公告本

I235611

發明專利說明書

（本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：91130400

※ 申請日期：92.10.31　　※IPC 分類：H04N7/12

壹、發明名稱：（中文/英文）

用於自動畫專家群第四階中間格式建立動畫專家群第四階本文代表之有效裝置

EFFICIENT MEANS FOR CREATING MPEG-4 TEXTUAL REPRESENTATION FROM MPEG-4 INTERMEDIA FORMAT

貳、申 請 人：（共 1 人）

姓名或名稱：（中文/英文）
美商萬國商業機器公司
INTERNATIONAL BUSINESS MACHINES CORPORATION

代表人：（中文/英文）
傑拉德 羅森賽
GERALD ROSENTHAL

住居所或營業所地址：（中文/英文）
美國紐約州阿蒙市新果園路
NEW ORCHARD ROAD, ARMONK, NY 10504, U.S.A.

國　籍：（中文/英文）
美國　　U.S.A.

參、發明人：（共 1 人）

姓　名：（中文/英文）
威廉 L. 路肯
WILLIAM L. LUKEN

住居所地址：（中文/英文）
美國紐約州約克城高地市法沃克路 2924 號
2924 FARM WALK ROAD, YORKTOWN HEIGHTS, NY 10598, USA

國　籍：（中文/英文）
美國　　U.S.A.

肆、聲明事項：

☐ 本案係符合專利法第二十條第一項☐第一款但書或☐第二款但書規定之期間，其日期為： 年 月 日。

☑ **本案申請前已向下列國家（地區）申請專利：**

1.美國； 2002 年 12 月 04 日； 10/309,571

2.

3.

4.

5.

☑ **主張國際優先權(專利法第二十四條)：**

【格式請依：受理國家（地區）；申請日；申請案號數 順序註記】

1.美國； 2002 年 12 月 04 日； 10/309,571

2.

3.

4.

5.

☐ **主張國內優先權(專利法第二十五條之一)：**

【格式請依：申請日；申請案號數 順序註記】

1.

2.

☐ **主張專利法第二十六條微生物：**

☐ 國內微生物 【格式請依：寄存機構；日期；號碼 順序註記】

☐ 國外微生物 【格式請依：寄存國名；機構；日期；號碼 順序註記】

☐ 熟習該項技術者易於獲得，不須寄存。

玖、發明說明：

【發明所屬之技術領域】

本發明一般係關於多媒體資訊的資料代表，更明確而言，關於一種稱為「動畫專家群第四階中間格式」的多媒體資訊代表形式，向另一種稱為「動畫專家群第四階本文代表」的多媒體資訊代表形式的變換。

【先前技術】

電腦常用於呈現各種數位媒體，包括影像、聲頻樣本(聲音)及視頻媒體，以及本文與幾何形狀。各類媒體可個別呈現，或數個此類媒體元件可一起呈現，其方式稱之為合成多媒體表現。

建立及分配合成多媒體表現的能力，對基於各類媒體的資訊配送非常重要。此外，已創造出代表合成多媒體表現的標準化裝置，使許多作者能夠創建可在各種電腦平臺，如個人電腦、視訊盒及其他元件上再現的表現形式。

由動畫專家群(Motion Pictures Experts Group；MPEG)開發的兩種熟知的標準化格式，係可擴充動畫專家群第四階本文(Extensible MPEG-4 Textual；XMT)格式及二進制編碼的動畫專家群第四階(mp4)格式。XMT格式非常適合製作合成多媒體表現，而mp4格式非常適合壓縮儲存與傳輸合成多媒體表現。因此，需要將XMT格式之表現有效轉換為mp4格式之表現。

【發明內容】

如下所詳述，本發明係一種用於將一二進制編碼的動畫

專家群第四階(mp4)格式，轉換為一可擴充的動畫專家群第四階本文(XMT)格式之方法、系統及設備。本發明利用極少量軟體組成的高效設施，只需適度資源，即可實現合成多媒體表現自mp4格式向XMT格式的轉換。

因此，本發明的一方面包括一用於將一動畫專家群第四階(mp4)二進制檔案，轉換為一可擴充動畫專家群第四階本文(XMT)檔案之方法。該方法包括一產生操作，產生一組代表該mp4二進制檔案的資料結構。一第一翻譯操作，將該組資料結構翻譯為至少一個中間結構文件，一第二翻譯操作將該中間結構文件翻譯為一XMT結構文件。然後，一建立操作，根據該XMT結構文件建立XMT檔案。

本發明的另一方面為一用於將一動畫專家群第四階(mp4)二進制檔案，轉換為一可擴充動畫專家群第四階本文(XMT)檔案之系統。該系統包括一二進制檔案解碼器，其配置成用以輸入該mp4二進制檔案，並產生代表該mp4二進制檔案的一組資料結構。一第一翻譯器，其係配置成用以輸入該組資料結構，並建立至少一個中間結構文件。一第二翻譯器，其係配置成用以輸入該中間結構文件，並產生一XMT結構文件。一XMT串聯器，其配置成用以輸入該XMT結構文件，並產生該XMT檔案。

本發明的另一方面係在一有形媒體中具體化的一電腦程式產品，用於將一動畫專家群第四階(mp4)二進制檔案，轉換為一可擴充動畫專家群第四階本文(XMT)檔案。該電腦程式執行以下操作：產生一組代表該mp4二進制檔案的資料

結構；將該組資料結構翻譯為至少一個中間結構文件；將該中間結構文件翻譯為一XMT結構文件；及根據該XMT結構文件建立該XMT檔案。

自下述參考附圖對各種具體實施例的特定說明，更可理解本發明的前述及其他功能、效用及優點。

標題表

動畫專家群第四階本文代表	1.0
動畫專家群第四階中間格式檔案	2.0
場景描述資料流(sdsm)	3.0
物件描述符資料流(odsm)	4.0
mp4-file文件	5.0
mp4-bifs文件	6.0
mp4至xmta轉換器	7.0

【實施方式】

本發明係一種用於將一可擴充動畫專家群第四階本文(XMT)格式(本文也稱為XMT-A文件及動畫專家群第四階本文代表)，轉換為一二進制編碼的動畫專家群第四階(mp4)格式(也稱為動畫專家群第四階中間二進制格式)之方法、系統及電腦程式。本發明用一新穎的方式來實現自XMT-A向mp4的轉換，只需極少量的軟體及適度的資源。以下參考圖1至65說明本發明。

1.0 動畫專家群第四階本文代表

動畫專家群第四階本文代表由代表一多媒體表現之結構的一「本文檔案」組成。一多媒體表現由聲音、靜態影像、

視頻剪輯及其他元件的同步組合或序列組成。一本文檔案係字母、數字及標點的二進制碼序列所組成的一電子資料結構。本文檔案一般可用通常稱為「文字編輯器」的軟體進行解譯。文字編輯器有許多實例，包括稱為「NotePad.exe」的軟體(對基於Windows (r)作業系統的電腦)，及「vi」(對使用各種統稱為UNIX之作業系統的電腦)。Windows係微軟公司的註冊商標，該公司位於華盛頓州的雷德蒙德市(Redmond, Washington)。包含動畫專家群第四階本文代表的本文檔案類型稱為「XMT-A」檔案。

在本文檔案的範圍內，XMT-A檔案係一可擴充標示語言(XML)檔案的範例。XML檔案為一結構文件，其根據全球資訊聯盟所規定的原則(請參閱http://www.w3.org/TR/2000/REC-XML-20001006)建立。XMT-A檔案代表XML檔案的特定具體實施例，如國際標準組織與國際電子技術委員會所規定(請參閱ISO/IEC文件14496-1: 2000 Amd. 2, 2000年10月，可自http://mpeg.telecomitalialab.com/working_documents.htm及國際標準化組織(International Organization for Standardization；ISO)獲取，其地址為瑞士CH-1211日內瓦20，Case postal 56，rue de Varembé)。關於XMT-A規格各部分的完整說明都將是極為冗長的。因而，下面有關XMT-A檔案的說明只限於說明本發明所需的規格部分。對於XMT-A檔案結構的完整說明，讀者應查看所引用的XMT規格說明文件。

與任何XML檔案一樣，一XMT-A檔案由一組階層式「元

件」組成。各元件可包含稱為子元件的附屬元件。此外，各元件可擁有一組稱為「屬性」的資料值。各屬性具有一名稱與一值。特定的屬性名稱及任一特定元件所可能擁有的子元件，皆取決於元件的類型。對各屬性值的解譯也取決於對應的屬性名稱及擁有該屬性的元件。

如圖1A所示，一XMT-A檔案100主要由兩部分組成，一Header元件110及一Body元件120。Header元件110包含一單一子元件，其定義為一InitialObjectDescriptor元件130。Body元件120包含一或多個「par」元件140，用作子元件。

InitialObjectDescriptor130具有一屬性，即ObjectDescriptorID (ODID)，其值為一字元串。如圖1B所示，該元件具有兩子元件，一Profiles元件150及一Descr元件160。Profiles元件150無子元件。Profiles元件150擁有若干屬性，包括「includeInclineProfileLevelFlag」、「sceneProfileLevelIndication」、「ODProfileLevel Indication」、「audioProfileLevelIndication」、「visualProfile LevelIndication」及「graphicsProfileLevelIndication」。

Descr元件160可具有若干類型的子元件。本發明所必需的唯一類型係單一「esDescr」元件170。一esDescr元件170可擁有一或多個「ES_Descriptor」子元件180、190。一ES_Descriptor元件指定一「基本資料流」的某些性質，「基本資料流」為動畫專家群第四階文件中所定義的一個概念。下面說明一ES_Descriptor元件的結構。

附屬於一InitialobjectDescriptor元件130的esDescr元件

170可擁有一或兩個ES_Descriptor元件180、190。在各情形中，對於定義為「sdsm」或「場景描述資料流」的基本資料流，皆應有一ES_Descriptor 180。此外，對於定義為「odsm」或「物件描述符資料流」的基本資料流，可能有一第二ES_Descriptor 190。對於依賴於聲頻資料、視覺資料或其他sdsm中未指定之媒體資料類型的XMT-A檔案，僅需要odsm的ES_Descriptor元件190。

