该量的

值还提供列表“同步样本”中的项数的上界。列表“iSyncSample”包括单调递增的样本索引值集合，其中媒体数据流中的第一个样本由样本索引值“1”表示。如果该列表中的项数小于iMediaSamples的值，则该流中的最后一个样本索引值之后有具有值0的项。

在操作 5410，将值 0 指派给量“iSyncSample”和“numSyncSamples”的值。

在操作5420，将numSyncSample的值与iMediaSample的值进行比较。

如果量“numSyncSamples”的值小于量“iMediaSample”的值，则在操作5430该过程完成。

否则，即量“numSyncSamples”的值小于量“iMediaSample”的值，则在操作5440将量“iSyncSample”的值指派给量“previousSample”的值。在操作 5450，将项“numSyncSamples”的值指派给量“iSyncSample”，在操作5460，将所得到的量“iSyncSample”的值与量“iSyncSample”的值进行比较。

如果量“iSyncSample”的值不大于量“iSyncSample”的值，则该过程在操作 5470 完成。否则，即量“iSyncSample”的值大于量“iSyncSample”的值，则在操作5480将量“iSyncSample”的值作为32位整数写入输出mp4文件。在操作5490，将量“numSyncSamples”的值增加 1，并重复操作 5420 的量“numSyncSamples”的值与量“iMediaSamples”的值的比较。

在完成图54中所示的过程之后，在操作5060，将mp4文件的文件位置指派给量“endPos”。将 mp4 文件的文件位置改为量“numEntriesPos”指定的值，并将量“numSyncSamples”的值作为32位整数写入mp4文件。然后，将stss原子974的原子大小的值更新为图50中所示的那样（操作5070到5095）。

7.2.4.12 处理stsd元素

使用标准 XML 手段获得从属于图 26B 中所示的“stbl”2652 元素

的单个“stsd”元素 2676。图 50 中示出的过程用于在输出 mp4 文件中创建具有原子 ID“stsd”978 的原子。在操作 5000，将输出 mp4 文件的当前文件位置指派给量“sizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值。在操作 5020，将原子 ID“stsd”写入输出 mp4 文件。

“stsd”元素 2676 具有属性“版本”和“标志”。在操作 5030，将这些属性的每一个的值指派给同名量。然后，在操作 5040，将量“version”的值作为 8 位整数写入 mp4 文件，并将量“flags”的值作为 24 位整数写入 mp4 文件。

将 mp4 文件的文件位置指派给量“numEntriesPos”，将 32 位 0 值写入 mp4 文件，并将值 0 指派给量“numEntries”。将量“sizePos”的值指派给量“stsdSizePos”。

然后，在操作 5050，使用标准 XML 手段获得从属于“stsd”元素 2676 的所有元素。这种从属元素的集合被期望包括类型“mp4s”、“mp4a”或“mp4v”的单个元素。这些元素类型在图 26B 中统一描述为“mp4*”2680。这些从属元素的每一个表示“样本描述表”中项，并将每一个这样的从属元素的量“numEntries”的值增加 1。

这些类型的元素中的每一个（“mp4s”、“mp4a”或“mp4v”）具有称作“dataRefIndex”的单个属性。

对于每一个类型“mp4s”类型的元素，图 50 中示出的过程用于在输出 mp4 文件中创建具有原子 ID“mp4s”982 的原子。在操作 5000，将输出 mp4 文件的当前文件位置指派给量“sizePos”，并将量“sizePos”的值指派给量“mp4xSizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值。在操作 5020，将原子 ID“mp4s”写入输出 mp4 文件。

向量“dataRefIndex”指派值 0。如果对于 mp4s 元素的“dataRefIndex”属性指定了一个值，则在操作 5030 将该属性的值指派给量“dataRefIndex”。然后，在操作 5040，将下列值写入 mp4 文件：

1. 具有值 0 的 6 字节

2. 表示量“dataRefIndex”的值的 16 位整数

对于每一个“mp4a”类型的从属元素，使用图 50 中示出的过程在输出 mp4 文件中创建具有原子 ID“mp4a”982 的原子。在操作 5000，将输出 mp4 文件的当前文件位置指派给量“sizePos”，并将量“sizePos”的值指派给量“mp4xSizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值。在操作 5020，将原子 ID“mp4a”写入输出 mp4 文件。

向量“dataRefIndex”指派值 0。如果对 mp4a 元素的“dataRefIndex”属性指定了一个值，则在操作 5030，将该属性的值指派给量“dataRefIndex”。然后，在操作 5040，将下列值写入 mp4 文件：

1. 具有值 0 的 6 字节
2. 表示量“dataRefIndex”的值的 16 位整数
3. 具有值 0 的两个 32 位整数
4. 表示值“2”的 16 位整数
5. 表示值“16”的 16 位整数
6. 具有值 0 的 32 位整数
7. 表示从“mdia”元素 2604 的相应属性获得的量“timeScale”的值的 16 位整数
8. 具有值 0 的 16 位整数

对于每一个“mp4v”类型的从属元素，使用图 50 中示出的过程在输出 mp4 文件中创建具有原子 ID“mp4v”982 的原子。在操作 5000，将输出 mp4 文件的当前文件位置指派给量“sizePos”，并将量“sizePos”的值指派给量“mp4xSizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值。在操作 5020，将原子 ID“mp4v”写入输出 mp4 文件。

向量“dataRefIndex”指派值 0。如果对 mp4v 元素的“dataRefIndex”属性指定了一个值，则在操作 5030，将该属性的值指派给量“dataRefIndex”。然后，在操作 5040，将下列值写入 mp4 文件：

1. 具有值 0 的 6 字节

2. 表示量“dataRefIndex”的值的 16 位整数
3. 具有值 0 的两个 32 位整数
4. 具有值“320”的 16 位整数
5. 具有值“240”的 16 位整数
6. 具有值“72”的 16 位整数
7. 具有值 0 的 16 位整数
8. 具有值“72”的 16 整数
9. 具有值 0 的 16 位整数
10. 具有值 0 的 32 位整数
11. 具有值“1”的 16 整数
12. 具有值 0 的 32 字节
13. 具有值“24”的 16 位整数
14. 具有值“-1”的 16 位整数

每一个“mp4s”、“mp4a”或“mp4v”类型的元素 2680 被期望具有类型“esds”的单个从属元素 2684，如图 26B 所示。对于每一个“esds”类型的从属元素，使用图 50 中示出的过程在输出 mp4 文件中创建具有原子 ID“esds”986 的原子。在操作 5000，将输出 mp4 文件的当前文件位置指派给量“sizePos”，并将量“sizePos”的值指派给量“mp4xSizePos”。在操作 5010，将值零写入输出 mp4 文件，替换原子大小值。在操作 5020，将原子 ID“esds”写入输出 mp4 文件。

“stsd”类型的元素具有属性“版本”和“标志”。在操作 5030，将这些属性的每一个的值指派给同名量。然后，在操作 5040，将量“version”的值作为 8 位整数写入 mp4 文件，并将量“flags”的值作为 24 位整数写入 mp4 文件。

每一个“esds”类型的元素被期望具有类型“ES_Descr”类型的单个从属元素 2688，如图 26B 中所示。使用下面在操作 5050 描述的手段对该从属“ES_Descr”元素进行处理。

在完成对从属于“esds”元素 2684 的“ES_Descr”元素 2688 的处理之后，在操作 5060，将 mp4 文件的文件位置指派给量“endPos”。将

量“esdsSizePos”的值指派给量“sizePos”，并将 esds 原子 986 的原子大小的值更新为图 50 所示的那样（操作 5070 到 5095）。

将量“mp4xSizePos”的值指派给量“sizePos”，并将“mp4s”、“mp4a”或“mp4v”的原子大小的值更新为图 50 中所示的那样（操作 5070 到 5095）。

将 mp4 文件的文件位置改为量“numEntriesPos”所指定的值，并将量“numEntries”的值作为 32 位整数写入 mp4 文件。将量“stsdSizePos”的值指派给“sizePos”。然后，将 sdsm 原子 978 的原子大小的值更新为图 50 中所示的那样（操作 5070 到 5095）。

7.2.4.13 处理 ES_Descr 元素

对于从属于“esds”元素 2684 的每一个“ES_Descr”元素 2688 和 2700，图 51 中示出的过程用于在输出 mp4 文件中创建 ES_Descr 对象结构 900 和 1000。在操作 5100，将结构标签“ES_DescrTag”（值 = 3）1004 作为 8 位整数写入输出 mp4 文件中。在操作 5110，将输出 mp4 文件的当前文件位置指派给量“sizePos”，并将“sizePos”的值指派给量“filePos1”。在操作 5120，向量“numSizeBytes”指派值“1”。在操作 5130，将值 0 作为初步大小值 1008 写入输出 mp4 文件。

为“ES_Descr”元素定义下列属性：“ES_ID”和“优先级”。向称作“ES_ID”和“优先级”的量指派值 0。如果对于这两个属性的任一个指定了值，则在操作 5135，将所指定的值指派给同名量。

在操作 5140，将下述值写入 mp4 文件：

1. 表示量“ES_ID”的值的 16 位整数
2. 具有值 0 的三位整数
3. 表示量“优先级”的值的 5 位整数

然后，使用标准 XML 手段获得从属于“ES_Descr”元素 2688 和 2700 的每一个 XML 元素。这些从属元素被期望包括具有元素名“DecoderConfigDescriptor”的一个元素 2710、和具有元素名“SLConfigDescriptor”的一个元素 2760，如图 27 中所示的那样。

对于从属于“ES_Descr”元素 2700 的每一个“DecoderConfigDescriptor”元素 2710，图 51 中示出的过程用于在输出 mp4 文件中创建 DecoderConfigDescriptor 对象结构 1024 和 1032。在操作 5100，将结构标签“DecoderConfigDescrTag”（值 = 4）1036 作为 8 位整数写入输出 mp4 文件中。在操作 5110，将输出 mp4 文件的当前文件位置指派给量“sizePos”，并将“sizePos”的值指派给量“filePos2”。在操作 5120，向量“numSizeBytes”指派值“1”。在操作 5130，将值 0 作为初步大小值 1040 写入输出 mp4 文件。

为“DecoderConfigDescriptor”元素 2710 定义下列属性：“对象类型”、“流类型”、“upStream”、“maxBitrate”和“avgBitrate”。如果对这些属性中的任何一个指定了值，则在操作 5135，将每一个指定的值指派给同名量。

量“bufferSize”的值由相应媒体数据流中最大样本的大小确定。如果量“streamType”的值为“1”，则将列表“OdsMSampleSize”中的最大项的值指派给量“bufferSize”。如果量“streamType”的值为“3”，则将列表“SdsMSampleSize”中的最大项的值指派给量“bufferSize”。如果量“streamType”的值为“4”且量“objectType”的值为“32”、或量“streamType”的值为“5”且量“objectType”的值为“64”、或量“streamType”的值为“5”且量“objectType”的值为“107”，则将列表“sampleSize”中的最大项的值指派给量“bufferSize”。如果“streamType”的值为“4”且量“objectType”的值为“193”，则向量“bufferSize”指派值“2400”。否则，媒体数据的值由列表 mediaDataSize 中的项的和所确定，其中列表 trackIdChunk 中的相应项与当前 track 的量“trackId”的值匹配。

在操作 5140，将下列值写入 mp4 文件：

1. 表示量“objectType”1044 的整数值的一字节
2. 表示量“streamType”1048 的整数值 6 位
3. 表示量“upStream”1052（如果真则为“1”，否则为“0”）的布尔值的一位

4. 具有值“1”的一位（作为“保留”）1056
5. 表示量“bufferSize”1060的整数值的三字节（作为“bufferSizeDB”）
6. 表示量“maxBitrate”1064的值32位整数
7. 表示量“avgBitrate”1068的值的32位整数

然后，使用标准 XML 手段获得从属于“DecoderConfigDescriptor”元素2710的任何元素。这可包括下列类型元素中的一个：“BIFS_DecoderConfig”2720、“JPEG_DecoderConfig”2730、“VisualConfig”2740、“AudioConfig”2750，如图27所示的那样。如果发现了这些从属元素的任何一个，则使用图51中所示的过程在输出mp4文件中创建DecoderSpecificInfo对象结构1072和1076。在操作5100，将结构标签“DecoderSpecificInfoTag”（值=5）1080作为8位整数写入输出mp4文件中。在操作5110，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5120，向量“numSizeBytes”指派值“1”。在操作5130，将值0作为初步大小值1084写入输出mp4文件。

每一类型的解码器详细信息元素的属性在下面描述。在这些属性已在操作5150处理之后，将DecoderSpecificInfo对象结构1076的大小值（numBytes）1084更新为图51中所示的那样（操作5160到5195）。将量“filePos2”的值指派给量“sizePos”，并将DecoderConfigDescriptor对象结构1032的大小值（numBytes）1040更新为图51中所示的那样（操作5160到5195）。

对于从属于“ES_Descr”元素2700的每一个“SLConfigDescriptor”元素2760，图51中示出的过程用于在输出mp4文件中创建SLConfigDescriptor对象结构1028和1088。为“SLConfigDescriptor”元素2760定义下列属性：“predefined”。如果对属性“predefined”指定了值，则将该属性的值指派给“predefined”量。否则，向量“predefined”指派值“2”。

将下列值写入mp4文件：

1. 具有“SLConfigDescrTag”的值（值 = 6）的一字节
2. 具有值“1”（numBytes）的一字节1094
3. 表示量“predefined”的值的一字节1098

在对这些从属元素中的每一个进行处理之后，将量“filePos1”的值指派给量“sizePos”，并将ES_Descr结构1000的大小值（numBytes）1008更新为图51中所示的那样（操作5160到5195）。

7.2.4.14 处理BIFS DecoderConfig元素

“BIFS_DecoderConfig”元素2720可具有“版本1”或“版本2”属性，这取决于量“objectType”的值。如果量“objectType”的值不为“2”，则为“BIFS_DecoderConfig”元素（版本1）定义下列属性：“nodeIdBits”、“routeIdBits”、“pixelMetric”、“pixelWidth”和“pixelHeight”。

在操作5135，将为这些属性指定的值指派给同名量。然后，在操作5140，将下列值写入mp4文件：

1. 表示量“nodeIdBits”的整数值的5位
2. 表示量“routeIdBits”的整数值的5位
3. 具有值“1”的1位
4. 由量“pixelMetric”的布尔值（1为“真”，否则为“0”）所确定的1位
5. 具有值“1”的1位
6. 具有量“pixelWidth”的整数值的16位整数
7. 具有量“pixelHeight”的整数值的16位整数
8. 具有值0的3位

如果量“objectType”的值为“2”，则为“BIFS_DecoderConfig”元素（版本2）定义下列属性：“nodeIdBits”、“routeIdBits”、“protoIdBits”、“use3DMeshCoding”、“usePredictiveMFField”、“pixelMetric”、“pixelWidth”和“pixelHeight”。

在操作5135，将为这些属性中的每一个指定的值指派给同名量。然后，在操作5140，将下列值写入mp4文件：

1. 由量“use3DMeshCoding”的布尔值（1为“真”，否则为“0”）确定的1位
2. 由量“usePredictiveMFField”的布尔值（1为“真”，否则为“0”）确定的1位
3. 表示量“nodeIdBits”的整数值的5位
4. 表示量“routeIdBits”的整数值的5位
5. 表示量“protoIdBits”的整数值的5位
6. 具有值“1”的1位
7. 由量“pixelMetric”的布尔值（1为“真”，否则为“0”）确定的1位
8. 具有值“1”的1位
9. 具有量“pixelWidth”的整数值的16位
10. 具有量“pixelHeight”的整数值的16位
11. 具有值0的4位

7.2.4.15 处理JPEG DecoderConfig元素

为“JPEG_DecoderConfig”元素定义下列属性：“headerLength”、“Xdensity”和“Ydensity”。

向同名量指派缺省值0、1和1。在操作5135，将为这些属性指定的值指派给同名量。

根据与该track相关的媒体数据的内容，向量“numComponents”指派值“1”或“3”。如果相关媒体数据表示灰度级图像，则在操作5135，向量“numComponents”指派值“1”，否则，向量“numComponents”指派值“3”。

然后，在操作5140，将下列值写入mp4文件：

1. 表示量“headerLength”的值的16位整数
2. 表示量“Xdensity”的值的16位整数
3. 表示量“Ydensity”的值的16位整数
4. 表示量“numComponents”的整数值的1字节

7.2.4.16 处理 VisualConfig 元素

为“VisualConfig”元素定义下列属性：
“profile_and_level_indication”。

向同名量指派缺省值1。如果为该属性指定了一个值，在操作5135，将指定的值指派给同名量。

对特定于该类型的解码器的信息的进一步处理依赖于当该track的mdat原子创建时从相应媒体数据中提取出的“mediaHeader”数据。

如果媒体头数据对该track可用，则测试该媒体数据“可视对象序列起始码”（0x000001b0）和“可视对象序列结束码”（0x000001b1）的存在。

如果媒体头数据对于该track可用，但该媒体数据不包括“可视对象序列起始码”，或如果没有媒体头数据对于该track可用，则在操作5140，将下列值写入mp4文件：

1. 具有值0x000001b0（可视对象序列起始码）的32位整数
2. 表示量“profile_and_level_indication”的值的一字节
3. 表示值0x000001bs（可视对象起始码）的32位整数
4. 具有值“9”的一字节

如果没有媒体头数据对于该track可用，则在操作5140将下列值写入mp4文件：

5. 具有值0x00000100（可视对象起始码）的32位整数

如果媒体头数据对于该track可用，则该媒体头数据的最后四字节不是“可视对象序列结束码”，则在操作5140将该媒体头数据的剩余部分复制到mp4文件中。

如果媒体头数据对于该track可用，但该媒体头数据的最后四字节包括“可视对象序列结束码”，则在操作5140，将该媒体头数据的剩余部分除了最后四字节之外复制到mp4文件。

7.2.4.17 处理 AudioConfig 元素

在操作5135，不为“AudioConfig”元素指定属性。从该track的媒体数据中提取出的媒体头数据中提取出要写入mp4文件的所有数据值。

将媒体头数据的第一字节指派给量“audioObjectType”。将媒体头数据的第二个字节指派给量“sampleRateIndex”。将该媒体头数据的第三字节指派给量“channelConfig”。然后，在操作5140，将下列值写入mp4文件：

1. 表示量“audioObjectType”的整数值的5位
2. 表示量“sampleRateIndex”的整数值的4位
3. 表示量“channelConfig”的整数值的4位
4. 具有值0的3位