如圖2A所示，各par元件140、200含有一或多個「par-child」元件210。「par-child」元件可為另一par元件、一odsm命令或一bifs命令。各par元件也包含一屬性，其名稱為「開始」。開始屬性之值指定欲執行par元件內的odsm或bifs命令的時間。由par元件的開始屬性所決定之時間值，係相對於任何母元件所隱含的時間值計算而得，且Body元件120暗示開始時間為零。

par-child元件210可包含兩類odsm命令元件實例，如圖2B所示。其包括ObjectDescriptorUpdate元件220及ObjectDescriptorRemove元件250。一ObjectDescriptorUpdate 220包含一單一OD子元件230，而OD元件230包含一單一ObjectDescriptor子元件240。下面詳細說明ObjectDescriptor元件240。ObjectDescriptorRemove元件250具有一屬性，但無子元件。ObjectDescriptorRemove元件250的屬性名稱為「ODID」。

par-child元件210可包含三類bifs命令元件實例，如圖3所示。其包括Insert元件300、Delete元件310及Replace元件320。

如圖3A所示，Insert元件300可具有一「xmtaBifsNode」子元件330或一「ROUTE」子元件340。Delete元件310無子元件。Replace元件320可具有一「xmtaBifsNode」子元件350、一「ROUTE」子元件360或一「Scene」子元件370。Scene元件具有一「xmtaTopNode」子元件380。Scene元件也可具有一或多個ROUTE子元件390。

ROUTE元件340、390無子元件。ROUTE元件340、390的屬性包括「fromNode」、「fromField」、「toNode」及「toField」。

術語「xmtaBifsNode元件」330代表約100個定義的BIFS節點元件中任何之一。此等元件各具有圖4所示的一般結構400。各xmtaBifsNode元件400代表一BIFS節點，即在動畫專家群第四階系統規格(2001年8月，第9章，ISO-IEC文件ISO/IEC 14496-1:2001)中所定義的二進制資料結構。關於該文件的資訊，可在http://mpeg.telecomitalialab.com/documents.htm及國際標準化組織(ISO)獲得，其地址為瑞士CH-1211日內瓦20，Case postale 56，rue de Varembé。各xmtaBifsNode元件400的元件標籤皆基於動畫專家群第四階系統規格中所定義的對應NodeName。某些類型的xmtaBifsNode元件可具有附屬(子)元件，其基於對應的BIFS節點的某些性質。此等稱為nodeField元件410。各nodeField元件可具有一或多個附屬元件，其由另外的xmtaBifsNode元件420組成。該配置可以遞迴方式重複，以描述BIFS節點的階層式樹狀結構。該階層的深度沒有限制。

各BIFS節點具有數個稱為「欄位」的性質。此等欄位各

具有一定義的欄位名稱(字串)及欄位資料類型(布林、整數、浮點等)。其中一種欄位資料類型為「節點」。除節點之外，所有欄位資料類型由xmtaBifsNode元件400的類似名稱屬性表示。具有「節點」類型的各欄位由xmtaBifsNode元件400的類似名稱子元件410表示。xmtaBifsNode元件400的各子元件410可具有一或多個xmtaBifsNode元件420，用作子元件(即xmtaBifsNode母元件400的孫元件(grandchildren))。

圖5說明一XMT-A BIFS節點的XML代表。各XMT-A BIFS節點元件由一NodeName標籤500、570識別，其唯一識別100個XMT-A BIFS節點可能類型之一。各節點元件可為一初始節點元件500或一再用節點元件570。在初始節點元件500的情形中，可用一可選屬性「DEF」510，對一特定節點進行唯一的文數描述。若該屬性已提供，則將該節點歸類為「可再用」。

一初始XMT-A BIFS節點元件也擁有一組欄位屬性520，對於為NodeName之節點類型而定義的各性質欄位皆具有一欄位屬性，且具有「節點」或「緩衝」之外的一節點資料類型。此等屬性在圖5A中標識為「field0」、「field2」、「field3」及「field5」。對於「NodeName」類型的節點，此等屬性各自的實際名稱由動畫專家群第四階系統規格中所定義的對應性質欄位名稱決定。指派給此等各屬性的值必須代表資料值，該等資料值具有動畫專家群第四階系統規格中所定義的一節點資料類型(布林、整數、浮點等)。

此外，初始XMT-A BIFS節點元件500可具有一或多個欄

位值子元件530、540，其元件標籤對應於具有「節點」或「緩衝」資料類型的性質欄位的欄位名稱。此類欄位值元件各具有一起始標籤530及一結束標籤540。圖5A中，此類欄位值元件530、540的範例由元件標籤<field1>...</field1>及<field4>...</field4>表示。

在具有「節點」資料類型之性質欄位的情形中，該欄位值元件可包含一或多個與BIFS-節點元件550對應的子元件。此類BIFS-節點子元件的範例由元件標籤<NodeName1.../>、<NodeName2.../>及<NodeName 3.../>表示。

在具有「緩衝」資料類型之性質欄位的情形中，該欄位值元件可包含一或多個與BIFS命令元件300、310、320對應的子元件。

若一XMT-A BIFS節點元件包括任何欄位值子元件，則該節點元件將遵循標準的XML原則，由</NodeName>結束標籤560終止。

前述對一XMT-A BIFS節點元件的定義可以遞迴形式應用於各附屬BIFS節點元件(<NodeName1>等)，允許建立一階層式節點樹結構。該XMT-A BIFS節點樹結構的深度也沒有限制。

在再用節點570的情形中，節點元件僅有一屬性，且無子元件。該唯一屬性為一「USE」屬性580，其值590為節點ID串。作為USE屬性之值提供的節點ID串，必須匹配指定為具有相同NodeName之一初始節點元件500的DEF屬性510之節點ID串。

術語「xmtaTopNode」代表所定義的xmtaBifsNode元件子集之一，其允許用作Scene元件的子元件。

如圖6A所示，ObjectDescriptor元件240、600(ObjectDescriptorUpdate元件220的孫元件)類似於上述的InitialobjectDescriptor元件130。與InitialobjectDescriptor元件130不同，ObjectDescriptor元件240、600缺乏Profiles子元件150。而與InitialobjectDescriptor元件130相同，ObjectDescriptor元件240、600都具有一「ObjectDescriptorID」(ODID)屬性606。典型的ObjectDescriptor元件240、600具有一單一Descr子元件610，Descr元件610具有一單一esDescr子元件620，而esDescr元件620具有一單一ES_Descriptor子元件630。Descr元件610、esDescr元件620及ES_Descriptor元件630，皆類似於InitialobjectDescriptor元件130的對應子元件160、170、180、190。

ES_Descriptor元件180、190、630可包含於ObjectDescriptor元件600，或InitialobjectDescriptor元件130中。在任一情形中，ES_Descriptor元件180、190、630皆具有圖6B中所示的結構640。ES_Descriptor元件640的「ES_ID」屬性636之值係對各資料流唯一的文數串。ES_Descriptor元件640始終具有一decConfigDescr子元件646及一slConfigDescr子元件660。若一ES_Descriptor元件630、640附屬於一ObjectDescriptor元件600，則ES_Descriptor元件630、640也具有一StreamSource子元件670。若一ES_Descriptor元件180、190、640附屬於一InitialobjectDescriptor元件130，則

ES_Descriptor元件180、190、640無StreamSource子元件670。

decConfigDescr元件646具有一DecoderConfigDescriptor子元件650。DecoderConfigDescriptor元件650具有若干屬性，包括「streamType」與「objectTypeIndication」，指示ES_Descriptor母元件640是否代表聲頻、視覺、sdsm、odsm或其他類型的媒體。DecoderConfigDescription元件650也可具有一decSpecificInfo子元件656，其取決於streamType與objectTypeIndication的值。

在一sdsm(場景描述資料流)的ES_Descriptor元件180的情形中，DecoderConfigDescriptor元件650具有一decSpecificInfo子元件656。如圖6C所示，decSpecificInfo元件680具有一BIFSConfig子元件686。BIFSConfig元件686擁有若干屬性，其指定將如何編碼BIFS節點。BIFSConfig元件686也擁有commandStream元件690，而commandStream元件690擁有一「size」元件696。

slConfig元件660具有一SLConfigDescriptor子元件666。SLConfigDescriptor元件666具有一名為「預定義」的屬性，無子元件。「預定義」屬性始終具有值「2」。

StreamSource元件670具有一屬性「url」，但無子元件。url屬性之值指定一檔案名稱，或指示多媒體資料檔案位置的一網際網路位址(URL，統一資源定位器)，該多媒體資料檔案包含聲頻資料、視覺資料或其他定義一特定資料流的實際聲音、影像等的資料。對於sdsm(場景描述資料流)或odsm(物件描述符資料流)，不存在StreamSource元件670，

由於此等流皆由XMT-A檔案所決定。

2.0 動畫專家群第四階中間格式檔案

動畫專家群第四階中間格式檔案係一電子資料形式，其具有動畫專家群第四階系統規格文件(ISO-IEC文件ISO/IEC 14496-1:2001，2001年8月)第13章中定義的結構與組成。該電子資料結構形式係通常稱之為「二進制檔案」的範例，因為其由一二進制資料值序列組成，該等資料值不限於代表字母、數字與標點。此即允許資料結構比典型的本文檔案(如XMT-A檔案)所能提供的結構更緊密。具有動畫專家群第四階中間格式所定義之結構的電子資料儲存形式稱為「mp4二進制檔案」。與XMT-A檔案不同，多數本文編輯軟體不能解譯mp4二進制檔案。

動畫專家群第四階中間格式源自於蘋果電腦公司1996年所定義的QuickTime (r)檔案格式，可造訪http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/refFileFormat96.htm and http://developer.apple.com/techpubs/quicktime/qtdevdocs/PDF/QTFileFormat.pdf線上獲取。QuickTime係蘋果電腦公司的註冊商標。

因動畫專家群第四階中間格式繼承於QuickTime (r)，故其保留了源於QuickTime (r)規格的數個特徵。此等特徵包括以「原子」作為一資料結構單位的概念。各原子具有兩部分，一標頭及一主體。標頭含有一原子大小值，其指定包括該原子(包括該標頭)的位元組數。標頭也含有一atomId，其指定該原子的類型。一原子主體包含該原子所載送的資

料。該資料可包括附屬原子。以其基本形式，一原子具有一包括四位元組(無符號整數)的原子大小值，atomId也包括四位元組(字元)。動畫專家群第四階規格說明中也定義了具有原子大小值，及4位元組以上atomId值的原子擴展形式。

如圖7A所示，mp4二進制檔案700由一或多個「mdat」原子706與一「moov」原子712組成。moov原子712可在mdat原子706之前或之後。如圖7B所示，各mdat原子718由原子大小值724、之後的四位元組atomId「mdat」730及稱為「chunks(大塊)」736的一資料區塊序列組成。如圖7C所示，各大塊742由一媒體資料「樣本」748序列組成。各樣本748指定與單一媒體資料流的特定時間點相關的資料區塊。單一大塊中的所有樣本必須代表相同的媒體資料流。藉由檢查一mdat原子700也必然不可能識別個別樣本748或大塊736、742。各樣本748與大塊736、742可用儲存在mp4二進制檔案中其他位置的表格來識別。

如圖7D所示，moov原子754由一原子大小值760、之後的四位元組atomId「moov」766及若干附屬原子組成，該等附屬原子包括一「mvhd」(moov標頭)原子772、一「iods」(初始物件描述符)原子778及一或多個「trak」原子790。對於各資料流，「moov」原子712、754包括一「trak」原子790，其包括(若有)sdsm(場景描述資料流)與odsm(物件描述符資料流)。「moov」原子712、754也可包括一可選的「udta」(使用者資料)原子784。「udta」原子784可用於將可選資訊(如版權訊息)嵌入mp4二進制檔案中。

mvhd原子772由原子大小值、之後的四位元組atomId「mvhd」及數個資料值組成，該等資料值包括時間與日期戳記、時間標度值及檔案持續時間值。mvhd原子的原子大小值始終為108。時間與日期戳記指示何時建立檔案。時間標度值指示每秒的刻點數，用於代表檔案的時間值。檔案持續時間值指示呈現檔案中材料所需的總時間，以時間標度值所指定的時間單位表示。

如圖8A所示，一iods原子800由一原子大小值804、之後的四位元組atomId「iods」808、一8位元version(版本)值812、一24位元flag(旗標)值816及一Mp4fInitObjDescr資料結構820組成。如圖8B所示，Mp4fInitObjDescr資料結構824由一位元組MP4_IOD-TAG值828、後續資料區塊中的位元組數832、一10位元ObjectDescriptorID836、兩旗標位元840、844、四個reserved(保留)位元848及五個設定檔等級指示值852、856、860、864、868組成。設定檔等級指示值之後為一或兩個動畫專家群第四階ES_ID_Inc資料結構872。ES_ID_Inc結構指示對應於sdsm(場景描述資料流)的trak原子790之ES_ID值。第二ES_ID_Inc結構(若有)指示對應於odsm(物件描述符資料流)的trak原子790之ES_ID。只有存在odsm時，才存在第二ES_ID_Inc結構。如圖8C所示，各ES_ID_Inc資料結構由一位元組ES_ID_IncTag值880、後續資料中的位元組數884(始終為4)及一32位元ES_ID值888組成。

如圖9A所示，各trak原子900由原子大小值903、之後的四位元組atomId「trak」906、「tkhd」(磁軌標頭)原子910及一

「mdia」(媒體)原子912組成。在代表odsm(物件描述符資料流)的trak原子情形中，trak原子也包括一「tref」(磁軌參考)原子940。在延遲啟動之磁軌的情形中，提供一「edts」(編輯列表)原子945，指示何時啟動磁軌。mdia原子912由一「mdhd」(媒體標頭)原子915、一「hdlr」(處置器)原子918、一「minf」(媒體資訊)原子920、一「stbl」(樣本表)原子933及一媒體資訊標頭原子936組成。標號「*mhd」代表若干媒體資訊標頭原子類型中任一個，該等類型包括「nmhd」(對於sdsm與odsm磁軌)、「smhd」(對於聲頻磁軌)、「vmhd」(對於視覺磁軌)等。

tkhd原子910、mdhd原子915及hdlr原子918包含數個資料值，包括一trackId號碼、時間與日期戳記、媒體時間標度及媒體持續時間值。各磁軌具有其自己的時間標度，其可與mvhd原子772中指定的總體時間標度不同。

如圖9B所示，樣本表原子950由一原子大小值954、之後的四位元組atomId「stbl」957及一系列樣本表原子960、963、966、970、974、978組成。各種樣本表原子可為任意順序。此等包括一「stsc」(樣本至大塊表)原子960、一「stts」(時間至樣本表)原子963、一「stco」(大塊偏移表)原子966、一「stsz」(樣本大小表)原子970、一可能的「stss」(同步樣本表)原子974及一「stsd」(樣本描述表)原子978。各樣本表原子皆包含二進制媒體資料736、748的資料描述性質，該等資料儲存在相關的mdat原子706、718中。

樣本至大塊表原子(stsc原子)960由一原子大小值、一四

位元組atomId(「stsc」)、一32位元無符號整數(numStscEntries)及一樣本至大塊資料記錄序列組成。numStscEntries之值指定樣本至大塊資料記錄序列中的項目數。各樣本至大塊資料記錄由三個32位元無符號整數組成，該等整數指定一起始大塊號碼、每一大塊中的樣本數及一樣本描述索引。樣本描述索引為樣本描述表978中一項目的索引。每一大塊中的樣本數，指定起始大塊號碼所指定的大塊736中，及由下一項目所指定的該起始大塊之前所有後續大塊的樣本748之數目。

時間至樣本表原子(stts原子)963由一原子大小值、一四位元組atomId(「stts」)、一32位元無符號整數(numSttsEntries)及一時間至樣本資料記錄序列組成。numSttsEntries之值指定時間至樣本資料記錄序列中的項目數。各時間至樣本資料記錄由兩個32位元無符號整數組成，其指定一樣本計數及以磁軌時間標度單位表示的一樣本持續時間。樣本計數值指定具有對應樣本持續時間的連續樣本748的數目。

大塊偏移表原子(stco原子)966由一原子大小值、一四位元組atomId(「stco」)、一32位元無符號整數(numstcoEntries)及一大塊偏移值序列組成。numstcoEntries之值指定大塊偏移值序列中的項目數。該序列中的各項目由一32位元無符號整數組成，其指定mp4檔案起端，與一mdat原子718中對應的大塊736之起端之間的位元組數。

樣本大小表原子(stsz原子)970由一原子大小值、一四位元組atomId(「stsz」)及一32位元無符號整數(iSampleSize)

組成，該整數指定與該trak原子900相關的所有媒體資料樣本748的大小。若與該trak原子相關的媒體資料樣本的大小並不全部相等，則iSampleSize之值指定為零，之後為一32位元無符號整數(numStszEntries)及一樣本大小值序列。numStszEntries之值指定樣本大小值序列中的項目數。該序列中的各項目由一32位元無符號整數組成，其指定與該trak原子900相關之媒體資料718中，對應樣本748中的位元組數。

同步樣本表原子(stss原子)974(若有)由一原子大小值、一四位元組atomId(「stss」)、一32位元無符號整數(numStssEntries)及一樣本索引值序列組成。numStssEntries之值指定樣本索引值序列中的項目數。該序列中的各項目由一32位元無符號整數組成，其指定一「隨機存取樣本」的樣本索引。隨機存取樣本索引識別一媒體資料樣本748，其對應於與該trak原子900相關之媒體資料中的一點，在該點處，一媒體播放器可開始處理該媒體資料，而不必考慮前面的任何樣本。該表中的樣本索引值必須單調遞增。

樣本描述表原子(stsd原子)978由一原子大小值、一四位元組atomId(「stsd」)、一32位元無符號整數(numStsdEntries)及一樣本描述資料記錄序列組成。numStsdEntries之值指定樣本描述資料記錄序列中的項目數。各樣本描述資料記錄指定用於編碼媒體資料樣本的裝置，該資料樣本由樣本至大塊資料記錄中對應的索引所識別。樣本描述資料記錄序列通常具有一單一項目(numStsdEntries=1)，其指定媒體類

型(聲頻、視覺、sdsm、odsm)、用於聲頻與視頻樣本等的壓縮演算法。各樣本描述表項目包含在一「mp4*」原子982中，其中「mp4*」為「mp4s」(對於sdsm與odsm樣本)、「mp4a」(對於聲頻樣本)及「mp4v」(對於視覺樣本)的一般替代表示。各「mp4*」原子982包含一「esds」(基本流描述符)原子986，各esds原子986包含一動畫專家群第四階基本流描述符(Es_Descr)資料結構990。

圖10A中顯示動畫專家群第四階基本流描述符1000的結構。該資料結構由一位元組標籤(ES_DescrTag)1004、之後的剩餘資料結構中位元組數指示1008、一16位元ES_ID值(通常為零)1012、三個1位元旗標(streamDependenceFlag、URL_Flag及OCRstreamFlag)1016及一5位元資料流優先值1020組成。若三旗標1016中任一個不為零，則資料流優先值1020後可有附加資料值(未顯示)。此等可選資料值並非本發明所必需。

資料流優先值1020後跟隨解碼器組態描述符資料結構1024及同步層配置描述符資料結構1028。圖10B中顯示解碼器組態描述符1024、1032的結構。該資料結構由一位元組標籤(DecoderConfigDescrTag)1036、之後的剩餘資料結構中位元組數指示1040及一系列資料值組成：objectType1044、streamType 1048、upStreambit 1052、reservedbit 1056、bufferSizeDB 1060、maxBitrate 1064及avgBitrate 1068。此等值之後可跟隨依賴streamType與objectType的解碼器特定資訊資料結構1072。解碼器特定資訊資料結構1072係sdsm

所需，而非odsm所需。多數聲頻與視覺媒體資料流也具有解碼器組態描述符1032中的解碼器特定資訊資料結構。

圖10C中顯示解碼器特定資訊資料1072、1076之結構。該資料結構由一位元組標籤(DecoderSpecificInfoTag)1080、之後的剩餘資料結構中位元組數指示1084組成。其餘位元組取決於objectType與streamType。在sdsm(場景描述資料流or BIFS)的情形中，解碼器特定資訊1072、1076包括用於編碼nodeID值的位元數及用於編碼routeID值的位元數之指示。此等值各由一5位元無符號整數代表。

圖10D中顯示同步層配置描述符1028、1088的結構。該資料結構由一位元組標籤(SLConfigDescrTag)1092、之後的剩餘資料結構中位元組數指示(始終為1)1094及一單一資料位元組(值2，「預定義」)1098組成，該資料位元組指示欲用於同步層之預定義配置。

3.0 場景描述資料流(sdsm)

用於編碼或解碼特定類型之聲頻與視覺資料流的裝置，並非由XMT-A規格說明或動畫專家群第四階中間檔案規格說明所決定，因而在此將不必涵蓋如何編碼此等流的細節。XMT-A文件包含有關場景描述資料流(sdsm)與物件描述資料流(odsm)內容的詳細資訊。因此，各XMT-A文件與動畫專家群第四階中間檔案中所含的sdsm與odsm資料流密切相關。該部分說明sdsm資料結構，其後部分說明odsm資料結構。