如果量“channelConfig”的值为0，则将“程序配置元素”（PCE）添加到解码器详细信息中。PCE的结构定义于对于编码音频数据的MPEG-4规范中。创建PCE所需的数据包含在媒体头数据数组中。该数据当相应mdat原子被创建时存储于该数据中。

7.2.4.18 处理媒体头元素

使用标准XML手段获得从属于“mdia”元素2604的单个媒体头元素（“*mhd”）2632，如图26A中所示。媒体头元素2632可以是“nmhd”元素（用于sdsm或odsm）、“smhd”元素（用于音频流）、或“vmhd”元素（用于视频流）。

如果媒体头元素2632为“nmhd”元素，则将图50中示出的过程用于在输出mp4文件中创建具有原子ID“nmhd”936的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“nmhd”写入输出mp4文件。

“nmhd”元素具有属性“version”和“flags”。在操作5030，这些属性中每一个属性值被指派给同名量。

然后，在操作5040，将下列量的值写入输出mp4文件：

1. 表示量“version”的值的8位整数

2. 表示量“flags”的值的24位整数

如果媒体头元素2632为“smhd”元素，则将图50中示出的过程用于在输出mp4文件中创建具有原子ID“smhd”936的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“smhd”写入输出mp4文件。

“smhd”元素具有属性“version”、“标志”和“balance”。在操作5030，将这些属性的每一个的值指派给同名量。

在操作5040，将下列量的值写入输出mp4文件：

1. 表示量“version”的值的8位整数

2. 表示量“flags”的值的24位整数

3. 表示量“balance”的值的16位整数

4. 具有值0的16位整数

如果媒体头元素2632为“vmhd”元素，则将图50中示出的过程用于在输出mp4文件中创建具有原子ID“vmhd”936的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“vmhd”写入输出mp4文件。

“vmhd”元素具有属性“版本”、“标志”、“transferMode”、“opColorRed”、“opColorGreen”、和“opColorBlue”。在操作5030，将这些属性的每一个的值指派给同名量。

在操作5040，将下列量的值写入输出mp文件：

1. 表示量“version”的值的8位整数

2. 表示量“flags”的值的24位整数

3. 表示量“transferMode”的值的16位整数

4. 表示量“opColorRed”的值的16位整数

5. 表示量“opColorGreen”的值的16位整数

6. 表示量“opColorBlue”的值的16位整数

在操作5050，媒体头元素2632不具有从属元素。在写入了媒体头原子的属性值之后，将该媒体头原子936的原子大小的值更新为图50中所示的那样（操作5060到5095）。

7.2.4.19 处理tref元素

使用标准XML手段获得可能从属于“trak”元素2600的单个“tref”元素2636，如图26A所示。将图50中示出的过程用于在输出mp4文件中创建具有原子ID“tref”940的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“tref”写入输出mp4文件。将量“sizePos”的值指派给量“trefSizePos”。

在操作5030和5040，“tref”元素不具有属性。

使用标准XML手段获得从属于“tref”元素2636的每一个元素。如图26A中所示，这些可以包括“mpod”元素2640。其他类型的包括“dpnd”元素和/或“sync”元素的元素可能还作为“tref”元素2636的从属元素。

对于每一个从属于“tref”元素2636的“mpod”、“dpnd”或“sync”元素2640，将图50中示出的过程用于在输出mp4文件中创建具有同值原子ID942的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“mpod”、“dpnd”或“sync”写入输出mp4文件。

每一个“mpod”、“dpnd”或“sync”元素具有一个属性，称作“trackID”。在操作5030，该属性包括trackID值的列表。在操作5040，将该列表中的每一个trackID值作为32位整数写入输出mp4文件。在“mpod”元素的情况下，还将每一个trackID值指派给列表“TrackIdForOdId”中的一项。

在操作5050，“mpod”、“dpnd”或“sync”元素不具有从属元素。在对该元素的trackID属性进行了处理之后，将mp4文件中相应原子

942的原子大小的值更新为图50中所示的那样（操作5060到5095）。

在完成对从属于“tref”元素2036的所有元素2640的处理之后，将量“trefSizePos”的值指派给“sizePos”，并将tref原子940的原子大小的值更新为图50中所示的那样（操作5060到5095）。

7.2.4.20 处理edts元素

使用标准XML手段获得可能从属于“trak”元素2600的单个“edts”元素2644，如图26A所示。将图50中示出的过程用于在输出mp4文件中创建具有原子ID“edts”945的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“edts”写入输出mp4文件。将量“sizePos”的值指派给量“edtsSizePos”。

在操作5030和5040，“edts”元素不具有属性。

使用标准XML手段获得从属于“edts”元素2644的单个“elst”元素2648，如图26A中所示。将图50中示出的过程用于在输出mp4文件中创建具有原子ID“elst”948的原子。在操作5000，将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010，将值零写入输出mp4文件，替换原子大小值。在操作5020，将原子ID“elst”写入输出mp4文件。

“elst”元素具有属性“版本”和“标志”。在操作5030，将这些属性的中的每一个的值指派给同名量。

在操作5040，将下列量的值写入输出mp4文件：

1. 表示量“version”的值的8位整数，及
2. 表示量“flags”的值的24位整数。

使用标准XML手段获得从属于“elst”元素2648的每一个元素。从属于“elst”元素2648的元素集合被期望包括两个“segment”元素。每一个“segment”元素具有三个属性“duration”、“startTime”和“rate”。

对从属于“elst”元素2648的每个片段元素执行下列操作：

1. 将每个属性“duration”、“startTime”和“rate”的值指派给同名量，

2. 将量“duration”的值作为32位整数写入输出mp4文件,
3. 将量“startTime”的值作为32位整数写入输出mp4文件, 及
4. 量“rate”的浮点值乘以256*256, 转换为整数, 且将该结果作为32位整数写入mp4文件。

处理完从属于“elst”元素2648的所有片段元素后, 将elst原子948的原子大小的值更新为图50中所示的那样(操作5060到5095)。

将量“edtsSizePos”的值指派给“sizePos”, 并将edts原子945的原子大小的值更新为图50中所示的那样(操作5060到5095)。

7.2.5 处理可选用户数据元素

创建输出mp4文件2230中的第四步包括对包含在moov元素2320中的任何可选“user data”(udta)元素2340进行处理。使用标准XML手段获得从属于图23A中示出的mp4file文档2300的“moov”元素2320的任何“udta”元素2340。使用下列方法处理每种“udta”元素:

图50中示出的过程用于在输出mp4文件中创建具有原子ID“udta”的原子784。在操作5000, 将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010, 将值零写入输出mp4文件, 替换原子大小值760。在操作5020, 将原子ID“udta”写入输出mp4文件。将“sizePos”的值指派给量“udta SizePos”。

在操作5030和5040, “udta”元素不具有属性。

每个“udta”元素可以具有诸如可用于在mp4文件中嵌入版权信息的“cpri”元素的从属元素。可能忽略任何未识别的从属元素。

如果发现“udta”元素具有从属“cpri”元素, 则使用图50中示出的过程在输出mp4文件中创建具有原子ID“cpri”的原子。在操作5000, 将输出mp4文件的当前文件位置指派给量“sizePos”。在操作5010, 将值零写入输出mp4文件, 替换原子大小值。在操作5020, 将原子ID“cpri”写入输出mp4文件。

“cpri”元素可具有名称为“version”、“flags”和“language”的属性。在操作5030, 这些属性中的每一个值指派给同名量。在操作5040,

“cp_{rt}”元素也将具有包含了表示消息的文本字符串的从属“text node”。

在操作5040，将下列量写入输出mp4文件中：

- a) 表示量“version”值的8 位整数，
- b) 表示量“flags”值的24 位整数，
- c) 表示量“language”值的16 位整数，及
- d) 表示从属文本节点值的序列字段，随后为空字节。

完成“cp_{rt}”元素的属性后，将cp_{rt}原子的原子大小的值更新为图50中所示的那样（操作5060到5095）。

完成对从属于“udta”元素2340的所有mp4文件元素进行处理后，将量“udtaSizePos”的值指派给“sizePos”，并将相应的udta原子784的原子大小的值更新为图50中所示的那样（操作5060到5095）。

7.2.6 更新odsm缓存大小

创建输出 mp4 文件 2230 中的最后步骤包括对 odsm 缓存大小进行更新。

输出 mp4 文件 2320 包括用于每个媒介流的 trak 原子 790。如图 9 中所示，每个 trak 原子 900 包括 ES-Descr 对象结构 990。ES-Descr 对象结构 990 和 1000 包含 DecoderConfigDescriptor 对象结构 1024 和 1032。如图 10 中所示，每个 DecoderConfigDescriptor 对象结构包括属性“bufferSizeDE”1060。在大部分情况下，在相关媒介数据流中由最大样本的大小来确定该属性值。在 odsm1900 的情况中，每个样本 1920 和 1940 可包括关于 Es-Descr 结构 1000 的 EsIdRef 结构 2170，及 odsm 样本缓存必须具有足够的大小以允许相应 Es-Descr 结构 1000 来取代每个嵌入 EsIdRef2160 结构。构成 Es-Descr 结构 1000 的字节数一般大于相应的 EsIdRef 结构 2160 的字节数。因此，必须增加 odsm 的最小最小缓存大小以允许相应 Es-Descr 结构 1000 来取代嵌入 EsIdRef 结构 2160。由下述方法实现。

执行这些操作前，如上述已构建每个 track 原子 900。odsm 的

track 原子包含用于 **bufferSizeDB** 属性的预备值。在将该预备值写入输出 **mp4** 文件之前，将 **mp4** 文件位置的值指派给量“**OdsmbufferSizeDB**”。另外，如上述的部分操作，已将每个 **odsm** 样本大小指派给列表“**OdsmbufferSize**”中的项，已将用于每个 **track** 原子的 **Es-Descr** 结构的大小指派给列表“**EsDescrSizeForTrack**”中的项，已将用于每个 **track** 原子的“**trackID**”属性的大小指派给列表“**TrackIdForTrack**”中的项，及已将联系到每个对象的 **TrackId** 的大小指派到列表“**TrackIdForOdId**”中的项。

在完成上述前面的步骤的处理之后，使用下列标准手段修订 **odsm** 的属性 **bufferSizeDB** 的值：

1. 使用标准 XML 手段获得 **mp4file** 文档 2300 中的每一个“**mdat**”元素 2310
2. 使用标准 XML 手段获得从属于每一个“**mdat**”元素 2310 和 2400 的每一个“**odsm**”元素 2420
3. 使用标准 XML 手段获得从属于每一个“**odsm**”元素 2420 和 2460 的每一个“**odsmChunk**”元素 2470
4. 使用标准 XML 手段获得从属于每一个“**odsmChunk**”元素 2470 和 2500 的每一个“**odsmSample**”元素（2510）
5. 列举具有量“**ithOdsSample**”的每一个“**odsmSample**”元素 2510
6. 使用标准 XML 手段获得从属于每一个“**odsmSample**”元素 2520 和 2510 的每一个“**ObjectDescrUpdate**” **odsm** 命令元素 2530 和 2540
7. 使用标准 XML 手段获得从属于每一个“**ObjectDescrUpdate**”元素 2540 的每一个“**ObjectDescriptor**”元素 2550
8. 将“**ObjectDescriptor**”元素 2550 的“**ODID**”属性的值指派给量“**OdId**”
9. 将列表“**TrackIdForOdId**”中的项 **OdId-1** 的值指派给量

“trackID”

10. 使用列表“TrackIdForTrack”确定“trackID”的值的“track”索引

11. 将列表“EsDescrSizeForTrack”中的项“track”的值指派给量“EsDescrSize”的值

12. 将量“EsDescrSize”的值添加到列表“OdsampleSize”中的项“IthOdsample”

13. 确定列表“OdsampleSize”中的最大项，并将结果指派给量 OdsampleSize

14. 将当前 mp4 文件位置指派给量“mp4FilePos”

15. 将当前文件位置改为量“OdsampleSizePos”所指出的值

16. 将表示量 OdsampleSize 的值的三字节作为 24 位整数写入输出 mp4 文件

17. 将 mp4 文件位置恢复到量“mp4FilePos”所指出的值

量 OdsampleSize 的值可能将对 odsm 缓冲区大小需要的值估计过高了，但这是可以接受的。这些手段完成了输出 mp4 文件的创建。

已为了示例和说明的目的，呈现了本发明的上述说明。其不是要穷举本发明或将本发明限制为所公开的确切形式，根据上面的教导其他的修改和变换都是可能的。例如，XMT-A 文件的精确定义可以随时间而变化或发展。同样，MPEG-4 中间格式的确切定义可以随时间而变化或发展。这里所描述的本发明不限于上面所述文档中指出的特定定义。因此，本发明的原理可应用于其他不相关的数据结构。SFNode 数据结构的其它期望形式也可定义于 MPEG-4 系统规范中。如在以下实施例中所述的扩展本发明以涵盖这种情况的手段对于本领域的技术人员来说是显而易见的。因此，所公开的实施例是为了更好地解释本发明的原理和其实际应用而选出的，从而使本领域的其他技术人员能够更好地以适于所设想的特定使用的各种实现和各种变体来应用本发明。所附权利要求旨在被认为包括除了在先有技术所限制的范围之外的本发明的其它可选实施例。

图1A

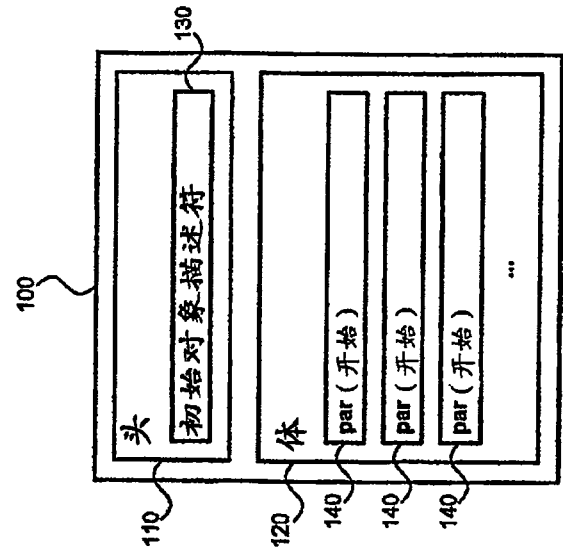


图1B

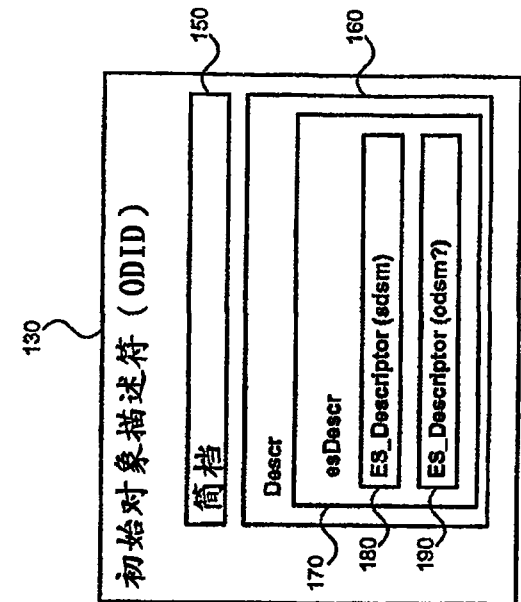


图 2A

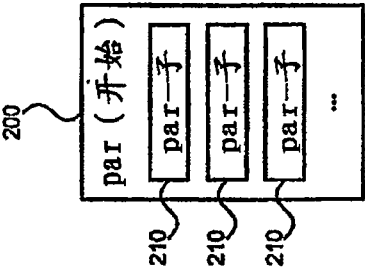


图 2B

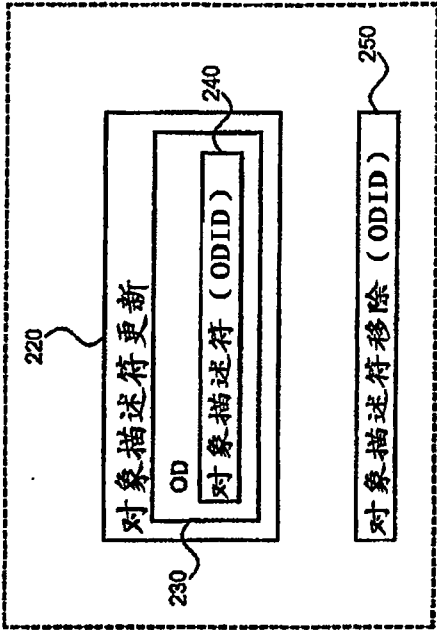


图 3C

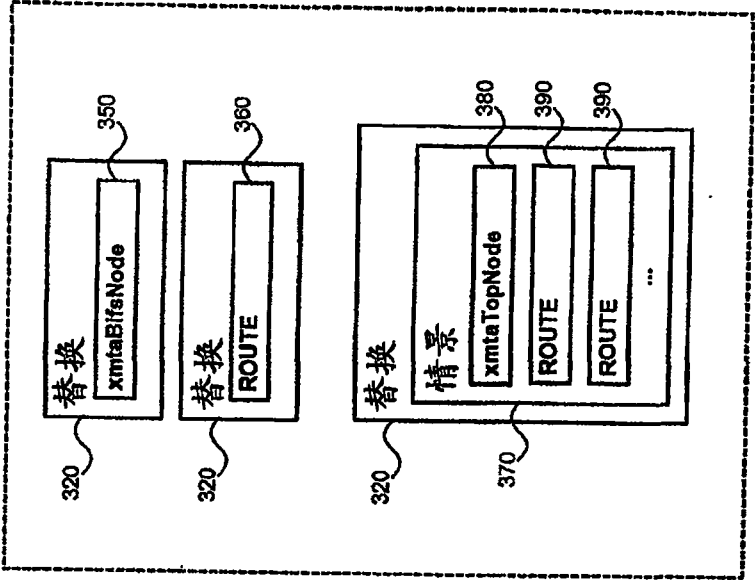


图 3A

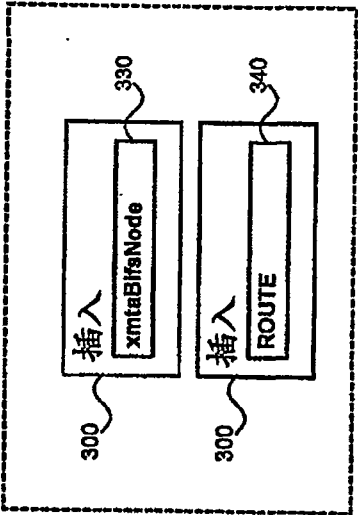


图 3B



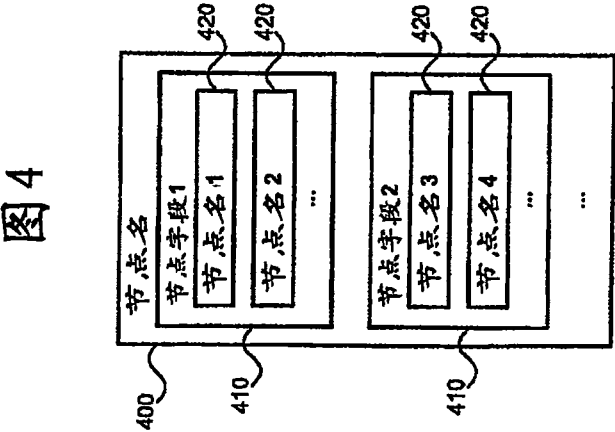


图 5A

500 ~<nodeName
510 ~DEF = "nodeIDstring"
520 ~field0 = "value0"
520 ~field2 = "value2"
520 ~field3 = "value3"
520 ~field5 = "value5" ...>
530 ~</field1>
540 ~<nodeName1 ... />
550 ~<nodeName2 ... />
540 ~...
530 ~</field4>
550 ~<nodeName3 ... />
540 ~...
560 ~</field4>
560 ~</nodeName>

图 5B

570 ~<nodeName
580 ~USE = "nodeIDstring" />
590 ~

图 6A

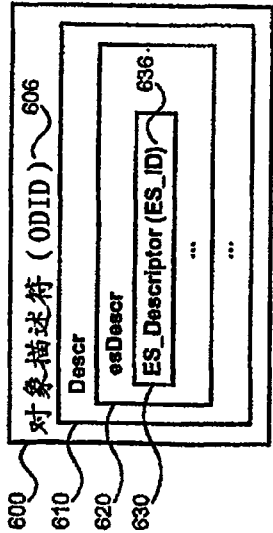


图 6B

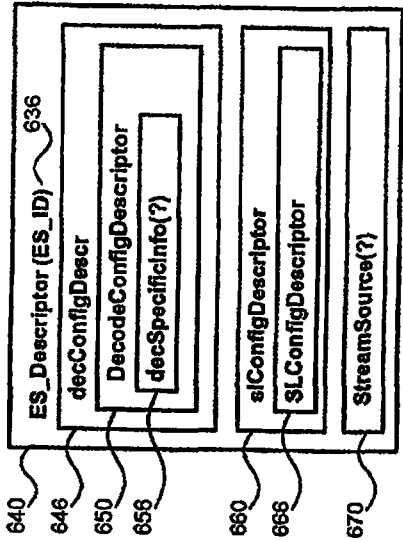
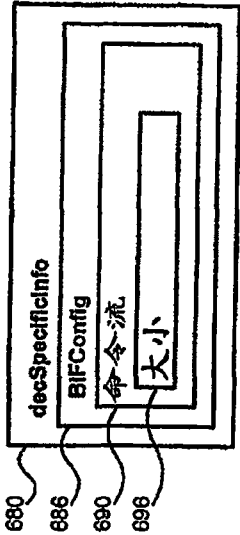


图 6C



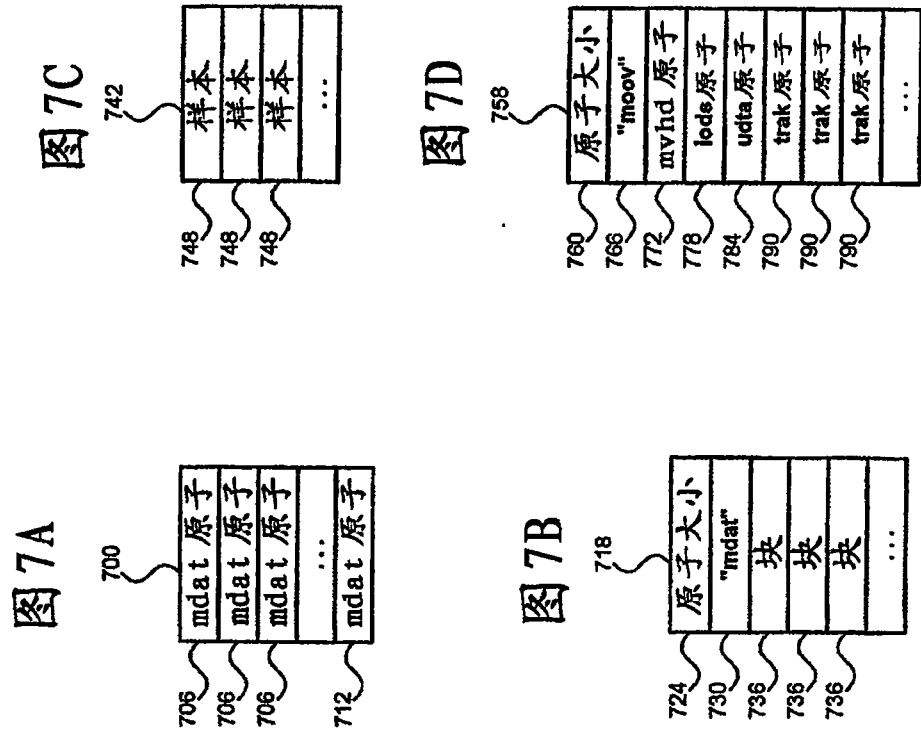


图 8A

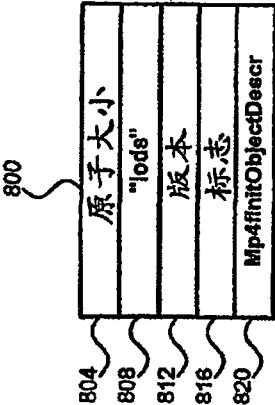


图 8B

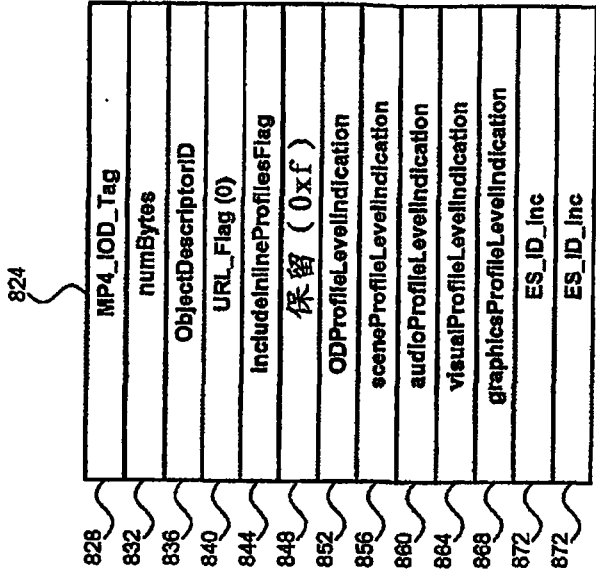


图 8C

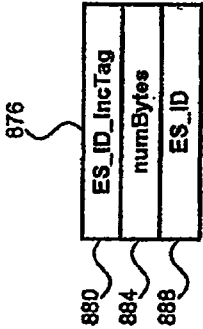


图9A

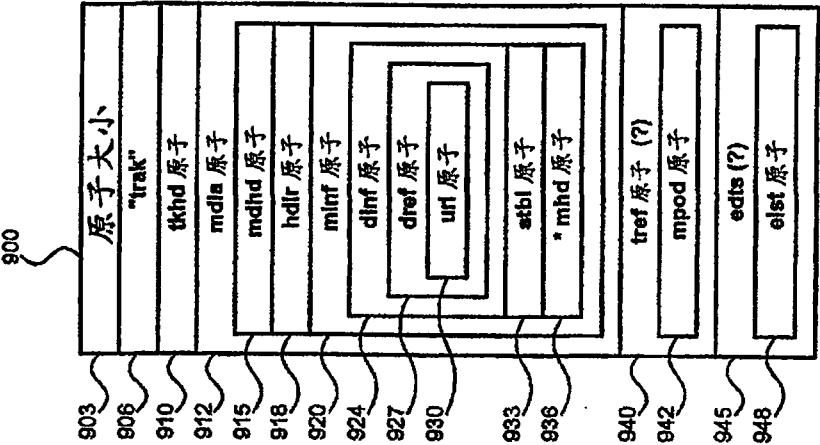


图9B

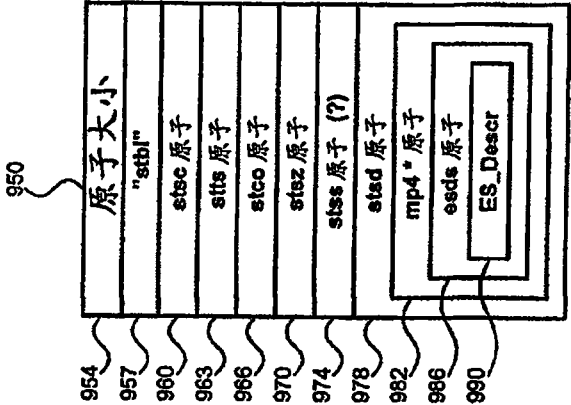


图10A

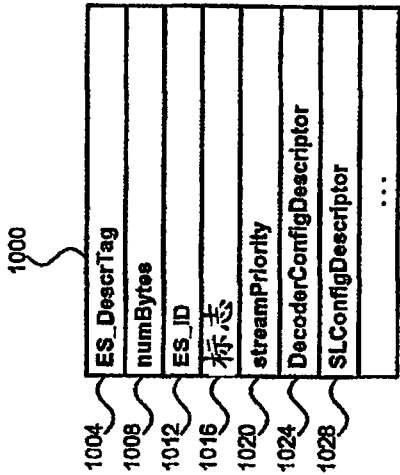


图10B

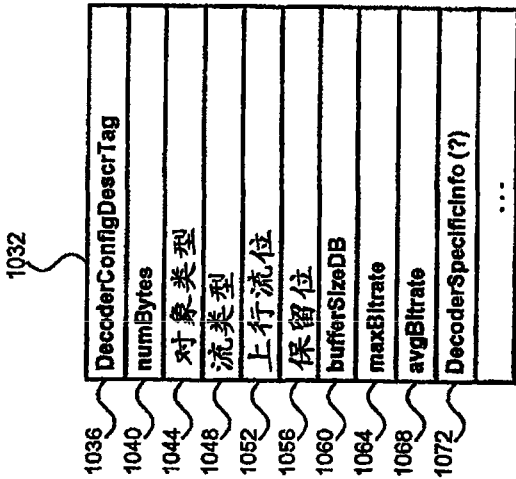


图10C

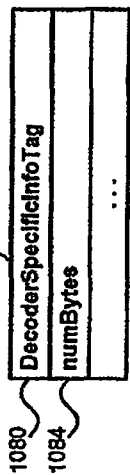
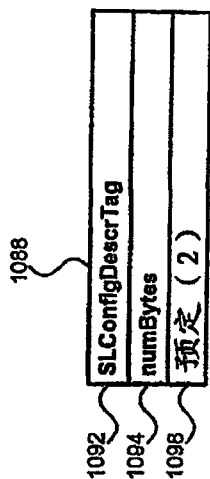


图10D



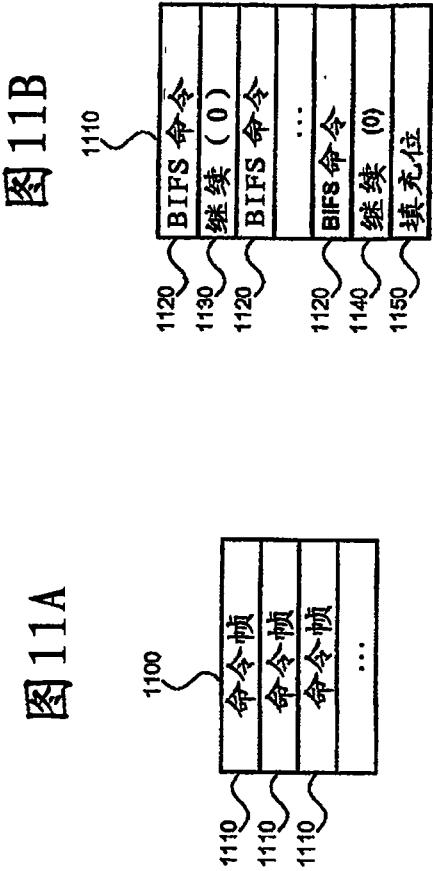


图12A

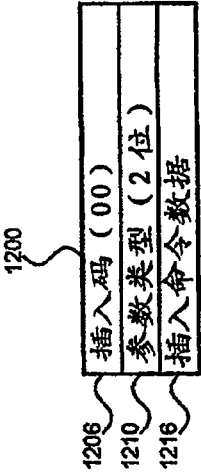


图12C

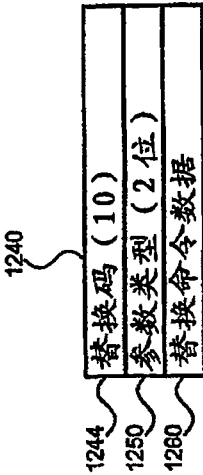


图12B

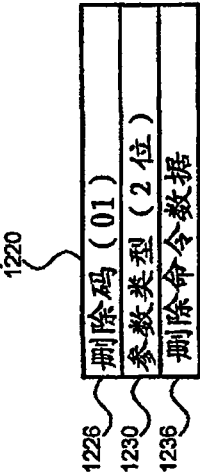


图12D



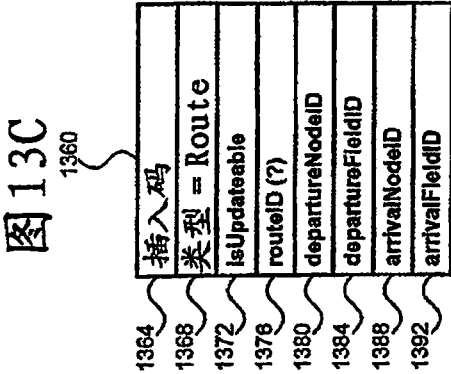
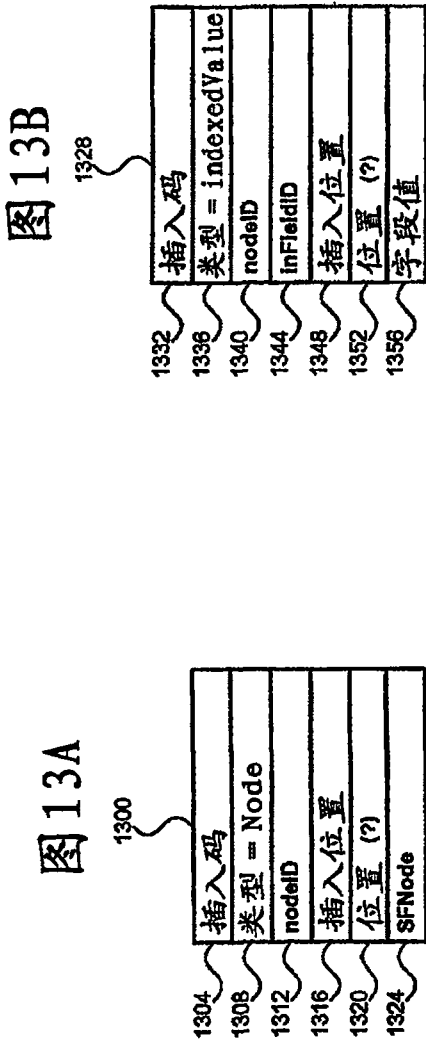


图 14A

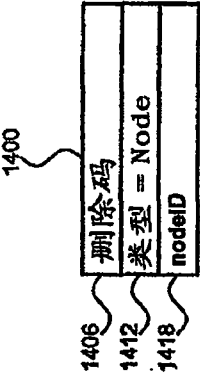


图 14B

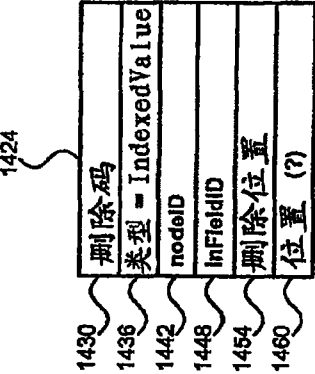
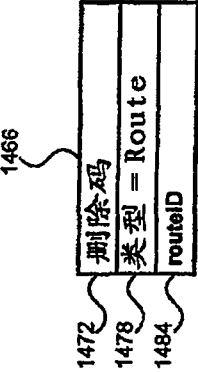
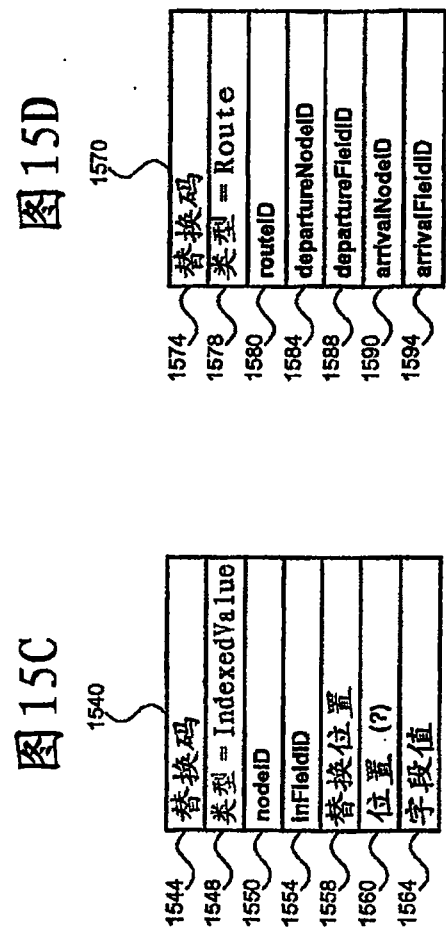
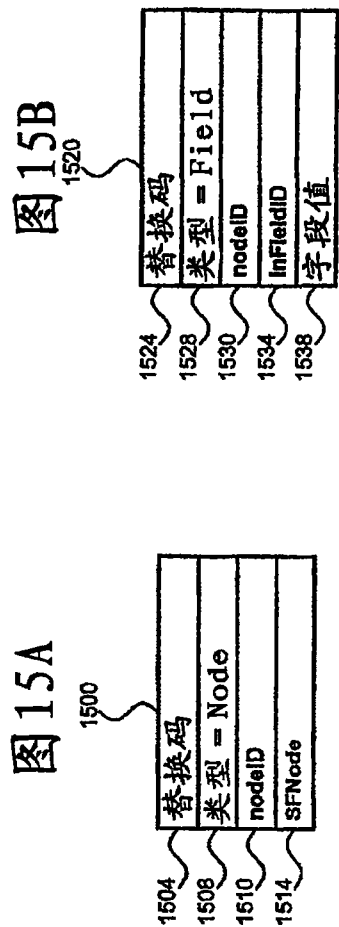