與其他任何媒體資料流一樣，sdsm資料包括一或多個大

塊，各大塊包括一或多個樣本。如圖11A所示，一sdsm二進制大塊1100中的各樣本定義為一「命令框」1110。各命令框1110係位元組對齊。如圖11B所示，各命令框1110由一或多個「BIFS命令」1120組成。「BIFS」代表「資料流的二進制格式」。各BIFS命令1120之後為一連續位元1130、1140。若連續位元1130的值為(1)，則其後跟隨另一BIFS命令。否則對於連續位元1140，其後跟隨足夠數目的空填充位元1150，以完成最後位元組。除了命令框中的第一BIFS命令，個別BIFS命令1120一般皆非位元組對齊。

如圖12所示，有四類BIFS命令：「插入」、「刪除」、「替換」及「場景替換」。BIFS插入命令1200由兩位元插入碼(值=「00」)1206、之後的兩位元參數類型碼1210及插入命令資料1216組成。BIFS刪除命令1220由兩位元刪除碼(值=「01」)1226、之後的兩位元參數類型碼1230及刪除命令資料1236組成。BIFS替換命令1240由兩位元替換碼(值=「10」)1244、之後的兩位元參數類型碼1250及替換命令資料1260組成。BIFS場景替換命令1270由兩位元場景替換碼(值=「11」)1280、之後的BIFS場景資料結構1290組成。

如圖13所示，有三類BIFS插入命令：(a)Node插入命令；(b)Indexed Value插入命令；及(c)Route插入命令。插入命令的類型由參數類型值1210決定。Node插入命令1300由兩位元插入碼(值=「00」)1304、之後的兩位元參數類型碼(值=「01」，類型=節點)1308、nodeID值1312、一兩位元插入位置碼1316及一SFNode資料結構1324組成。若插入位置碼

1316的值為零，則該插入位置碼1316後跟隨一8位元位置值1320。nodeID值1312指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼該nodeID值及其他nodeID值的位元數，在sdsm資料流的解碼器特定資訊1072中指定。下面解說SFNode資料結構1324之結構。

Indexed Value插入命令1328由兩位元插入碼(值=「00」)1332、之後的兩位元參數類型碼(值=「10」，類型=Indexed Value)1336、一nodeID值1340、一inFieldID值1344、一兩位元插入位置碼1348及一欄位值資料結構1356組成。若插入位置碼1348的值為零，則該插入位置碼1348後跟隨一8位元位置值1352。nodeID值1340指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼nodeID值1340的位元數，在sdsm資料流的解碼器特定資訊1072中指定。inFieldID值1344識別nodeID 1340之值所指定的BIFS節點的資料欄位之一。用於編碼inFieldID值1344的位元數取決於動畫專家群第四階系統規格中包含的表。

對於一特定BIFS節點的一特定資料欄位，欄位值資料結構的內容取決於欄位資料類型(布林、整數、浮點、字串、節點等)。可如一位元般簡單，或如一SFNode資料結構般複雜。各BIFS節點之各資料欄位的欄位資料類型，由動畫專家群第四階系統規格中包含的表指定。

Route插入命令1360由兩位元插入碼(值=「00」)1364、之後的兩位元參數類型碼(值=「11」，類型=Route)1368、一「isUpdateable」位元1372、一departureNodeID值1380、一

departureFieldID 值 1384 、 一 arrivalNodeID 值 1388 及 一 arrivalFieldID值1392組成。若「isUpdateable」位元之值為(1)，則「isUpdateable」位元1372之後跟隨一routeID值1376。用於編碼departureNodeID值1380與arrivalNodeID值1388的位元數，在sdsm資料流的解碼器特定資訊1072中指定。用於編碼departureFieldID值1384的位元數與用於編碼arrivalFieldID值1392的位元數，取決於動畫專家群第四階系統規格中包含的表。

如圖14所示，有三類BIFS刪除命令。(a)Node刪除命令；(b)Indexed Value刪除命令；及(c)Route刪除命令。刪除命令的類型由參數類型值1230決定。Node刪除命令1400由兩位元刪除碼(值=「00」)1406、之後的兩位元參數類型碼(值=「00」，類型=Node)1412及nodeID值1418組成。nodeID值1418指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼nodeID值1418的位元數，在sdsm資料流的解碼器特定資訊1072中指定。

Indexed Value刪除命令1424由兩位元刪除碼(值=「01」)1430、之後的兩位元參數類型碼(值=「10」，類型=Indexed Value)1436、一nodeID值1442、一inFieldID值1448及一兩位元刪除位置值1454組成。nodeID值1418指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼nodeID值1418的位元數，在sdsm資料流的解碼器特定資訊1072中指定。

Route刪除命令1466由兩位元刪除碼(值=「10」)1472、之

後的兩位元參數類型碼(值=「11」，類型=Route)1478及routeID值1484組成。routeID值1484指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼routeID值1484的位元數，在sdsm資料流的解碼器特定資訊1072中指定。

如圖15所示，有四類替換命令：(a)Node替換命令；(b)Field替換命令；(c)Indexed Value替換命令；及(d)Route替換命令。替換命令的類型由參數類型值1250決定。Node替換命令1500由兩位元替換碼(值=「10」)1504、之後的兩位元參數類型碼(值=「01」，類型=Node)1508、nodeID值1510及一SFNode資料結構1514組成。nodeID值1510指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼nodeID值1510的位元數，在sdsm資料流的解碼器特定資訊1072中指定。下文將解說SFNode資料結構1514之結構。

Field替換命令1520由兩位元替換碼(值=「10」)1524、之後的兩位元參數類型碼(值=「01」，類型=Field)1528、一nodeID值1530、一inFieldID值1534及一欄位值資料結構1538組成。nodeID值1530指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼nodeID值1530的位元數，在sdsm資料流的解碼器特定資訊1072中指定。inFieldID值1534識別nodeID 1530之值所指定的BIFS節點的資料欄位之一。用於編碼inFieldID值1534的位元數取決於動畫專家群第四階系統規格中包含的表。

Indexed Value替換命令1540由兩位元替換碼(值=「10」)1544、之後的兩位元參數類型碼(值=「10」，類型=

Indexed Value)1548、一nodeID值1550、一inFieldID值1554、一兩位元替換位置碼1558及一欄位值資料結構1564組成。若替換位置碼1558的值為零，則該替換位置碼1558後跟隨一8位元位置值1560。nodeID值1550指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼nodeID值1550的位元數，在sdsm資料流的解碼器特定資訊1072中指定。inFieldID值1554識別nodeID 1550之值所指定的BIFS節點的資料欄位之一。用於編碼inFieldID值1554的位元數取決於動畫專家群第四階系統規格中包含的表。

Route替換命令1570由兩位元替換碼(值=「10」)1574、之後的兩位元參數類型碼(值=「11」，類型=Route)1578、一routeID值1580、一departureNodeID值1584、一departureFieldID值1588、一arrivalNodeID值1590及一arrivalFieldID值1594組成。routeID值1580指定BIFS命令中其他地方所定義的一組可更新節點之一。用於編碼routeID值1580的位元數，在sdsm資料流的解碼器特定資訊1072中指定。用於編碼departureNodeID值1584與arrivalNodeID值1590的位元數，在sdsm資料流的解碼器特定資訊1072中指定。用於編碼departureFieldID值1588的位元數與用於編碼arrivalFieldID值1594的位元數，取決於動畫專家群第四階系統規格中包含的表。

如圖12D所示，ReplaceScene BIFS命令1270由兩位元場景替換碼(值=「11」)1280、之後的BIFS場景資料結構1290組成。如圖16所示，BIFS場景資料結構1600由一6位元保留欄

位1610、兩個一位元旗標(USENAMES 1620與protoList 1630)、一SFTopNode資料結構1640及一位元旗標(hasRoutes)1650組成。若protoList旗標1630為真(1)，則該protoList旗標後跟隨動畫專家群第四階系統規格中所定義的附加資料。SFTopNode資料結構1640係圖17所示之SFNode資料結構的特殊情形。若hasRoutes旗標1650為真(1)，則該hasRoutes旗標後跟隨一Routes資料結構1660。圖18中顯示Routes資料結構之結構。

如圖17A、17B及17C所示，SFNode資料結構可為三種形式之一：(a)再用；(b)遮罩節點；及(c)列表節點。三種形式皆以一位元旗標(isReused)開始。在再用SFNode 1700的情形中，isReused旗標之值為「1」(真)1704，剩餘的SFNode資料結構由nodeIDref值1708組成。nodeIDref 1708之值必須匹配sdsm資料中，其他地方所定義的一可更新SFNode的nodeID值。

若isReusedFlag為假(0)1712、1732，則根據maskAccess旗標位元1722、1742之值，SFNode類型可具有圖17B與17C所示的兩形式之一。在任一情形中，SFNode之資料包括：本地節點類型值(localNodeType)1714、1734，一位元旗標(isUpdateable) 1716、1736，及第二一位元旗標(maskAccess) 1722、1742。用於編碼本地節點類型1714、1734的位元數取決於動畫專家群第四階系統規格中規定的表。若isUpdateable旗標1716、1736為真(1)，則該isUpdateable旗標後跟隨一nodeID值1718、1738。若該isUpdateable旗標1716、

1736為真(1)且相關的BIFSScene資料結構1600中的USENAMES旗標1620也為真(1)，則該nodeID值1718、1738後跟隨一空終止字串(「名稱」)1720、1740。

若該maskAccess位元為真(1)1722，則SFNode具有「遮罩節點」結構1710。在此情形中，如圖17B所示，maskAccess位元1722後跟隨一遮罩位元1726之有序序列，即對於具有localNodeType 1714之值所給定之節點類型的BIFS節點，動畫專家群第四階規格說明中所定義的各nFields性質欄位的一序列。在此等遮罩位元之一為真(1)的各情形中，該遮罩位元後跟隨根據一欄位資料類型(整數、布林、字串、節點等)編碼的一二進制欄位值1728，該欄位資料類型由localNodeType 1734、欄位數(在遮罩位元序列中的位置)及動畫專家群第四階規格說明中所定義的表決定。

若該maskAccess位元為假(0)1742，則SFNode具有「列表節點」結構1730。在此情形中，如圖17C所示，MaskAccess位元1742後跟隨一或多個欄位參考記錄。各欄位參考記錄以一位元結束旗標1744、1750開始。若結束旗標為假(0)1744，則結束旗標1744後跟隨為該本地節點類型1734而定義的一性質欄位的欄位參考索引號碼(fieldRef)1746，而該fieldRef值1746後跟隨根據欄位資料類型(整數、布林、字串、節點等)編碼的二進制欄位值1748，該欄位資料類型由本地節點類型1734及fieldRef值1746所指示的性質欄位決定。用於編碼fieldRef值1746的位元數由動畫專家群第四階系統規格中所定義的表決定。若結束旗標為真(1)1750，則