图 14C





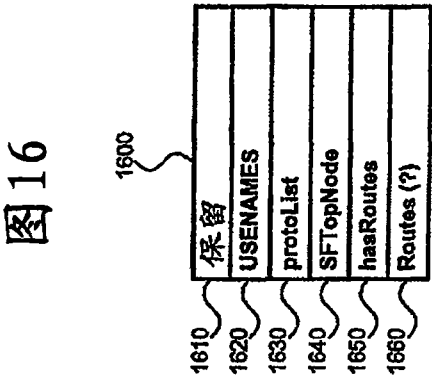


图 17A

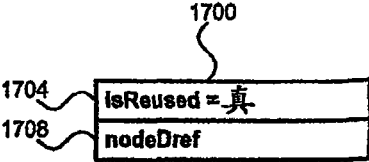


图 17B

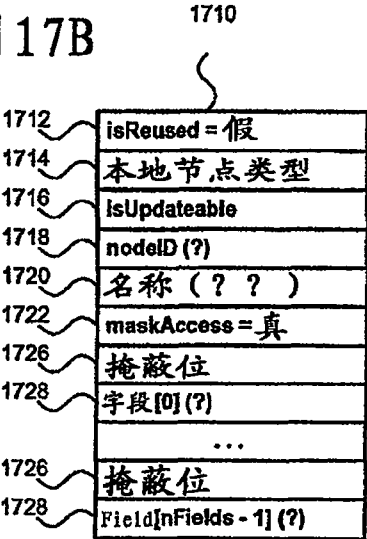


图 17C

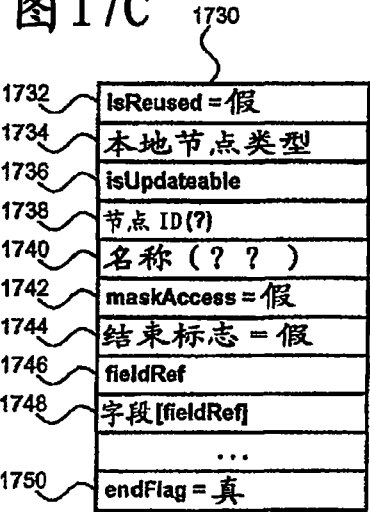


图 17D

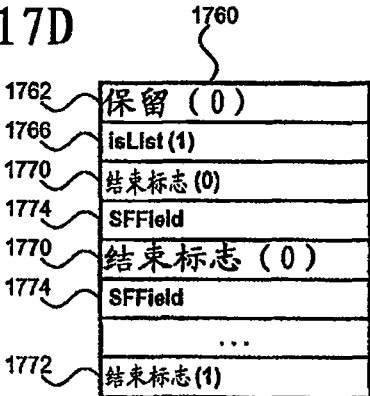


图 17E

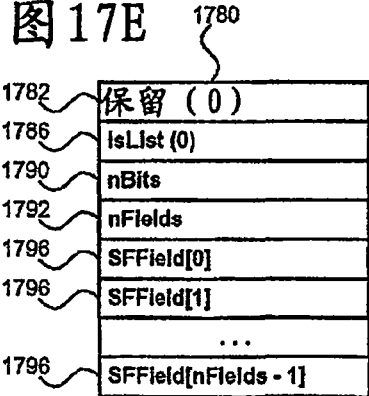


图18A

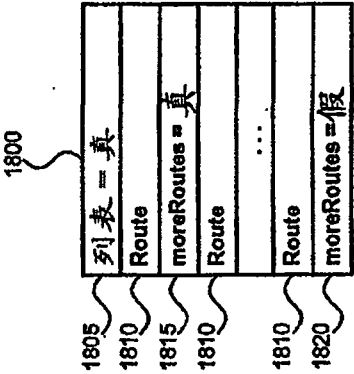


图18B

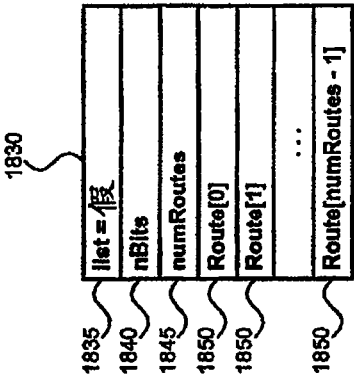


图18C

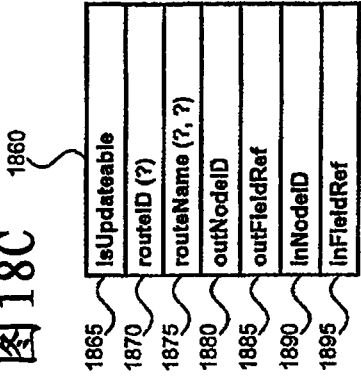


图19A

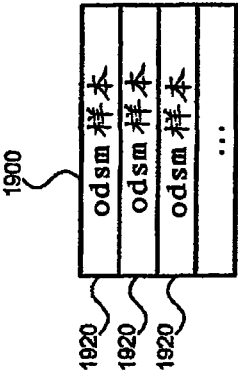


图19B

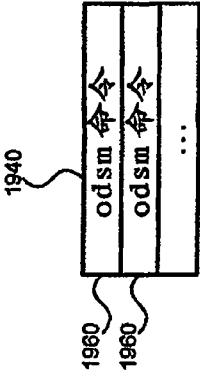


图 20A

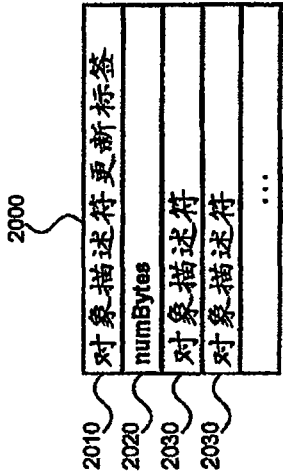


图 20B

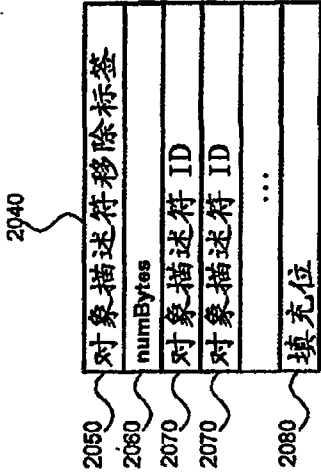


图 21A

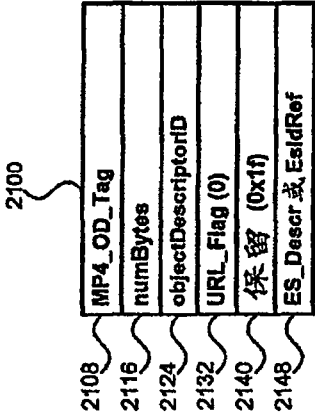


图 21B

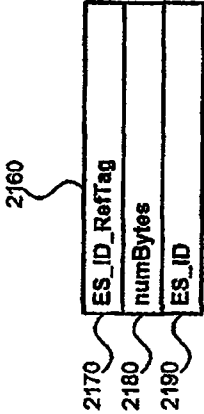


图22

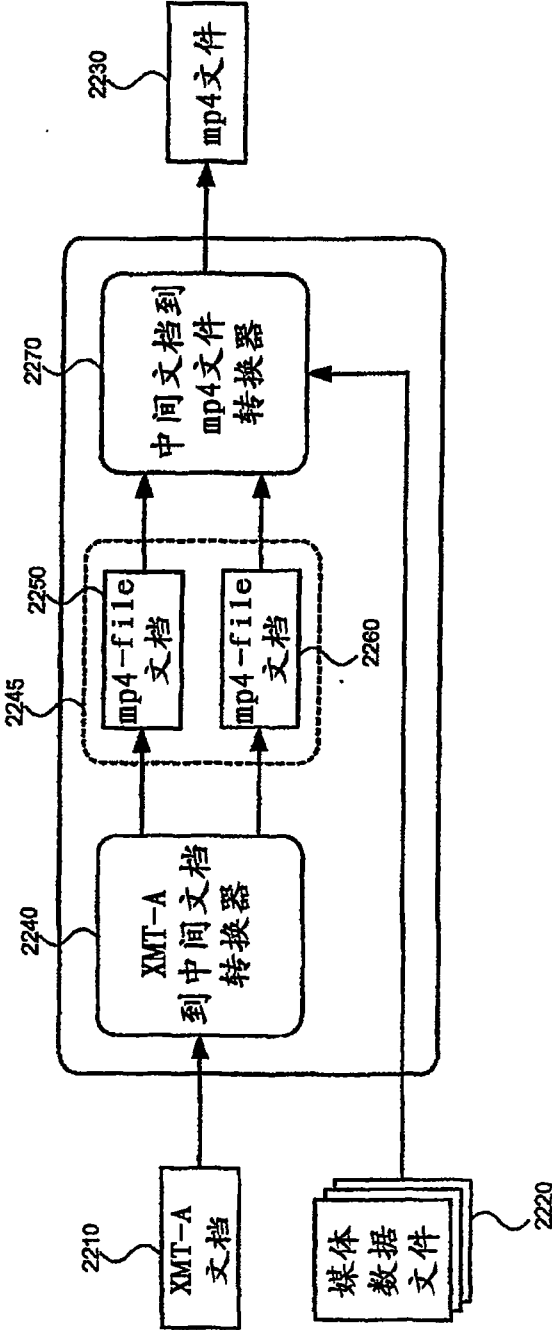


图23A

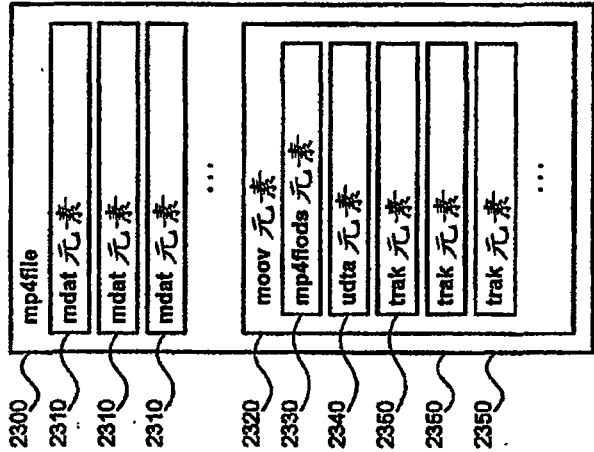


图 23B

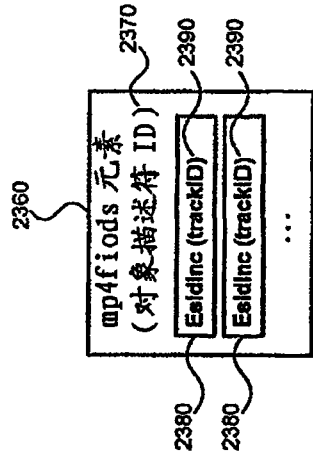


图 24A

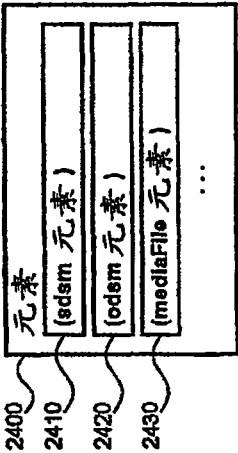


图 24B

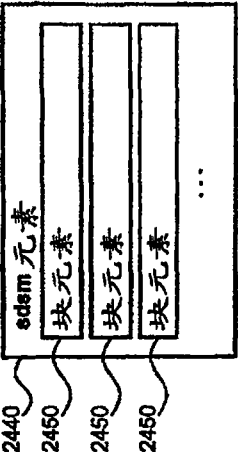


图 24C

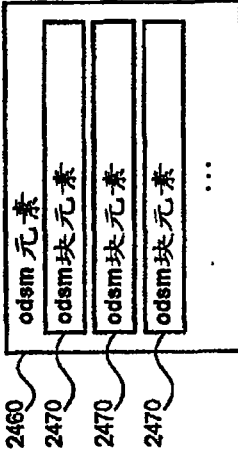


图 24D

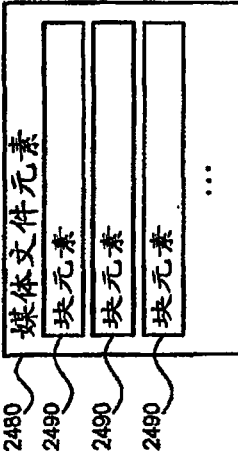


图 25A

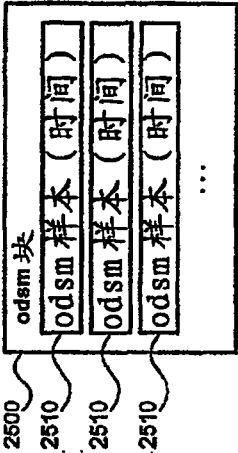


图 25B

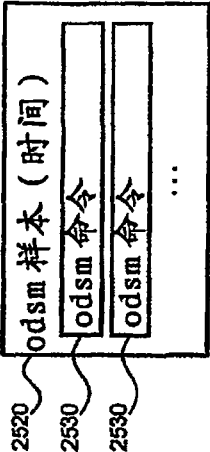


图 25C

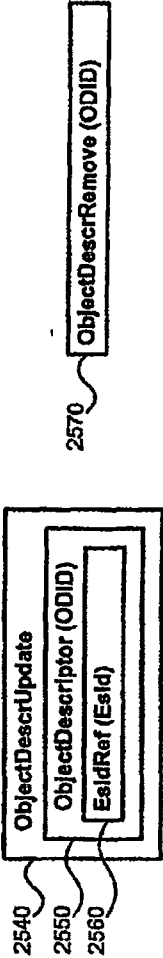


图 26A

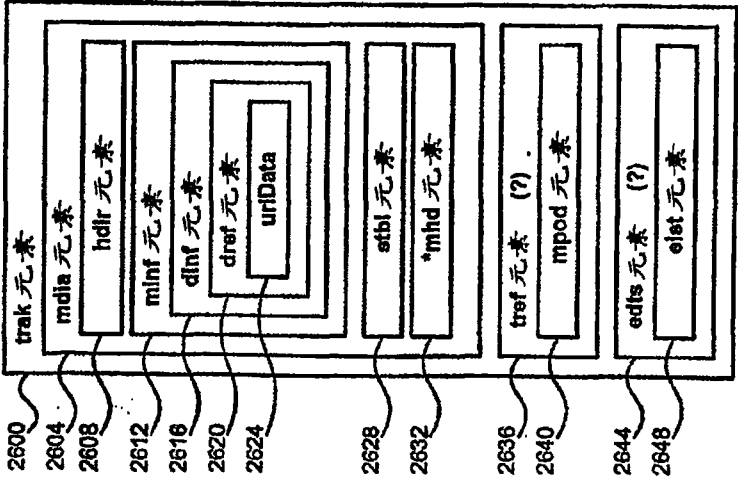


图 26B

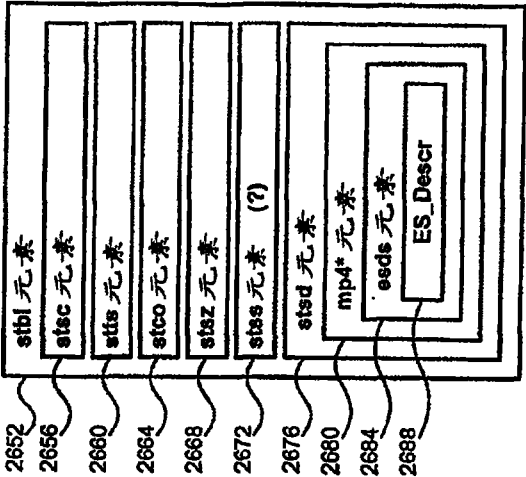
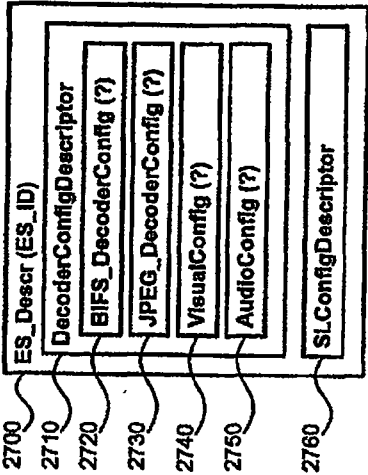


图 27



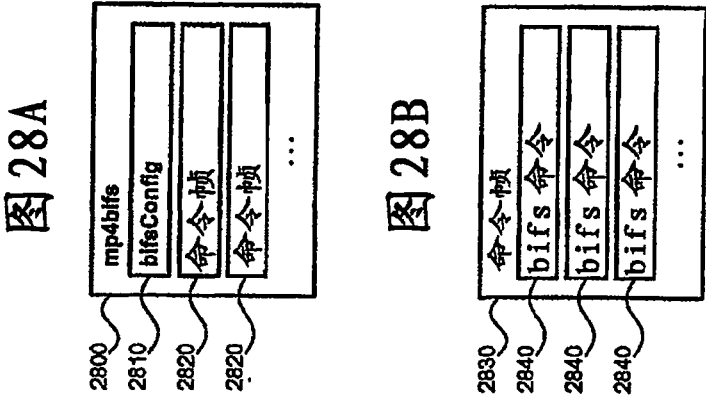


图 29A

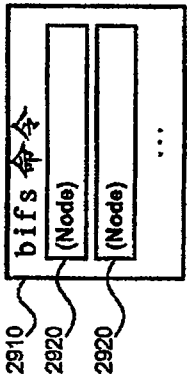
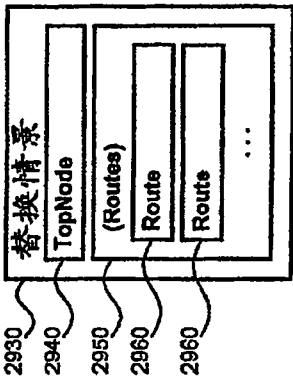