欄位值列表終止。

包含在一SFNode結構中的各性質欄位值，可由一單一資料值(SFField資料結構)或多個資料值(MFField資料結構)組成。各MFField資料結構包含零或多個MFField組件。如圖17D與圖17E所示，根據isList位元1766、1786之值，有兩種形式的MFField結構，列表形式1760與向量形式1780。兩種形式皆以一位元保留位元1762、1782開始，其後跟隨isList位元1766、1786。

若isList位元具有值(1) 1766，則MFField資料結構具有列表形式1760。在此情形中，isList位元1766後跟隨一位元endFlag值1770、1772之序列。若endFlag位元之值為「0」1770，則endFlag位元後跟隨一SFField資料結構1774。若endFlag位元之值為「1」1772，則MFField資料結構結束。

若isList位元具有值(0) 1786，則MFField資料結構具有向量形式1780。在此情形中，isList位元1786後跟隨5位元欄位(nBits)1790，其指定後續欄位計數值(nFields)1792中的位元數。之後跟隨一nFields SFField結構1796之序列。

各SFField值之結構取決於與對應的性質欄位相關的特定欄位資料類型，如動畫專家群第四階系統規格中指定表格中所指示內容。例如，一布林欄位由一單一位元組成。包括整數、浮點、字串、SFNode的其他情形，在動畫專家群第四階系統規格中定義且描述。

BIFS場景資料結構1600的最後組件係一可選Routes資料結構1660。如圖18A與18B所示，有兩種形式的Routes資料

結構，列表形式1800與向量形式1830。兩種形式的Routes資料結構皆以一位元列表旗標1805、1835開始。若列表旗標之值為真(1)1805，則Routes資料結構具有列表形式1800。在此情形中，列表位元1805後跟隨一或多個Route資料結構1810，而各Route資料結構1810之後跟隨一位元moreRoutes旗標1815、1820。若moreRoutes旗標之值為真(1)1815，則之後跟隨另一Route資料結構1810。若moreRoutes旗標之值為假(0)1820，則Routes資料結構1800結束。

若Routes資料結構中的列表旗標之值為假(0)1835，則Routes資料結構具有向量形式1830。在此情形中，列表位元1835之後跟隨一五位元nBits欄位1840。nBits欄位中所含的無符號整數值指定編碼後續numRoutes值1845所用的位元數。以numRoutes值1845編碼的無符號整數，指定跟隨在numRoutes值之後的Route資料結構1850數。

如圖18C所示，Route資料結構1860由一位元旗標(isUpdateable)1865、一outNodeID值1880、一outFieldRef值1885、一inNodeID值1890及一inFieldRef值1895組成。若該isUpdateable旗標1865之值為真(1)，則isUpdateable旗標1865後跟隨一routeID值1870。若該isUpdateable旗標1865之值為真(1)，且對應的BIFS場景資料結構1600中的USENAMES旗標1620之值也為真(1)，則routeID值1870後跟隨一空終止字串(routeName)1875。用於編碼outNodeID值、inNodeID值及routeID值的位元數，在sdsm資料流的解碼器特定資訊1072中指定。用於編碼outFieldRef與inFieldRef的位元數，由動

畫專家群第四階系統規格中所定義的表決定。

4.0 物件描述符資料流(odsm)

與任何其他動畫專家群第四階基本資料流一樣，odsm(物件描述符資料流)也包含在一或多個大塊736序列中。如圖19所示，各odsm大塊1900包括一odsm樣本1920序列，而各odsm樣本1940包括一odsm命令1960序列。各odsm大塊1900中的odsm樣本1920數，由物件描述符資料流的trak原子790、900中的樣本至大塊表原子(stsc)960的內容決定。各odsm樣本1940中的odsm命令1960數，由物件描述符資料流的trak原子790、900中的樣本大小表原子(stsz)970決定。

可能有兩種odsm命令，ObjectDescriptorUpdate命令與ObjectDescriptorRemove命令。如圖20A所示，ObjectDescriptorUpdate命令2000由一位元組ObjectDescriptorUpdateTag2010、剩餘命令中的位元組數(numBytes)指示2020及ObjectDescriptors 2030序列組成。圖21概述一ObjectDescriptor結構。如圖20B所示，ObjectDescriptorRemove命令2040由一位元組ObjectDescriptorRemoveTag2050、剩餘命令中的位元組數(numBytes)指示2060、ObjectDescriptorsId值2070序列及2至6個填充位元2080組成。

各numBytes值2020、2060指定剩餘odsm命令中的位元組數。若numBytes之值小於128，則該值以單一位元組編碼。否則，numBytes之值以大小位元組序列編碼。各大小位元組中的高階位元指示是否將跟隨另一大小位元組。若該高

階位元為「1」，則跟隨另一大小位元組。各大小位元組中剩餘的七位元指定最終的numBytes無符號整數值的七位元。

各objectDescriptorId值2070以10位元編碼，且將objectDescriptorRemove命令2040中找到的10位元objectDesciptorId值序列壓縮於一位元組序列中。若objectDescriptorId數值不是4的倍數，則在最後的objectDescriptorId值後跟隨兩、四或六個空位元2080，以填充該命令中的最末位元組。

如圖21A所示，ObjectDescriptorUpdate命令中的一ObjectDescriptor 2100由一位元組MP4_OD_Tag 2108、之後跟隨的numBytes值2116、十位元ObjectDescriptorID值2124、一位元URL_Flag值2132、五位元保留欄位(0x1f)2140及一ES_Descr資料結構或一EsIdRef資料結構2148組成。在該形式的ObjectDescriptor中，URL_Flag 2132的值總為假(0)。numBytes值2116指定包括該剩餘物件描述符的位元組數，且係以對ObjectDescriptorUpdate命令2000或ObjectDescriptorRemove命令2040中找到的numBytes值2020、2060而指定的相同方式進行編碼。

圖10A中顯示ES_Descr資料結構1000之結構。如圖21B所示，EsIdRef資料結構2160由一位元組ES_ID_RefTag 2170、numBytes值2180及16位元基本資料流ID(ES_ID)值2190組成。在此情形中，numBytes之值始終為「2」，且該值指定為一8位元整數。

圖22一般顯示本發明之操作。根據一動畫專家群第四階中間檔案2210的內容，過程2200建立一XMT-A文件2220及一組媒體資料檔案2230。輸入的動畫專家群第四階中間檔案2210也可稱之為「mp4二進制檔案」或「mp4檔案」。輸出的XMT-A文件可由基於XMT-A規格說明ISO/IEC 14496-1:2000 Amd.2之本文檔案，或一組代表此類檔案之資料結構組成。輸出的媒體資料檔案2230代表mp4檔案2210中包含的聲頻、視頻及影像資料。各輸出媒體資料檔案由輸出的XMT-A文件2220中所含的StreamSource參考670識別。媒體資料檔案2230數可為零。

由本發明執行的邏輯操作2200可實施為(1)在一電腦系統上運行的電腦執行步驟序列；及/或(2)計算系統中的互連機器模組。該實施方案係根據應用本發明之系統的性能需求而選擇。因此，本文說明的構成本發明具體實施例的邏輯操作可替代性稱之為操作、步驟或模組。

而且，本發明執行的操作可為電腦可讀取程式，其具體化為電腦可讀取媒體。藉由範例說明，而不是限制，電腦可讀取媒體可包含電腦儲存媒體與通訊媒體。電腦儲存媒體包括以任何方法或技術實施的揮發性與非揮發性、可移與不可移式媒體，用於儲存諸如電腦可讀取指令、資料結構、程式模組或其他資料之資訊。電腦儲存媒體包括，但不限於，RAM、ROM、EEPROM、快閃記憶體或其他記憶體技術、CD-ROM，數位多功能光碟(DVD)或其他光學儲存器、磁性卡帶、磁帶、磁碟儲存器或其他磁性儲存元件，

或任何其他可用來儲存所需的資訊且可由電腦存取的媒體。通訊媒體通常具體化電腦可讀取指令、資料結構、程式模組或調變資料信號中的其他資料，例如載波或其他運輸機制，並包括任何資訊傳遞媒體。術語「調變資料信號」意謂一信號，其具有一或多個特徵集、或以此類方式變化以便編碼該信號中之資訊。藉由範例說明，而不是限制，通訊媒體包括有線媒體，如有線網路或直接線路連接，以及無線媒體，如聲音、射頻(RF)、紅外線與其他無線媒體。上述任意組合亦包含於電腦可讀取媒體的範疇中。

如圖22所示，以兩步驟完成建立輸出的XMT-A文件2220及媒體資料檔案2230之過程。在第一步驟，一mp4檔案至中間文件轉換器2240解譯一輸入mp4檔案2210，並建立一組中間文件2245與一組媒體資料檔案2230。該組中間文件2245由一mp4-file文件2250及一mp4-bifs文件2260組成。在第二步驟，根據中間文件2245，中間文件至XMT-A轉換器2270產生輸出的XMT-A 2230。

將該過程2200分成此等兩步驟的一原因為，輸入的mp4檔案2210代表的資訊與輸出的XMT-A文件2220及媒體檔案2230所代表者相同時，輸入mp4檔案2210的組織結構與輸出XMT-A文件的組織結構明顯不同。例如，mp4檔案2210的結構與Quicktime媒體資料檔案的結構密切相關，但XMT-A文件2220的結構卻無Quicktime媒體資料檔案的特徵。XMT-A文件2220包含對場景描述資料流(sdsm)與物件描述符資料流(odsm)之描述，但其可以任何順序混合在一起。mp4檔案

2210也包含對sdsm與odsm的描述，但各代表一個別的時序樣本流。

由於mp4檔案2210的結構組織與XMT-A文件2220的結構組織如此不同，因而，將根據mp4檔案2210建立XMT-A文件2220的過程，分為兩步驟(a)解碼及(b)重組較為有利。在第一步驟，解譯mp4檔案中編碼的二進制資訊，並用於建立一組結構文件，其代表mp4檔案、sdsm及odsm之結構與組織。其他任何二進制資料僅複製為新媒體資料檔案。執行該步驟可不必考慮XMT-A文件之定義。在第二步驟，用第一步驟中建立的結構文件建立一XMT-A文件。執行該步驟可不必考慮mp4檔案中二進制形式所代表的該資訊之方式。

將根據mp4檔案2210建立XMT-A文件2220的過程分為此等兩步驟2240、2270，為了實現該目的，必需定義新結構文件2245，其代表(a)mp4檔案之結構組織；(b)mp4檔案中所代表的場景描述資料流(sdsm)之結構組織；及(c)mp4檔案中所代表的物件描述符資料流(odsm)之結構組織。可藉由定義三類新結構文件來實現，一類代表mp4檔案，一類代表sdsm，還有一類代表odsm。其中，代表mp4檔案與sdsm所需的結構文件相當複雜，而代表odsm所需的結構文件非常簡單。因此，便於將對odsm之描述併入用於代表mp4檔案的結構文件中。因而，在本發明該項具體實施例中，引入兩個新結構文件，一用於mp4檔案與odsm，一用於sdsm。此等兩結構文件識別為mp4-file文件2250及mp4-bifs文件2260。其統稱為中間文件2245。

應注意，選擇使用兩類結構文件已成為便捷性問題。藉由定義三類結構文件(僅mp4-檔案、僅odsm及僅sdsm)可實現相同的目的，或可將三類資訊合併於一單一複合結構文件中。

在本發明一項具體實施例中，為代表重組資訊而建立的特定類型結構文件，係基於「XML」(可擴充標示語言)技術。其優勢係因以下因素：

(a)對XMT-A文件之定義係基於XML技術；

(b)XML技術提供代表結構文件的標準化裝置；及

(c)存在標準化軟體工具，用以與基於XML技術的結構文件合作。

因此，在本發明一項具體實施例中，將重組XMT-A檔案中包含之資訊的過程，減少為一XML至XML變換。此外，存在與XML檔案合作之標準化軟體，使之可管理輸入的XMT-A文件以及中間文件，而無需開發新專屬軟體來執行相同的功能。

儘管該具體實施例係以使用XML技術代表中間文件為基礎，但也可建立採用其他類型結構文件的本發明之替代具體實施例。

以下材料說明(1)mp4-file文件2250之結構；(2)mp4-bifs文件2260之結構；(3)XMT-A至中間文件轉換器2240之操作；及(4)中間文件至mp4檔案轉換器2270之操作。

5.0 mp4-file文件

如圖23A所示，mp4-file文件2300之結構非常類似於mp4

二進制檔案700之結構。mp4-file文件2300包含一組一或多個媒體資料(mdat)元件2310及一單一moov元件2320。該moov元件2320包括一mp4fiods(mp4檔案初始物件描述符)元件2330及一或多個trak元件2350。moov元件2320也可包括一可選的使用者資料(udta)元件2340。該udta元件2340可用於包括版權通告之類的資訊。mvhd原子772之性質由moov元件2320之屬性代表。

如圖23B所示，mp4fiods元件2360擁有一objectDescriptorID屬性2370。mp4fiods元件2360也擁有若干其他圖23B中未顯示之屬性。此等額外屬性包括布林屬性「includeInlineProfilesFlag」與整數屬性「sceneProfileLevelIndication」、「ODProfileLevelIndication」、「audioProfileLevelIndication」、「visualProfileLevelIndication」及「graphicsProfileLevelIndication」。mp4fiods元件2360也包括一或多個EsIdInc元件2380。各EsIdInc元件2380擁有一trackID屬性2390，其匹配相關的trak元件2340的trackID屬性。

如圖24A所示，各mdat元件2400可包含一或多個下列元件：sdsm元件2410、odsm元件2420及mediaFile元件2430。此等元件各擁有一唯一「trackID」屬性，其匹配相關的trak元件2340的trackID屬性。各mediaFile元件2430具有一「名稱」屬性，其指定一外部二進制檔案的檔案名稱，該檔案包含相關的媒體資料(聲頻資料、視覺資料等)。各sdsm元件2410具有一「xmlFile」屬性，其指定代表相關mp4-bifs文件2260的XML檔案名稱。在一項具體實施例中，建立代表

mp4-file文件及/或mp4-bifs文件的XML檔案有益於診斷，但此類檔案並非本發明操作所需。

如圖24B與24D所示，各sdsm元件2440與各mediaFile元件2480皆包含一或多個大塊元件2450、2490。各大塊元件2450、2490擁有一「大小」屬性，其指示相關二進制資料中的位元組數(若知道)。各大塊元件2450、2490還擁有一「偏移」屬性，其指示二進制sdsm資料或媒體資料檔案之起端，與二進制sdsm資料或媒體資料檔案中目前大塊資料的起端之間的位元組數(若知道)。描述場景描述資料流(sdsm)的附加資訊包含在mp4-bifs文件中。

如圖24C所示，各odsm元件2460包含一或多個odsmChunk元件2470。各odsmChunk元件2470擁有一「大小」屬性，其指示物件描述符資料流之相關部分中的位元組數(若知道)。各odsmChunk元件2470還擁有一「偏移」屬性，其指示相關物件描述符資料流之二進制資料的起端，與該流中目前大塊資料的起端之間的位元組數。

如圖25A所示，各odsmChunk元件2500包含一或多個odsmSample元件2510。如圖25B所示，各odsmSample元件2520包含一或多個odsm命令元件2530。如圖25C所示，各odsm命令元件可為一ObjectDescrUpdate元件2540或一ObjectDescrRemove元件2570。各ObjectDescrUpdate元件2540包含一ObjectDescriptor元件2550，而包含於ObjectDescrUpdate元件2540中的ObjectDescriptor元件2550又包含一EsIdRef元件2560。

各odsmSample元件2510、2520擁有一「時間」屬性，其指定以秒表示之時間，此時將要執行odsmSample元件2510、2520中所包含的命令。各ObjectDescriptor元件2550與各ObjectDescrRemove元件2570擁有一「ODID」屬性2555、2575，其指定數字式物件描述符ID。各EsIdRef元件2560擁有一「EsId」屬性2565，其指定數字式基本資料流ID。

如圖26A所示，trak元件2350、2600之結構與mp4檔案700中的trak原子790、900之結構非常類似。各trak元件2600包含一mdia元件2604。trak元件2600也可包含一tref(磁軌參考)元件2636及/或一edts(編碼列表)元件2644。但無tkhd元件可與mp4檔案中的tkhd原子910類比。而是tkhd原子910中所包含的性質表現為trak元件2600之屬性。

mdia元件2604包含一hdlr元件2608與一minf元件2612。mdhd原子915的性質表現為mdia元件2604的屬性。minf元件2612包含一dinf元件2616、一stbl元件2628及一媒體標頭元件2632。媒體標頭元件(「*mhd」)2632可具有依據相關資料流中資料類型的若干形式之一。在與sdsm或odsm相關之trak元件中的媒體標頭元件2632由一「nmhd」元件代表。在聲頻資料流之相關trak元件中的媒體標頭元件2632由一「smhd」元件代表，而在視覺資料流之相關trak元件中的媒體標頭元件2632由一「vmhd」元件代表。

如圖26B所示，一stbl(樣本表)元件2628、2652包含一stsc(樣本至大塊表)元件2656、一stts(時間至樣本表)元件2660、一stco(大塊偏移表)元件2664、一stsz(樣本大小表)

元件2668及一stsd(樣本描述表)元件2676。根據資料流或媒體類型，stbl元件2652也可包括一stss(同步樣本表)元件2672。stsd元件2676可包含若干類型的附屬元件之一，其表示為圖26B中的「mp4*元件」2680。在stsd元件2676包含於sdsm或odsm資料流之相關trak元件2600的情形中，stsd元件2680包含一「mp4s」元件。在stsd元件2680包含於聲頻資料流之相關trak元件2600的情形中，stsd元件2680包含一「mp4a」元件。在stsd元件2680包含於視覺資料流之相關trak元件2600的情形中，stsd元件2680包含一「mp4v」元件。在各情形中，「mp4*」元件2680包含一esds元件2684，而esds元件2684包含一ES_Descr元件2688。

如圖27所示，ES_Descr元件2700包含一DecoderConfigDescriptor元件2710及一SLConfigDescriptor元件2760。DecoderConfigDescriptor元件2710可包含若干類型的解碼器特定資訊元件之一，包括一BIFS_DecoderConfig元件2720、JPEG_DecoderConfig 2730、VisualConfig 2740或AudioConfig 2750。各種類型的解碼器特定資訊元件各代表二進制DecoderConfigDescriptor結構1032中所包含的一形式之DecoderSpecificInfo資料結構1072。二進制ES_Descr結構1000、DecoderConfigDescriptor結構1032、SLConfigDescriptor結構1088及DecoderSpecificInfo結構1076的性質，由mp4file文件2300的對應元件2700、2710、2760、2720、2730、2740、2750之屬性代表。

6.0 mp4-bifs文件

如圖28A所示，mp4-bifs文件2800包含一單一bifsConfig元件2810，其後跟隨一或多個commandFrame元件2820之序列。如圖28B所示，各commandFrame元件2830包含一或多個mp4bifsbifsCommand元件2840。各commandFrame元件2820、2830擁有一屬性「時間」，其指定以秒表示之時間，此時將要執行commandFrame元件中所包含的命令。

各mp4bifsbifsCommand元件2840代表十一個可能的動畫專家群第四階BIFS命令之一：InsertNode、InsertIndexed Value、InsertRoute、DeleteNode、DeleteIndexed Value、DeleteRoute、ReplaceNode、ReplaceField、ReplaceIndexed Value、ReplaceRoute及Replacescene。如圖29A所示，mp4bifsbifsCommand元件2910可包含一或多個mp4bifs Node元件2920。在十一類型的bifsCommand元件中，InsertNode、InsertIndexed Value、ReplaceNode、ReplaceField、ReplaceIndexed Value及ReplaceScene可包含附屬的mp4bifs Node元件2920。InsertIndexed Value、ReplaceIndexed Value及ReplaceField bifsCommand元件可具有附屬的bifsCommand元件2910。

如圖29B所示，ReplaceScene bifsCommand元件2930可僅包括一單一的附屬mp4bifs Node元件，且其必須為一「TopNode」元件2940。TopNode元件2940對應於動畫專家群第四階BIFS節點之一特定子集中的一成員。該子集於動畫專家群第四階系統規格中定義。此外，ReplaceScene bifsCommand元件2930也可包括一附屬「Routes」元件2950，

而該「Routes」元件2950可包含一或多個附屬「Route」元件2960。mp4bifs Route元件2960具有屬性「routeId」、「arrivalNodeId」、「arrivalField」、「departureNodeId」及「departureField」。

除了可能的附屬mp4bifs Node元件，各類mp4bifs bifsCommand元件擁有下列屬性值：

1.InsertNode：「parentId」、「insertionPosition」及「position」。

2.InsertIndexed Value：「nodeId」、「inFieldName」、「insertionPosition」、「position」及「value」。