图 29B



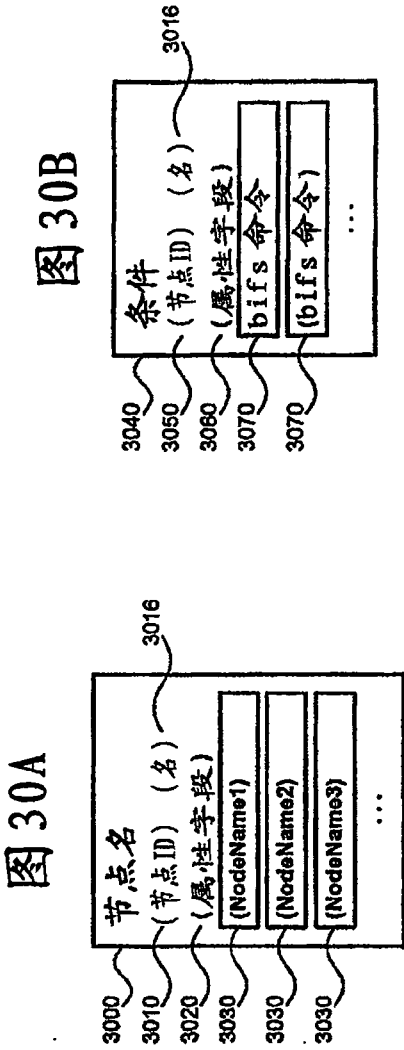


图 30B

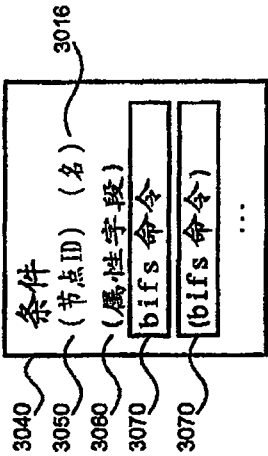


图 30C



图 31A

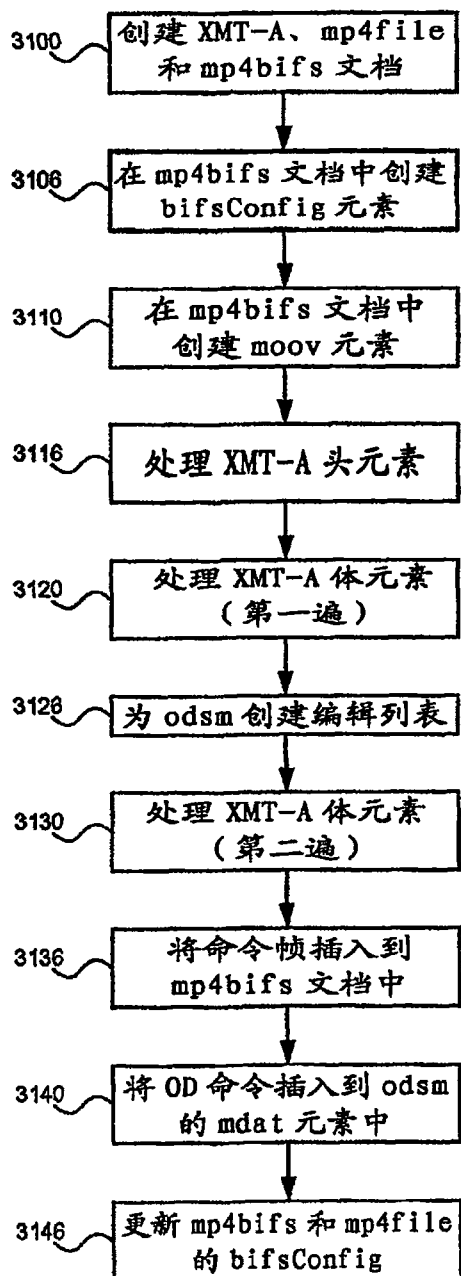


图 31B

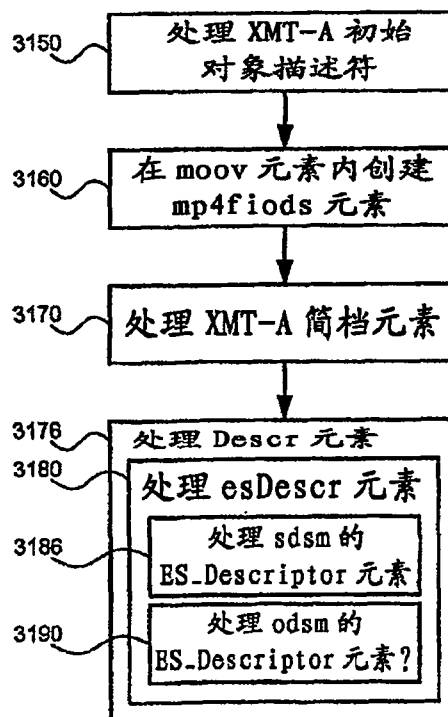


图 32

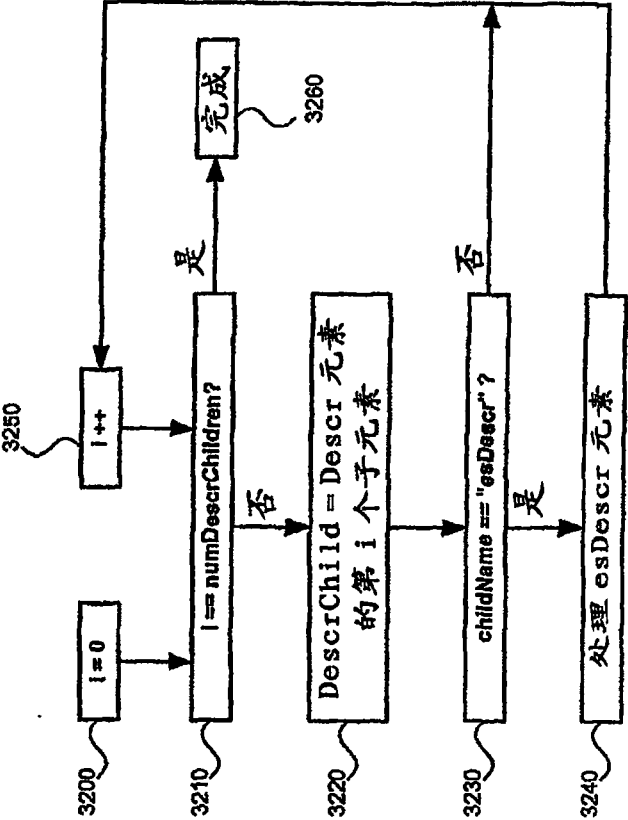


图 33

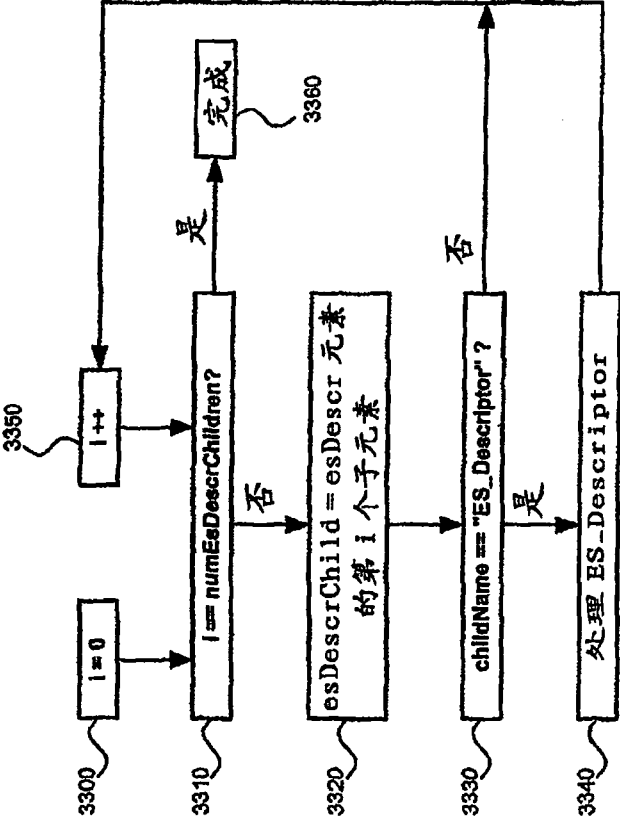


图 34

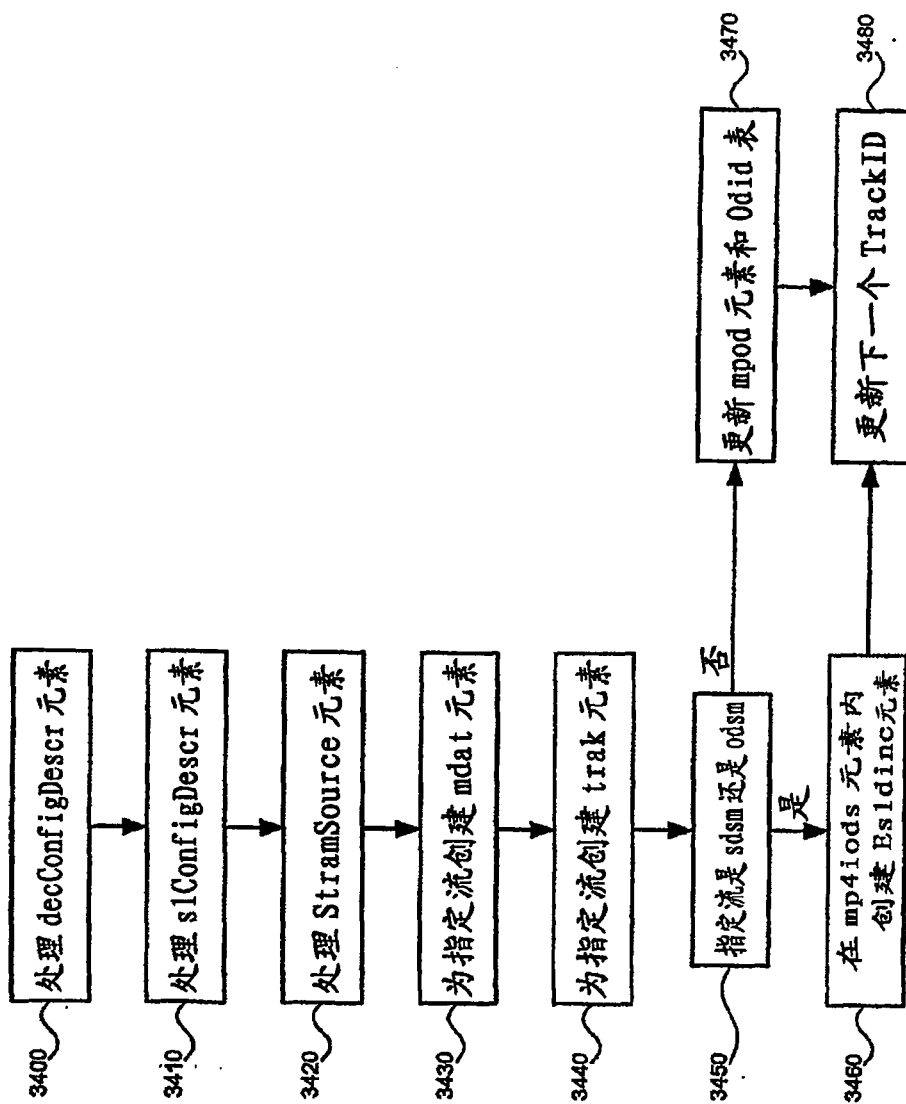


图35

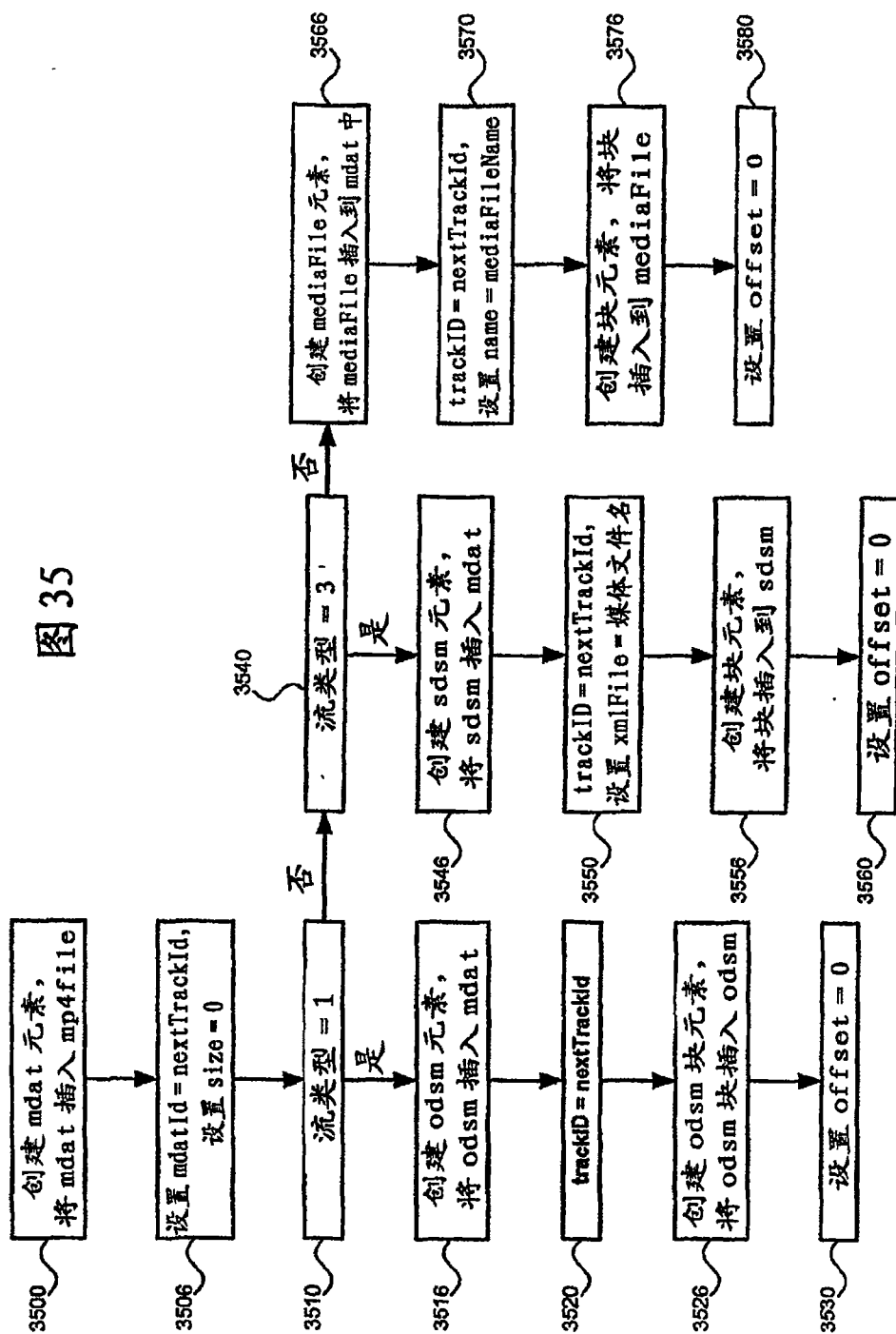


图 36A

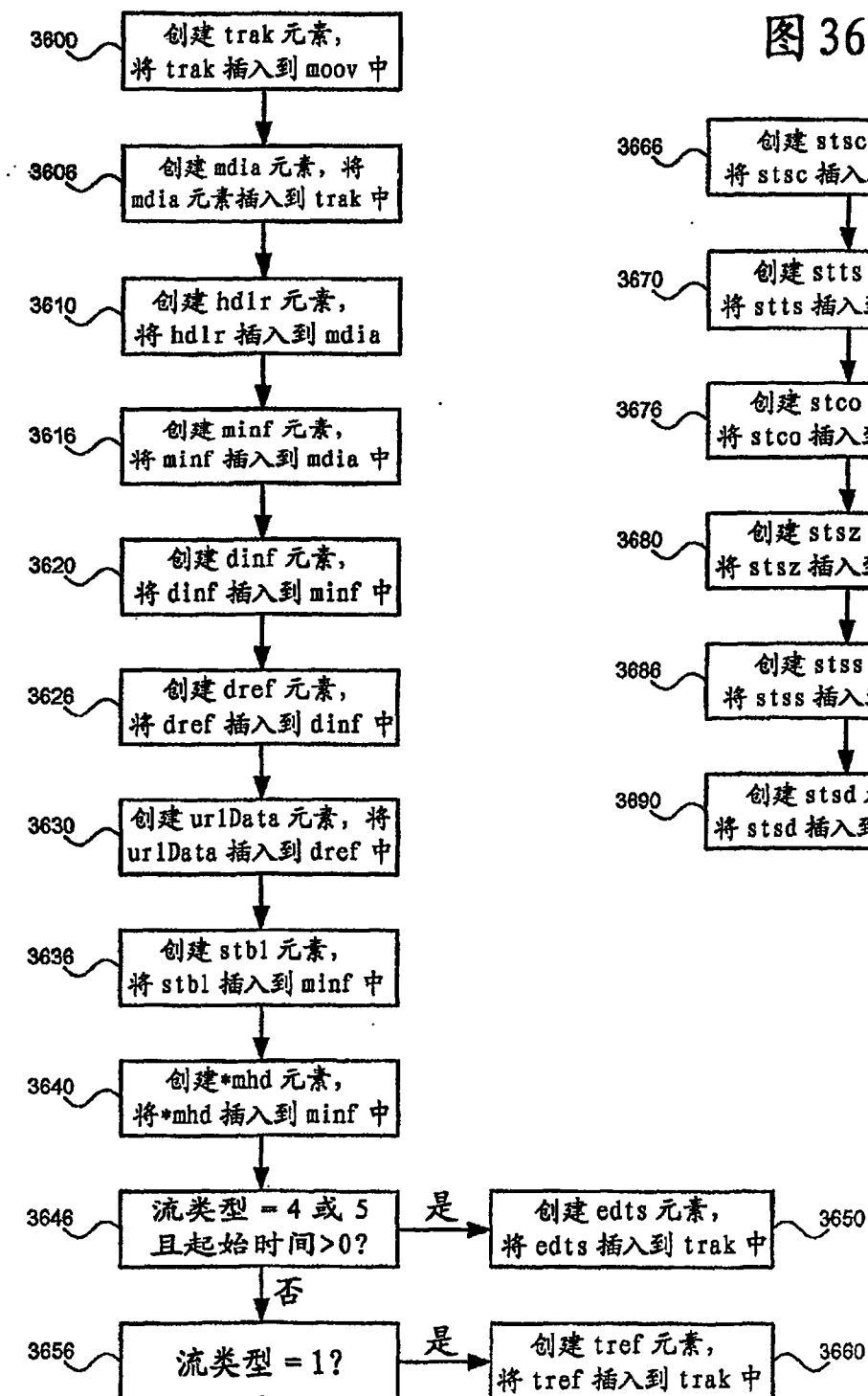


图 36B

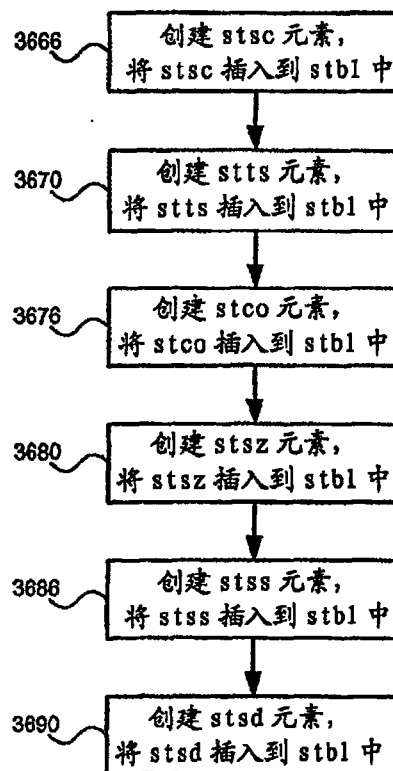


图 37