3.InsertRoute：「RouteId」、「departureNode」、「departureField」、「arrivalNode」及「arrivalField」。

4.DeleteNode：「nodeId」。

5.DeleteIndexed Value：「nodeId」、「inFieldName」、「deletionPosition」及「position」。

6.DeleteRoute：「routeId」。

7.ReplaceNode：「parentId」。

8.ReplaceField：「nodeId」、「inFieldName」及「value」。

9.ReplaceIndexed Value：「nodeId」、「inFieldName」、「insertionPosition」、「position」及「value」。

10.ReplaceRoute：「routeId」、「departureNode」、「departureField」、「arrivalNode」及「arrivalField」。

11.ReplaceScene：「USENAMES」(一布林值)。

對於bifsCommand元件InsertIndexed Value、ReplaceField

及ReplaceIndexed Value，若「inFieldName」屬性所指定的性質欄位具有一節點資料類型「Node」(根據動畫專家群第四階規格說明)，則該元件將包含一或多個附屬mp4bifs節點元件2920，且該「value」屬性將包含與各附屬的節點元件相關的節點名稱列表。

mp4bifs Node元件2920代表許多類型的動畫專家群第四階BIFS節點資料結構之一。動畫專家群第四階系統規格中定義了100種以上不同類型的BIFS節點。各類動畫專家群第四階BIFS節點具有一特定的NodeName與一組性質欄位。

有兩種基本類型的mp4bifs節點元件：初始節點元件與再用節點元件。如圖30A所示，mp4bifs初始節點元件3000由一「NodeName」識別，該「NodeName」對應於動畫專家群第四階系統規格中定義的BIFS節點之一的NodeName性質。

mp4bifs初始節點元件3000可具有一可選的NodeId屬性3010。若對於NodeId屬性3010指定一值，則節點元件3000歸類為一「可再用節點」。NodeId屬性3010的值(若指定)係一整數，其在1至目前場景中所定義之可再用節點數之間的範圍內。若已對NodeId屬性3010指定一值，且相關的ReplaceScene命令之「USENAMES」屬性值為「真」，則節點元件將也具有一「名稱」屬性3016。

除了NodeId 3010與名稱3016屬性之外，各初始節點元件具有數個性質欄位屬性3020。對於一特定節點元件的NodeName所識別的節點類型，各性質欄位屬性3020對應於動畫專家群第四階系統規格中所定義的性質欄位之一。各

性質欄位具有一定義的欄位資料類型，如布林、整數、浮點等。該組可能的欄位資料類型包括「SFNode」與「MFNode」。若一特定初始節點元件的NodeName對應於一動畫專家群第四階BIFS節點，其具有一性質欄位或欄位資料類型為「SFNode」與「MFNode」之欄位，則該節點元件可擁有一或多個附屬節點元件3030。若如此，對應的性質欄位屬性之值，由該性質欄位相關的各附屬節點元件之NodeName串組成。

例如，若具有NodeName「Group」的一特定mp4bifs節點元件擁有一附屬mp4bifs節點元件，其具有與該「children」屬性相關的NodeName「Transform2D」、「Valuator」及「TimeSensor」，則「children」屬性值將為「Transform2D Valuator TimeSensor」。

在Conditional BIFS節點的特定情形中，性質欄位之一具有性質欄位名稱「緩衝」，「緩衝」性質欄位的欄位資料類型為「命令緩衝」，「緩衝」性質欄位之值由一或多個BIFS命令組成。在此情形中，對應的mp4bifs節點元件3040之NodeName為「Conditional」。對於Conditional節點元件3040，NodeId屬性3050與名稱屬性3056之值可指定為其他任何mp4bifs初始節點元件3000。Conditional節點元件不具有附屬節點元件3030，而擁有一或多個附屬bifsCommand元件3070，且「緩衝」屬性值由附屬的bifsCommand元件3070的命令名稱之有序列表組成。

例如，若一特定Conditional節點元件擁有一附屬

InsertRoute bifsCommand元件，其後為一附屬DeleteNode bifsCommand元件，則「緩衝」屬性值將為「InsertRoute DeleteNode」。

初始節點元件擁有附屬節點元件或bifsCommand元件的能力，可以遞迴方式對階層式BIFS命令與節點元件集合重複。

如圖30C所示，一再用節點元件3080具有一NodeName「ReusedNode」。ReusedNode元件3080無附屬元件。ReusedNode元件3080具有名為「nodeRef」3090的單一屬性。對於可再用的初始節點元件3000、3040之一，nodeRef屬性3090的值必須匹配NodeId屬性3010、3050之值。

7.0 mp4至xmta轉換器

如圖22所示，根據一動畫專家群第四階中間二進制檔案(「mp4檔案」)2210，過程2000建立一XMT-A文件2220及一組零或多個二進制媒體資料檔案2230。

該過程主要包括兩個步驟：

a.根據一mp4二進制檔案2210，建立2240一對中間xml文件2250、2260及一組媒體資料檔案2230。

b.根據該對中間xml文件2250、2260，建立2270一xmta xml文件2220。

此等各步驟說明如下。

根據一mp4二進制檔案，建立一對中間xml文件及一組媒體資料檔案

根據一mp4二進制檔案2210，建立一對中間xml文件2250、

2260及一組媒體資料檔案2230的過程2240包括兩部分：

(1)根據一mp4二進制檔案2210的內容，建立一QtInfo資料結構3100。

(2)根據最終的QtInfo資料結構3100，建立一組媒體資料檔案2230及一對中間xml文件2250、2260。該組媒體資料檔案2230可為空。

在本發明的一項替代性具體實施例中，此等兩部分可組合為一單一過程，其中中間xml文件2250、2260與媒體資料檔案2230直接自mp4二進制檔案2210中建立，而無需建立一QtInfo資料結構3100。如圖7A所示，mp4二進制檔案700由一moov原子712與一或多個mdat原子706組成。若無moov原子712中指定的資訊，就不能解譯mdat原子706，其包含建立媒體資料檔案2230所需之資料。因此，建立一臨時資料結構較為有利，如QtInfo資料結構3100，其在建立中間xml檔案2250、2260與媒體資料檔案2230之前，代表mp4二進制檔案2210的內容。

QtInfo資料結構

QtInfo資料結構代表一mp4二進制檔案700的內容。如圖31A所示，QtInfo資料結構3100由以下組件組成：

1. maxNumTracks 3103：一整數，其指定mdat陣列3113與trak陣列3120中的項目數；

2. Mp4fInitObjectDescr 3106：Mp4fInitObjectDescr資料結構3130的位址，該結構代表該QtInfo資料結構3100所代表的mp4二進制檔案700中，包含的moov原子712、754中所包含

的iods原子778之性質；

3. mediaDataSize 3108：一整數，其指定該QtInfo資料結構3100所代表的mp4二進制檔案700中，所有mdat原子706、718之大小724的總和；

4. numMdatAtoms 3110：一整數，其指定該QtInfo資料結構3100所代表的mp4二進制檔案700中，包含的mdat原子706之數目；

5. mdat陣列3113：一maxNumTracks 3103mdat結構3160之序列的位址。或者，其可指定mdat結構3160之maxNumTracks 3103位址序列的位址。該陣列中的各個第一numMdatAtomsmdat結構，代表該QtInfo資料結構3100所代表的mp4二進制檔案700中，包含的mdat原子706之一；

6. MoovHeader 3116：MoovHeader資料結構3170的位址，該結構代表該QtInfo資料結構3100所代表的mp4二進制檔案700中，包含的moov原子712、754中所包含的mvhd原子772之性質；

7. numTracks 3118：一整數，其指定該QtInfo資料結構3100所代表的mp4二進制檔案700中，trak原子790之數目；

8. trak陣列3120：一maxNumTracks 3103 trak結構3200之序列的位址。或者，其可指定trak結構3200之maxNumTracks 3103位址序列的位址。該陣列中的各個第一numTrakAtoms trak結構，代表該QtInfo資料結構3100所代表的mp4二進制檔案700中，包含的moov原子712、754中所包含的trak原子790之一；

9. (UserData)：一可選資料結構的位址，該結構代表該QtInfo資料結構3100所代表的mp4二進制檔案700中，moov原子712之外一可能的udta原子；

Mp4fInitObjectDescr資料結構3130代表iods原子778、800之性質。如圖31B所示，Mp4fInitObjectDescr資料結構3130由以下組件組成：

1. size 3133：一整數，其指示iods原子778中的位元組數；

2. version 3136：一整數，其指示iods原子800的version(版本)812與flags(旗標)816性質之值；

3. ObjectDescrID 3138：一整數，其指示iods原子800中所包含的Mp4fInitObjectDescr物件824之ObjectDescriptorID性質836的值；

4. urlFlag 3140：一布林值，其指示iods原子800中所包含的Mp4fInitObjectDescr物件824之URLFlag性質840的值；

5. urlString 3143：若數量「urlFlag」之值為「假」，則數量「urlstring」之值為零(空)。否則，該數量之值指定代表一字元串之空終止序列位元組的位址。

6. inLineProfiles 3146：一布林值，其代表iods原子800中所包含的Mp4fInitObjectDescr物件824之includeInlineProfilesFlag性質844的值；

7. ProfileLevels 3148：一組五個整數，即ODProfileLevel、sceneProfileLevel、audioProfileLevel、visualProfileLevel及graphicsProfileLevel，其指示iods原子800中所包含的Mp4fInitObjectDescr物件824之性質ODProfileLevel

Indication 852、sceneProfileLevelIndication 856、audioProfileLevelIndication 860、visualProfileLevelIndication 864及graphicsProfileLevelIndication 868的值；

8. maxNumEsIdIncs 3153：一整數，其指示EsIdInc陣列3156中的項目數(值為「2」即足夠)；

9. numEsIdIncs 3153：一整數，其指示iods原子800中所包含的Mp4fInitObjectDescr物件824之ES_ID_Inc物件872的數目；

10. EsIdInc陣列3156：一maxNumEsIdIncs EsIdInc結構序列，或一maxNumEsIdIncs EsIdInc結構序列之位址。

若URL_Flag性質840之值為「0」，則將值「假」指派給組件「urlFlag」3140，否則，將值「真」指派給數量「urlFlag」3140。URL_Flag性質840之預期值為「0」。

若includeInlineProfilesFlag性質844之值為「0」，則將值「假」指派給組件「inLineProfiles」3146，否則，將值「真」指派給數量「inLineProfiles」3146。

一EsIdInc結構由兩整數組成，「size」與「trackID」。此等整數代表一Mp4fInitObjectDescr物件824中，對應的ES_ID_Inc物件872、876中的數量「numBytes」884與「ES_ID」888的值。