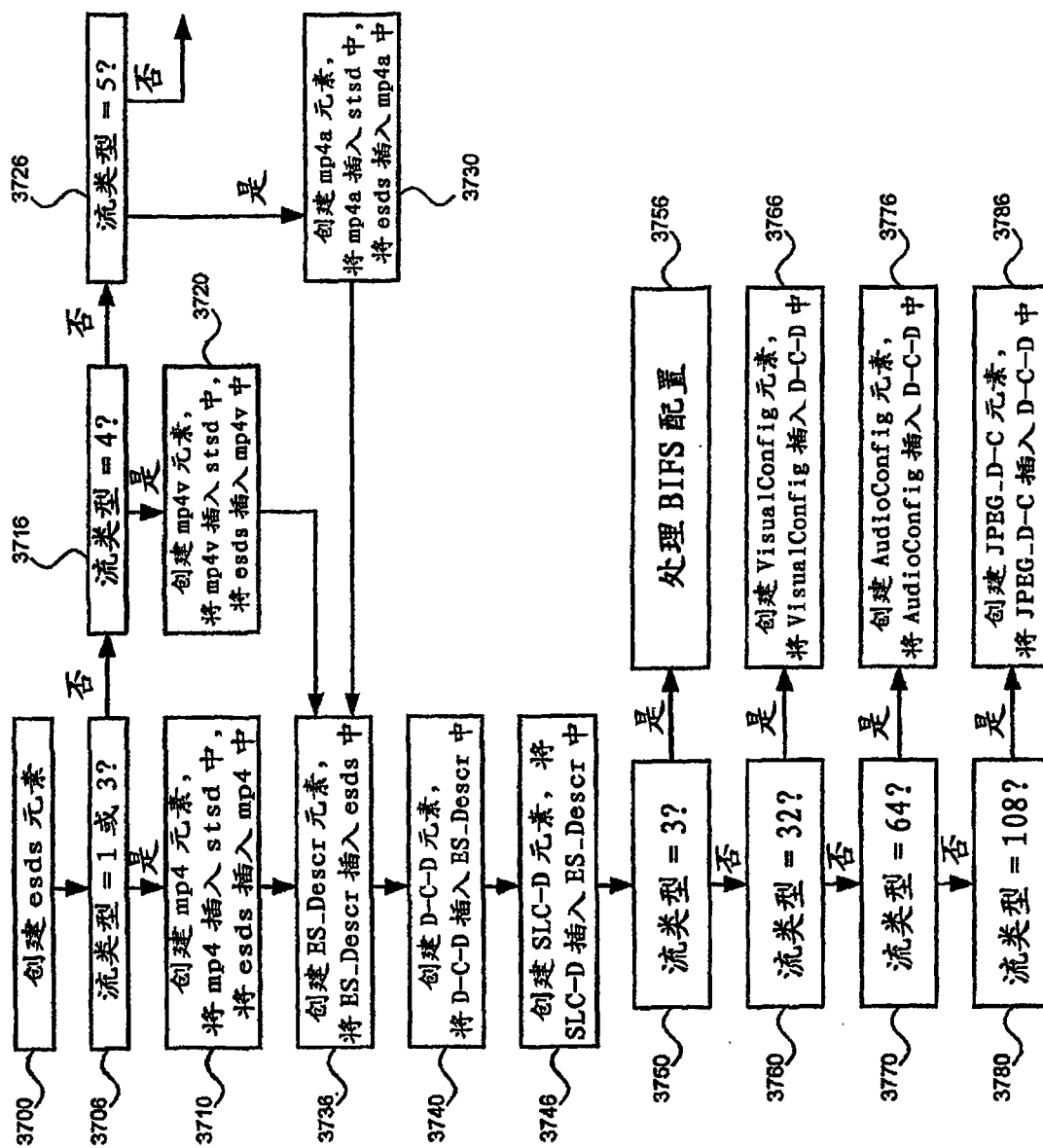


图 38

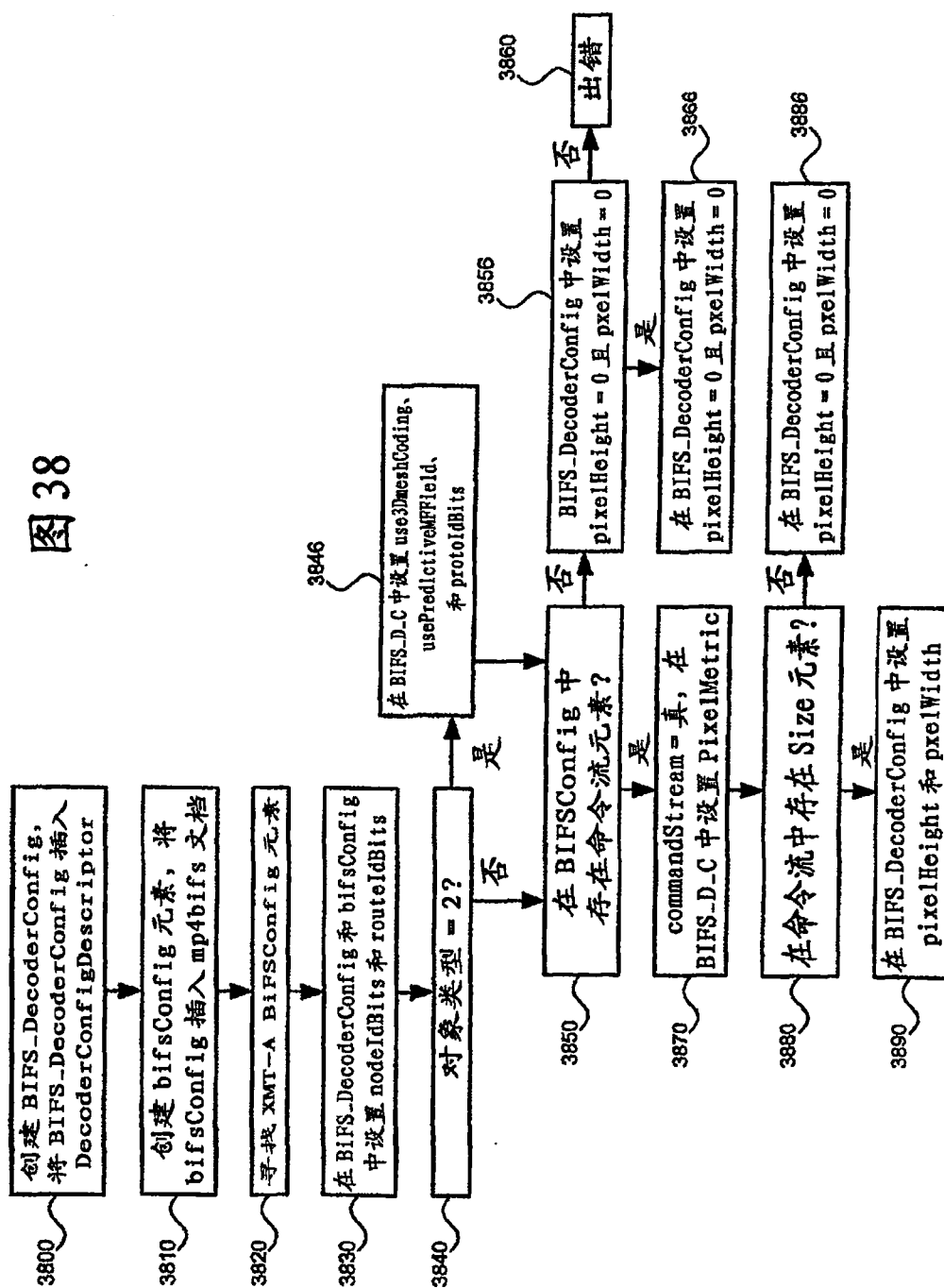


图 39A

3900	3910	3920	3930	3940	3950
位置	对象描述符 ID	OdId	开始时间	停止时间	EsId
0	OdString0	1	parTime1	parTime2	esId1
1	OdString1	2	parTime3	parTime4	esId2
...	...	...	...	...	...

图 39B

3960	3966
位置	节点串
0	NodeString0
1	NodeString1
...	...

图 39C

3970	3976
位置	路由串
0	RouteString0
1	RouteString1
...	...

图 39D

3980	3986
位置	替换情景时间
0	time0
1	time1
...	...

图 39E

3990	3992	3994	3996
项	OdId	OdId	OdId
0	OdId1	time0	真
1	OdId2	time1	真/假
...	...	...	...

图 40

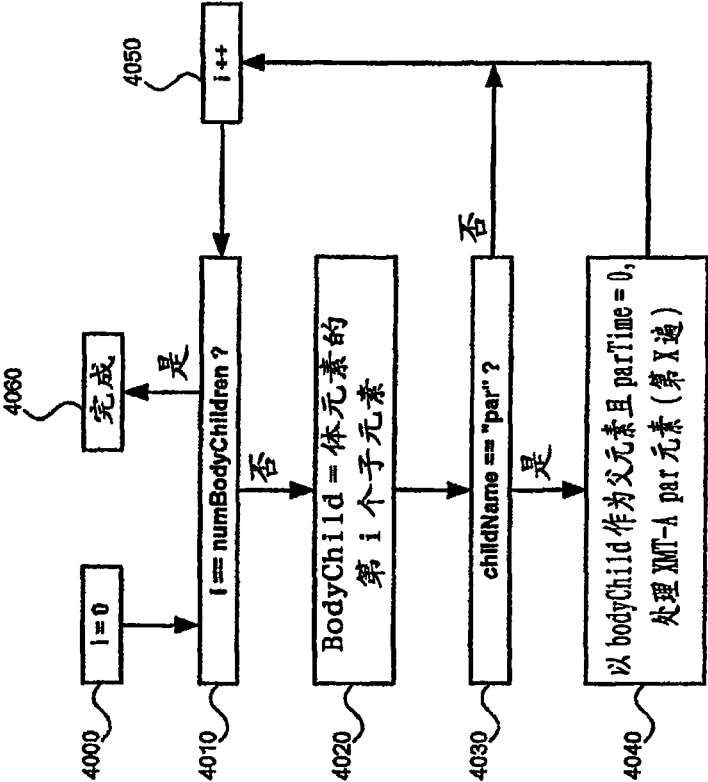


图 41

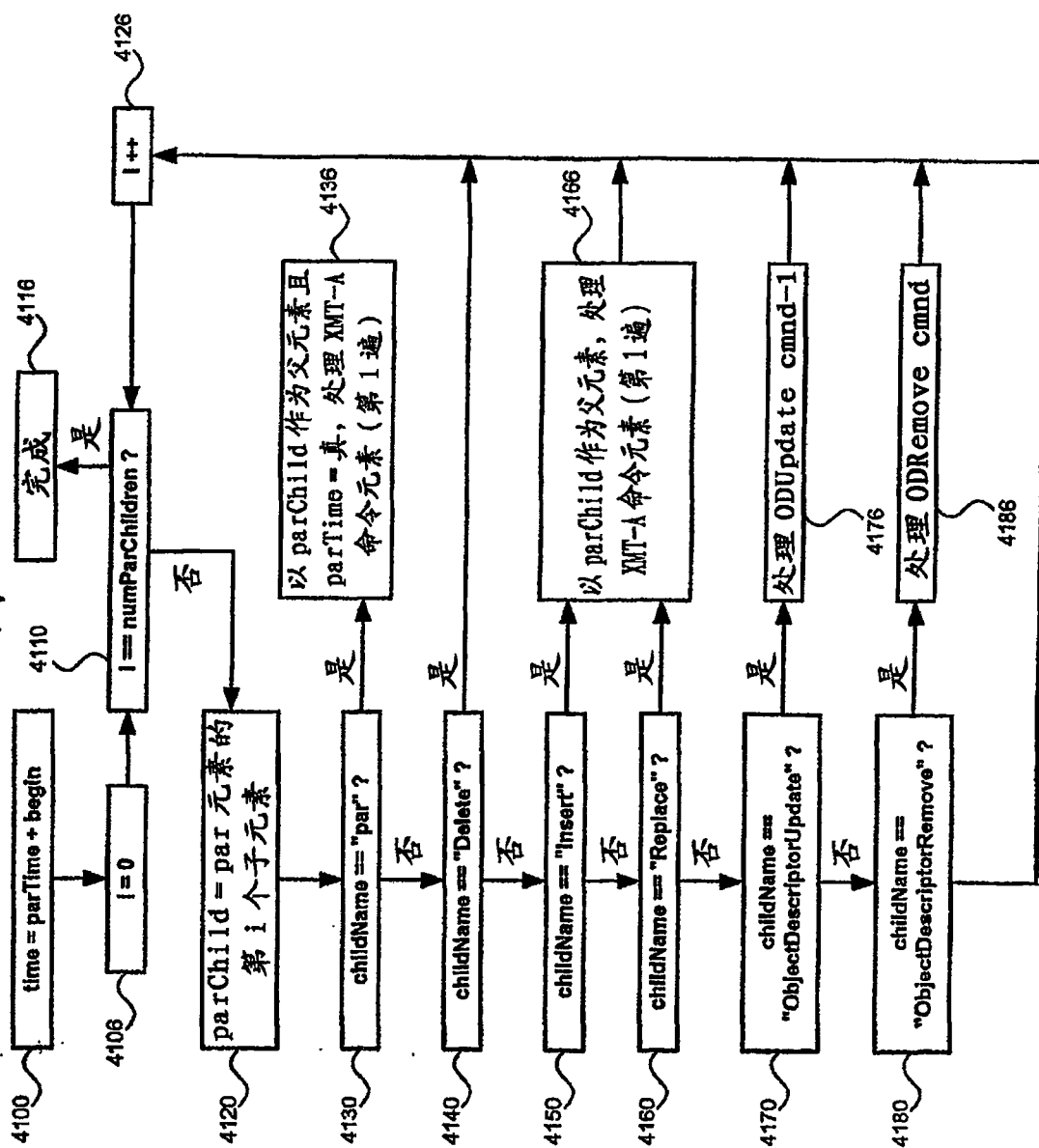


图 42

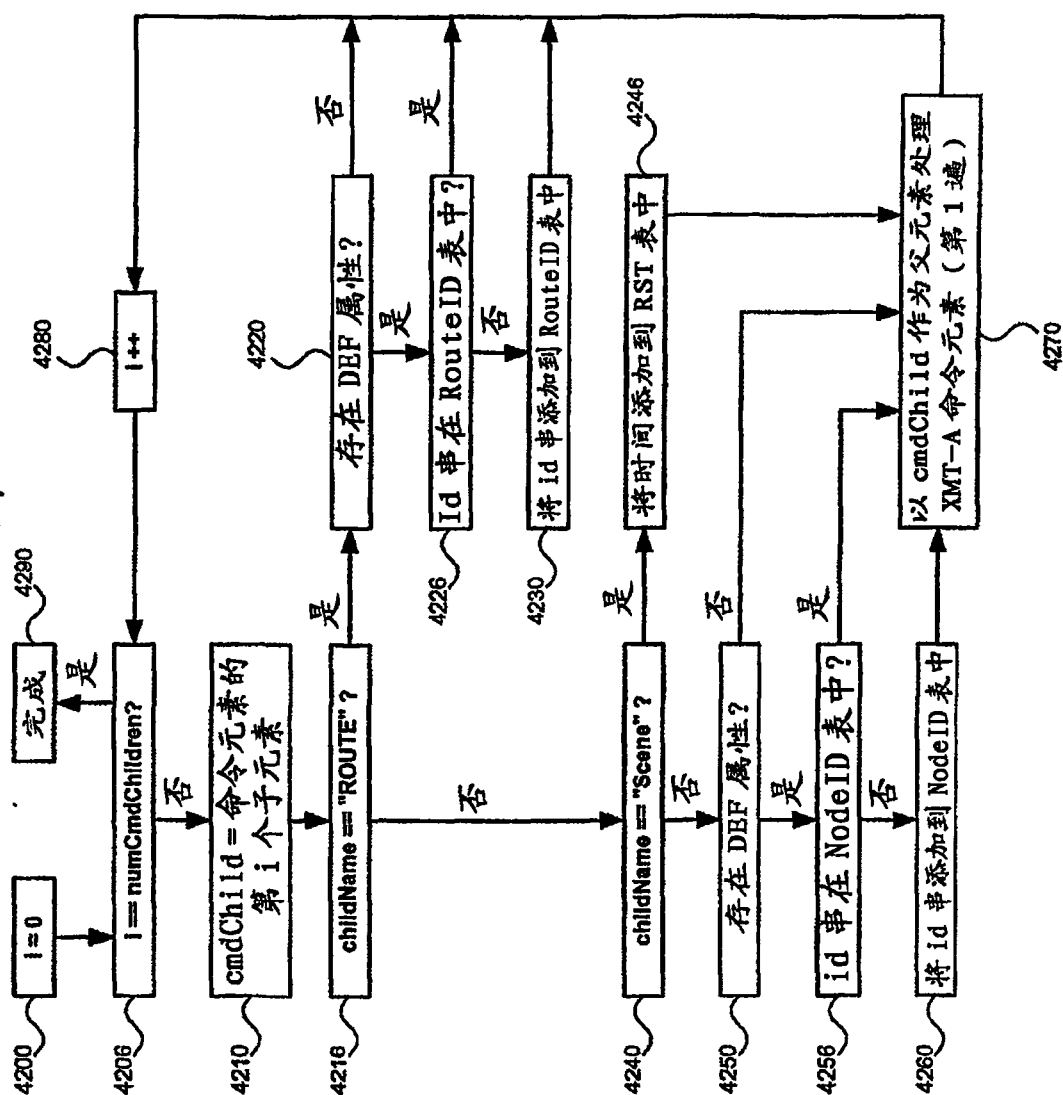


图43

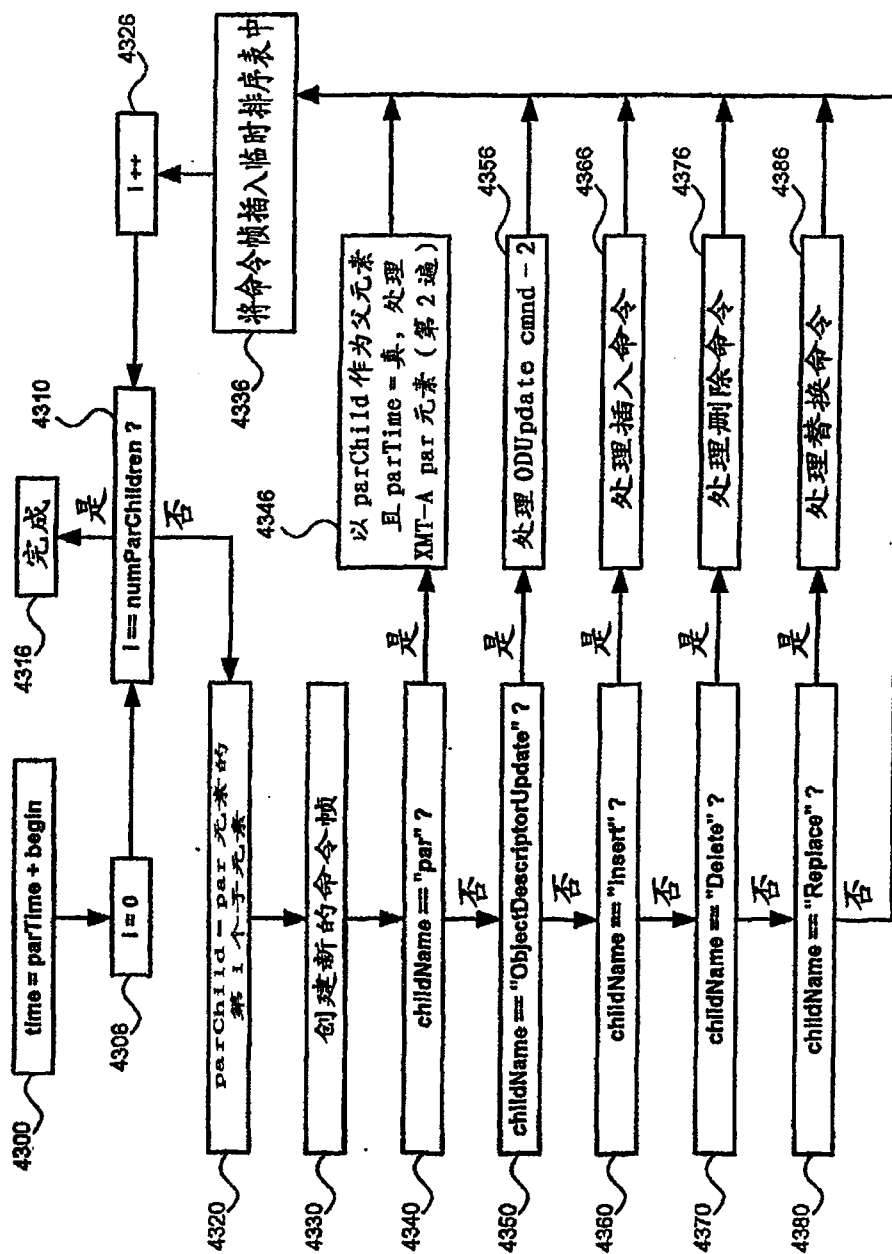


图 44

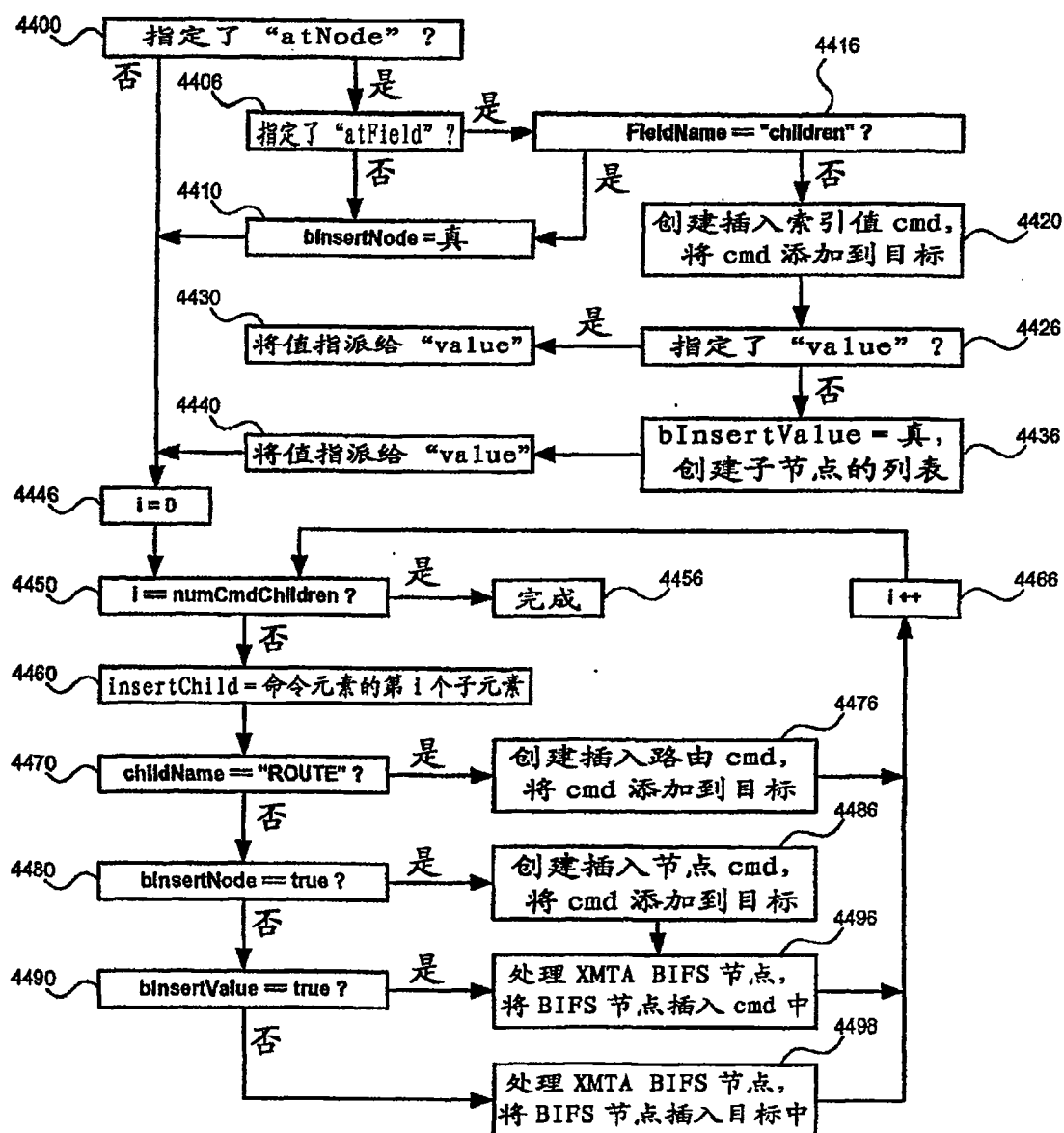


图 45

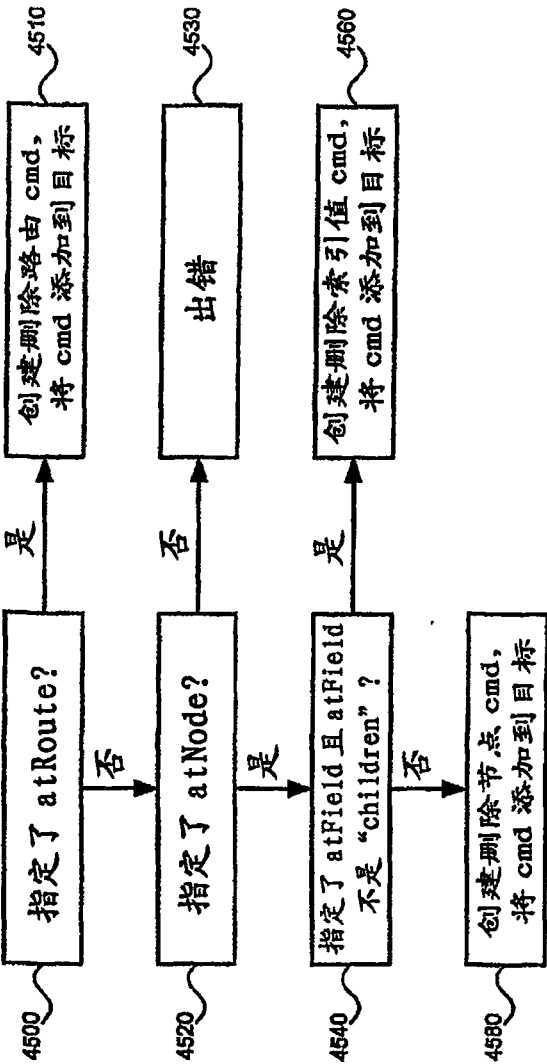


图 46

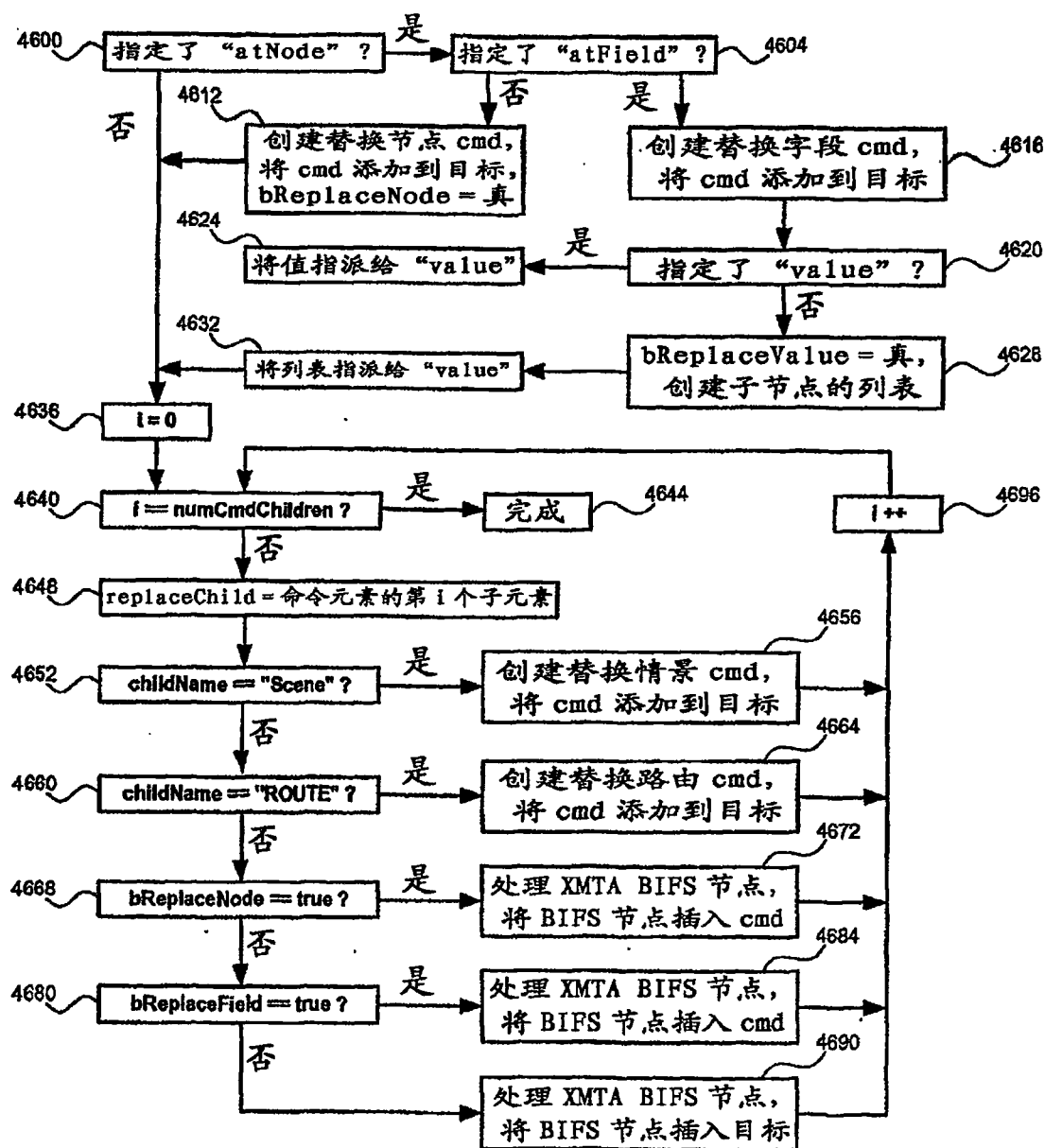


图 47

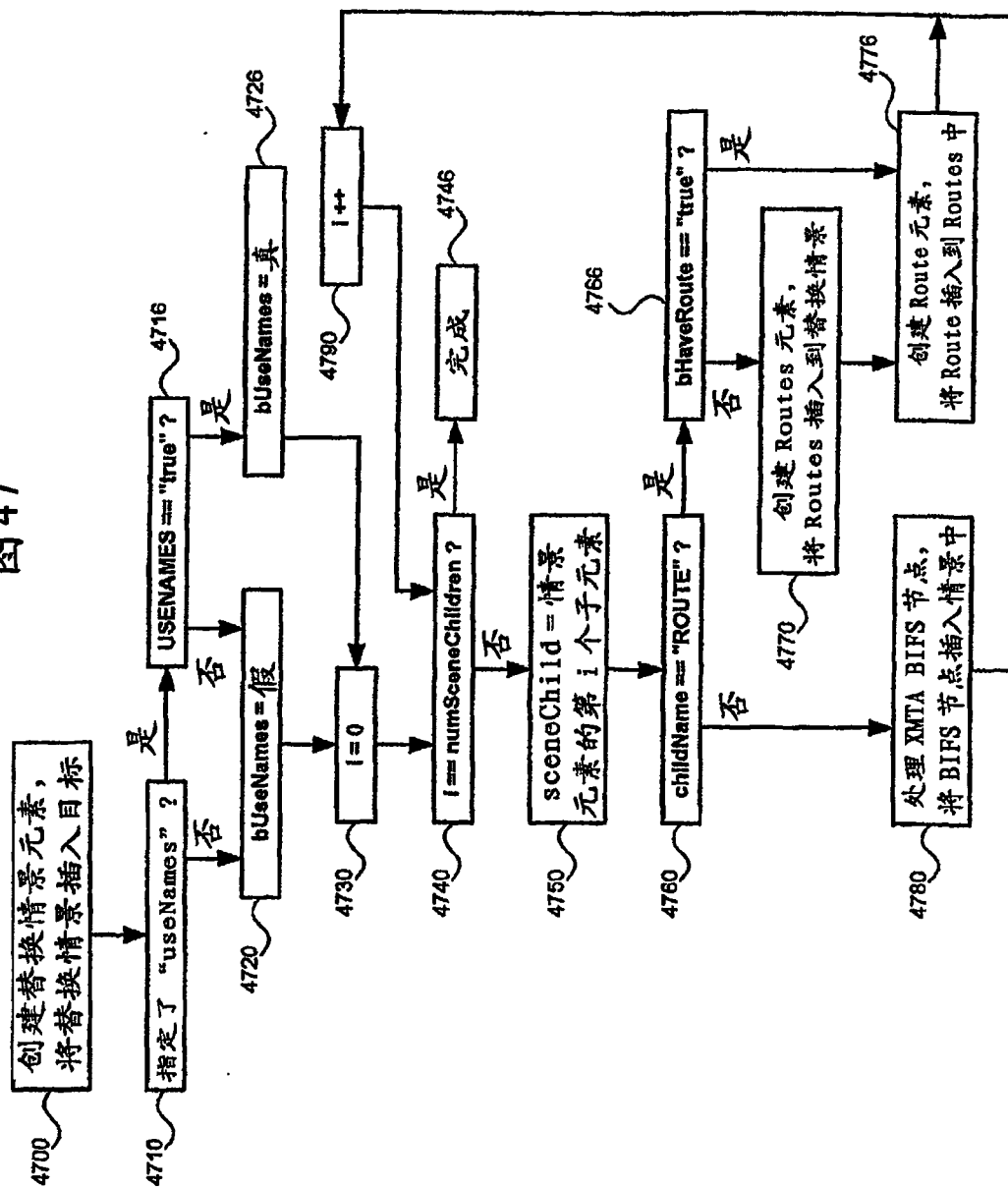


图48

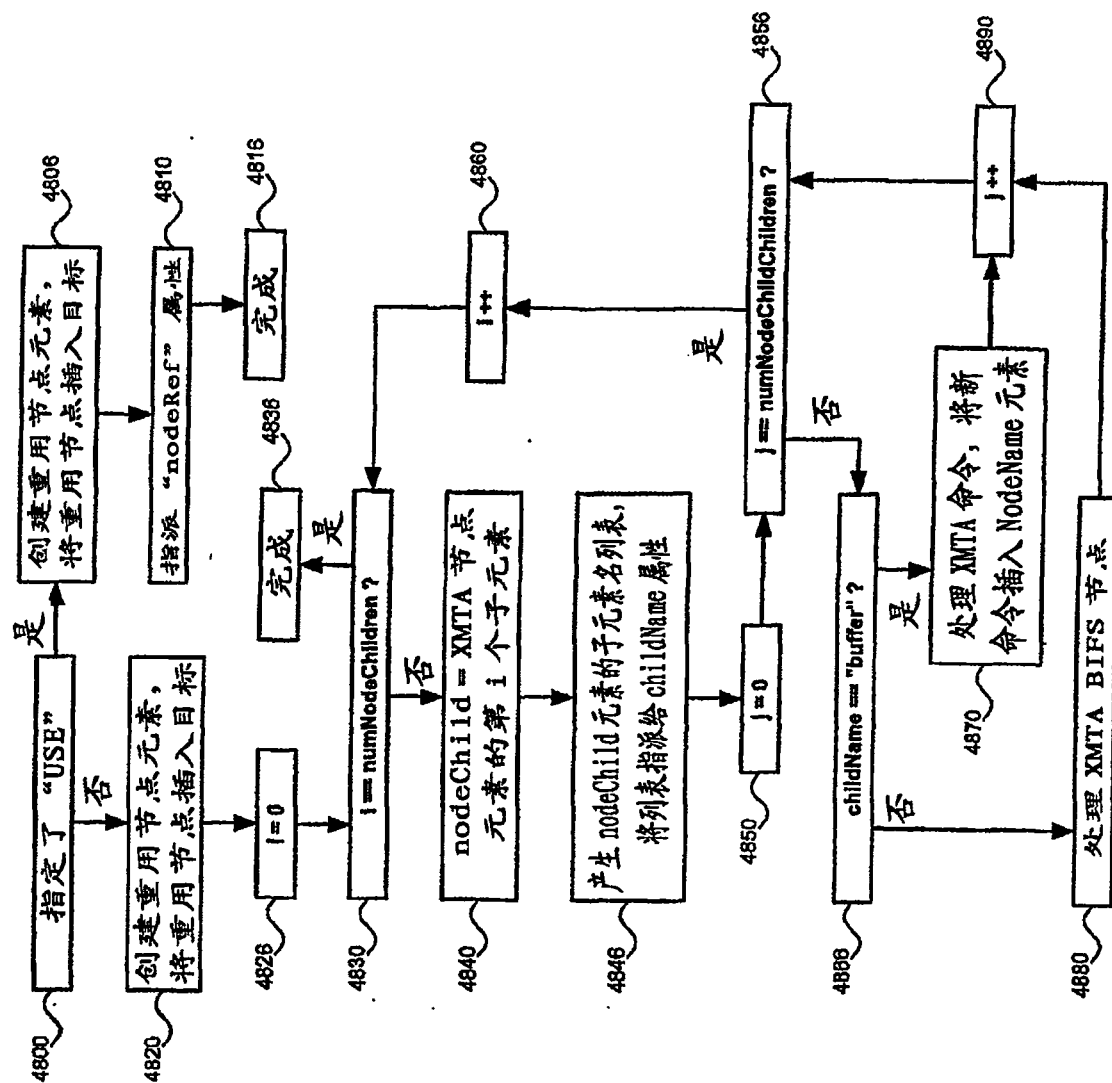


图 49

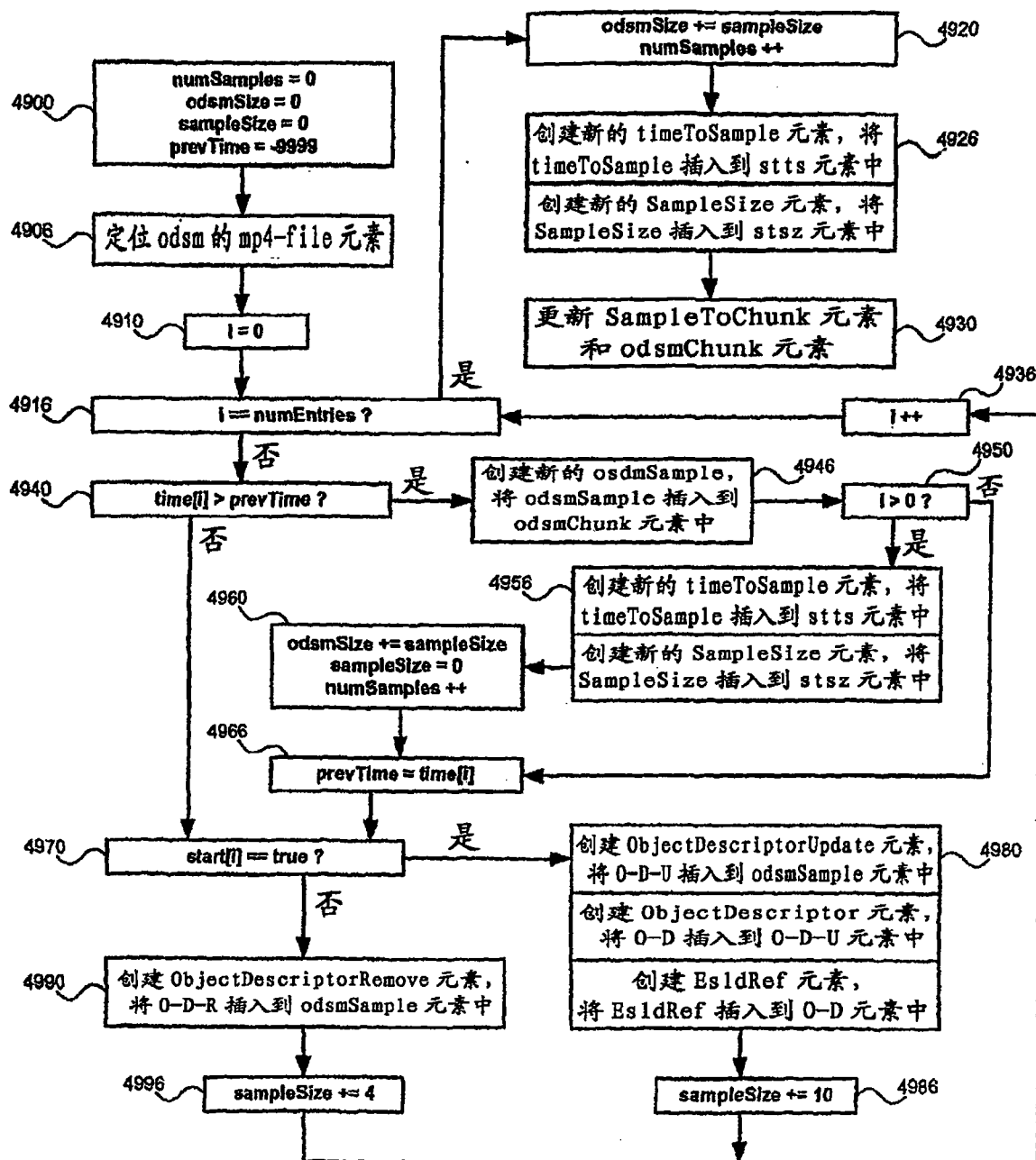


图 50

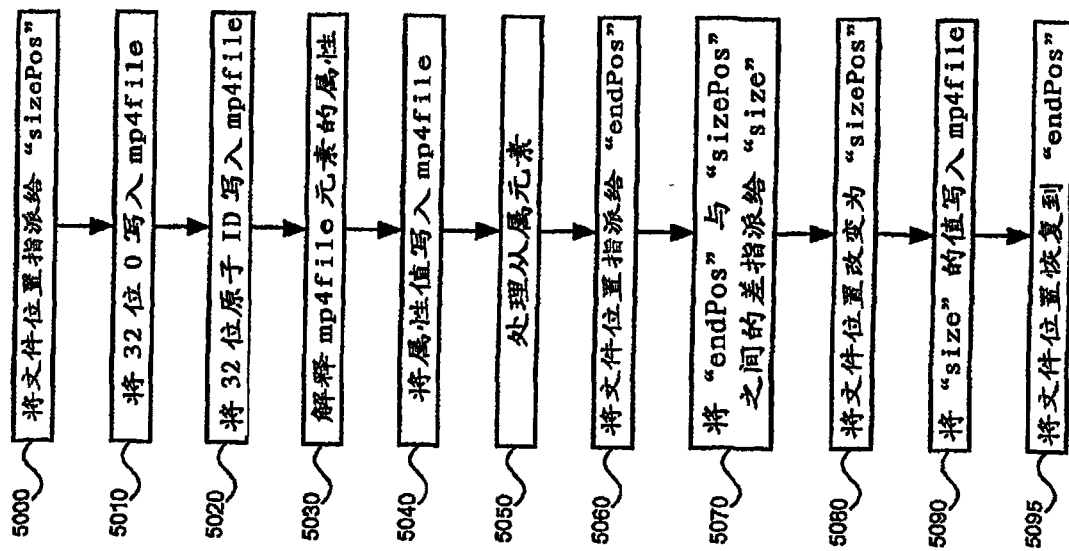


图 51

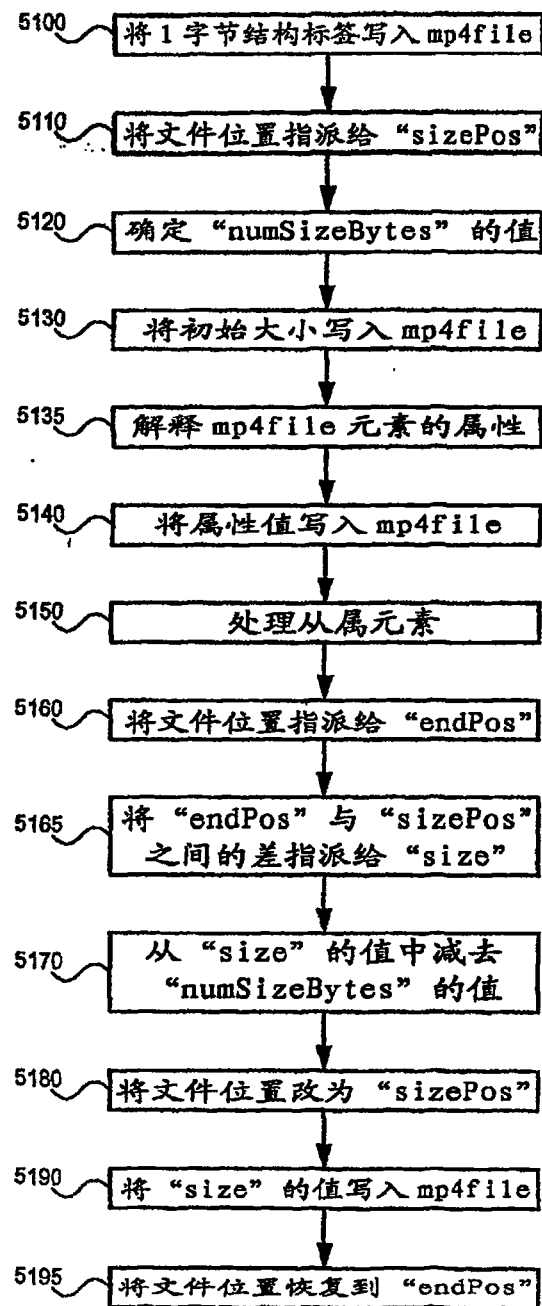


图 52

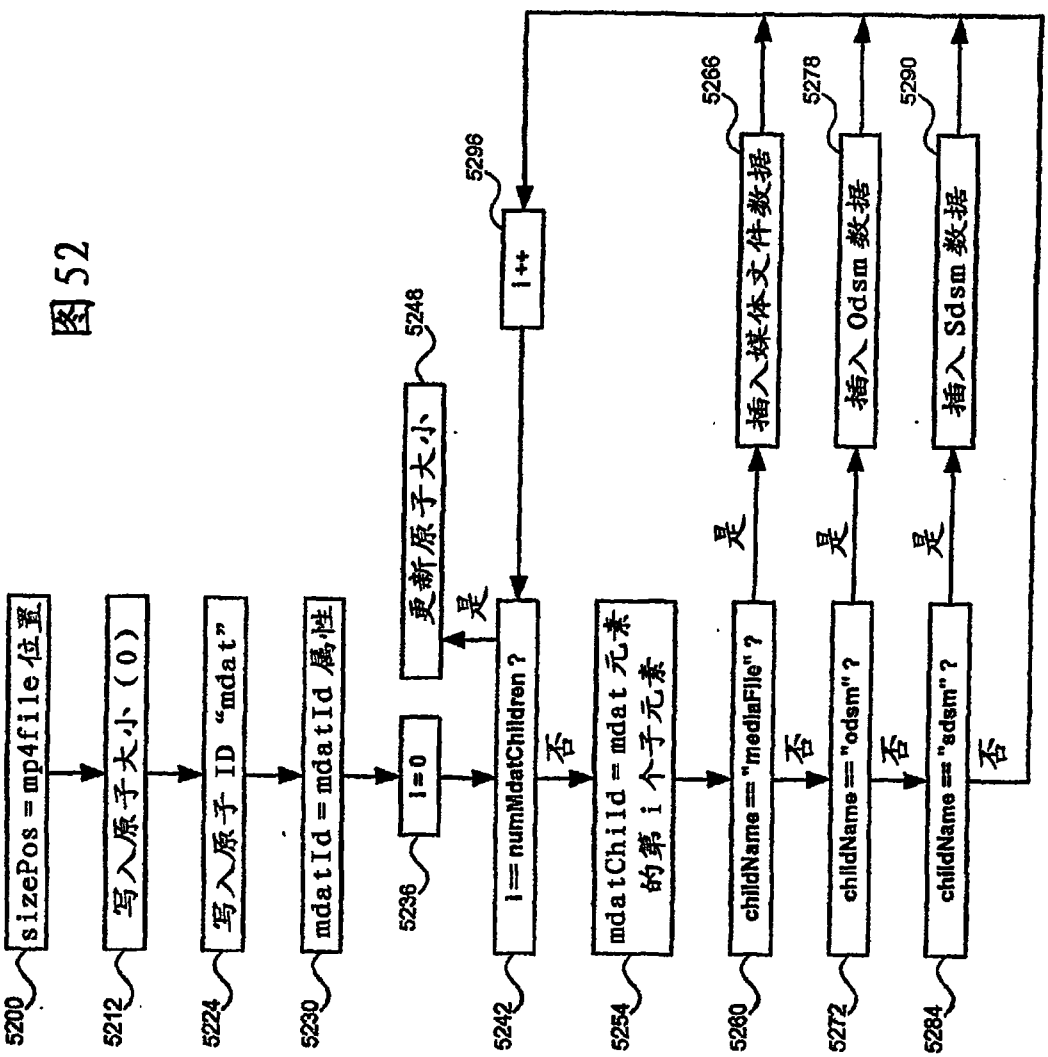


图 53

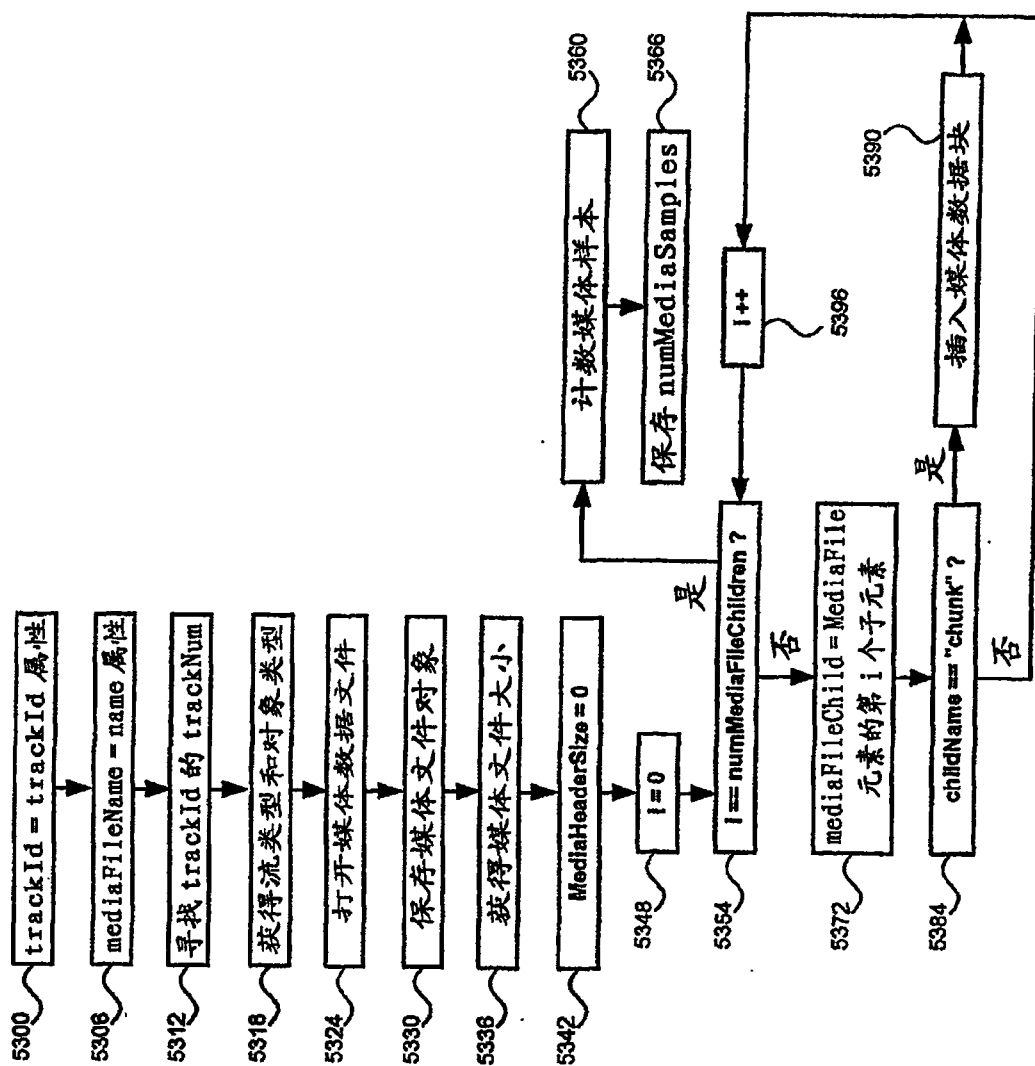


图 54