各mdat結構3160代表一mdat原子706、718的性質。如圖31C所示，mdat結構3160由以下組件組成：

1. size 3163：包括mdat原子706的位元組數。此與mdat原子718的「原子大小」性質724相同。

2. start 3166：其指定mp4二進制檔案700起端與mdat原子706的第一位元組之間的位元組數。

3. MediaData 3168：其為含有mdat原子706中所包含的該組大塊736之副本的位元組(size-8)陣列的位址。

MoovHeader結構3170代表mvhd原子772的性質。如圖31D所示，MoovHeader結構3170由以下值組成：

1. size 3173：一整數，其指示mvhd原子772中的位元組數；

2. version & Flags 3176：一對整數，其指示mvhd原子772的「version」與「flags」性質之值；

3. creationTime 3178：一整數，其指示mvhd原子772的「creationTime」性質之值；

4. modifiedTime 3180：一整數，其指示mvhd原子772的「modificationTime」性質之值；

5. timeScale 3183：一整數，其指示mvhd原子772的「timescale」性質之值；

6. duration 3186：一整數，其指示mvhd原子772的「duration」性質之值；

7. nextTrackID 3188：一整數，其指示mvhd原子772的「nextTrackID」性質之值；

各trak結構3200代表一trak原子790、900的性質。如圖32A所示，各trak結構3200可包含下列組件：

1. size 3202：一整數，其指示trak原子900中的位元組數903；

2. TrackHeader 3204：TrackHeader資料結構3212的位址，

該結構代表該trak結構3200所代表的trak原子900中所包含的tkhd原子910之性質；

3. TrackMedia 3206：TrackMedia資料結構3226的位址，該結構代表該trak結構3200所代表的trak原子900中所包含的mdia原子912之性質；

4. TrackEdit 3208：可選TrackEdit資料結構3236的位址，該結構代表該trak結構3200所代表的trak原子900中所包含的可選edts原子945之性質；

5. TrackReference 3210：可選TrackReference資料結構3264的位址，該結構代表該trak結構3200所代表的trak原子900中所包含的可選tref原子940之性質；

TrackHeader結構3212代表tkhd原子910的性質。如圖32B所示，TrackHeader結構3212由以下組件組成：

1. size 3214：一整數，其指示tkhd原子910中的位元組數；

2. version & Flags 3216：一對整數，其指示tkhd原子910的「version」與「flags」性質之值；

3. creationTime 3218：一整數，其指示tkhd原子910的「creationTime」性質之值；

4. modifiedTime 3220：一整數，其指示tkhd原子910的「modificationTime」性質之值；

5. duration 3222：一整數，其指示tkhd原子910的「duration」性質之值；

6. trackID 3224：一整數，其指示tkhd原子910的「trackID」性質之值；

TrackMedia結構3226代表mdia原子912的性質。如圖32C所示，TrackMedia結構3226由以下組件組成：

1. size 3228：一整數，其指示mdia原子912中的位元組數；

2. MediaHeader 3230：MediaHeader結構3300的位址，該結構代表該mdia原子912中所包含的mdhd原子915之性質；

3. Handler 3232：Handler結構3320的位址，該結構代表該mdia原子912中所包含的hdlr原子918之性質；

4. MediaInfo 3234：MediaInfo結構3333的位址，該結構代表該mdia原子912中所包含的minf原子920；

TrackEdit結構3236代表edts原子945的性質。如圖32D所示，TrackEdit結構3236由以下組件組成：

1. size 3238：一整數，其指示edts原子945中的位元組數；

2. version & Flags 3240：一對整數，其指示edts原子945的「version」與「flags」性質之值；

3. EditList 3242：EditList結構3244之位址，該結構代表edts原子945中所包含的elst原子948。

EditList結構3244代表elst原子948的性質。如圖32E所示，EditList結構3244由以下組件組成：

1. size 3246：一整數，其指示elst原子948中的位元組數；

2. version & Flags 3248：一對整數，其指示elst原子948的「version」與「flags」性質之值；

3. numSegments 3250：一整數，其指定EditSegment陣列3252中的EditSegment結構3256數；

4. EditSegment array 3252：此可為numSegments Edit

Segment結構3256序列，或EditSegment結構3256的numSegments位址序列。

各EditSegment結構3256代表elst原子948中所包含的「編輯區段」性質，該原子由包含該EditSegment結構3256的EditList結構3244代表。如圖32F所示，各EditSegment結構3256由三個整數組成：startTime 3258、duration 3260及relativeRate 3262。此等整數各代表對應編輯區段的類似名稱性質。

TrackReference結構3264代表tref原子940的性質。如圖32G所示，TrackReference結構3264由以下組件組成：

1. size 3266：一整數，其指示tref原子940中的位元組數；

2. Mpod 3268：Mpod結構3270之位址，該結構代表tref原子940中所包含的mpod原子942。

Mpod結構3270代表mpod原子942的性質。如圖32H所示，Mpod結構3270由以下組件組成：

1. size 3272：一整數，其指示mpod原子942中的位元組數；

2. numEntries 3274：一整數，其指定trackID陣列3276中的項目數；

3. trackID陣列3276：numEntries整數序列之位址。此等整數各代表mpod原子942中所指定的trackID值之一。

UserData結構3280代表udta原子784的性質。如圖32I所示，UserData結構3280由以下組件組成：

1. size 3282：一整數，其指示udta原子784中的位元組數；

2. numCprt 3284：一整數，其指定cprt陣列3286中的成員

數；

3. cprt陣列3286：cprt結構3290序列之位址，或cprt結構3290位址之序列。

cprt結構3290代表cprt原子的性質。如圖32J所示，cprt結構3290由以下組件組成：

1. size 3282：一整數，其指示cprt原子中的位元組數；

2. version & Flags 3294：一對整數，其指示cprt原子的「version」與「flags」性質之值；

3. language 3296：一整數，其指示cprt原子的「language」性質之值；

4. notice 3298：一空終止串，其指示cprt原子中指定的訊息。

MediaHeader結構3300代表mdhd原子915的性質。如圖33A所示，MediaHeader結構3300包括以下值：

1. size 3303：一整數，其指示mdhd原子915中的位元組數；

2. version & flag 3306：一對整數，其指示mdhd原子915的「version」與「flags」性質之值；

3. creationTime 3308：一整數，其指示mdhd原子915的「creationTime」性質之值；

4. modifiedTime 3310：一整數，其指示mdhd原子915的「modificationTime」性質之值；

5. timeScale 3313：一整數，其指示mdhd原子915的「timeScale」性質之值；

6. duration 3316：一整數，其指示mdhd原子915的

「duration」性質之值；

7. language 3318：一整數，其指示mdhd原子915的「language」性質之值；

Handler結構3320代表hdlr原子918的性質。如圖33B所示，Handler結構3320包括以下值：

1. size 3323：一整數，其指示hdlr原子918中的位元組數；

2. version & Flags 3326：一對整數，其指示hdlr原子918的「version」與「flags」性質之值；

3. handlerType 3328：一4字元標籤，其指定hdlr原子918的處置器類型性質；及

4. name 3330：提供描述或診斷用途的空終止字串的位址。

MediaInfo結構3333代表minf原子920的性質。如圖33C所示，MediaInfo結構3333包括MediaDataInfo結構3338、3350的位址、SampleTables結構3340、3400的位址及MediaInfoHeader結構3343的位址。MediaInfoHeader結構3343代表以下媒體資訊標頭原子936之一：

1. nmhd原子(對於場景描述資料流sdsm或物件描述符資料流odsm)；

2. vmhd原子(對於視覺資料流)；或

3. smhd原子(對於聲頻資料流)。

MediaInfoHeader結構3343包括下列值：

1. size：一整數，其指示媒體資訊標頭原子936中的位元組數；

2. version & Flags：一對整數，其指示媒體資訊標頭原子936的「version」與「flags」性質之值。

MediaDataInfo結構3350代表dinf原子924的性質。如圖33D所示，MediaDataInfo結構3350包括以下組件：

1. size 3353：一整數，其指示dinf原子924中的位元組數；

2. version & Flags 3356：一對整數，其指示dinf原子924的「version」與「flags」性質之值；

3. DataRefTable 3358：DataRefTable資料結構3360的位址。

DataRefTable結構3360代表dref原子927的性質。如圖33E所示，DataRefTable結構3360包括以下組件：

1. size 3363：一整數，其指示dref原子927中的位元組數；

2. version & Flags 3366：一對整數，其指示dref原子927的「version」與「flags」性質之值；

3. numDataRefs 3368：一整數，其指示dataRef陣列3370中的項目數；及

4. dataRef陣列3370：dataRef資料結構3380序列之位址。

dataRef結構3380代表資料參考表中的一項目。如圖33F所示，dataRef結構3380包括以下組件：

1. size 3383：一整數，其指示該dataRef結構3380所代表的資料參考結構中的位元組數；

2. version & Flags 3386：一對整數，其指示資料參考結構的「version」與「flags」性質之值；

3. dataRefType 3388：一四字元序列，其指示該dataRef結構3380所代表的資料參考結構類型；及

4. dataRefData 3390：代表剩餘資料參考結構的size-12位元組序列之位址。

SampleTables結構3400代表一stbl原子933、950的性質。如圖34A所示，SampleTables結構3400包括以下組件：

1. size 3403：一整數，其指示stbl原子950中的位元組數954；

2. SampleToChunkTable 3406：代表一stsc原子960內容之資料結構的位址；

3. ChunkOffsetTable 3408：代表一stco原子966內容之資料結構的位址；

4. TimeToSampleTable 3410：代表一stts原子963內容之資料結構的位址；

5. SampleSizeTable 3413：代表一stsz原子970內容之資料結構的位址；

6. SyncSampleTable 3416：代表一stss原子974內容之可選資料結構的位址；

7. SampleDescriptionTable 3418：代表一stsd原子978內容之資料結構的位址。

除了SampleSizeTable 3413，SampleTables結構3400中包含的各樣本表具有下列成員，如圖34B中3420所示：

1. size 3423：一整數，其指示該資料結構所代表的樣本表原子中的位元組數；

2. version & Flags 3426：一對整數，其指示該資料結構所代表的樣本表原子的「version」與「flags」性質之值；

3. numEntries 3428：一整數，其指示table array(表陣列)3430中的項目數；及

4. table array 3430：整數陣列或資料結構的位址。

如圖34C所示，SampleSizeTable結構3440與其他樣本表類似，只是其包括一額外成員「sampleSize」3450，即指定所有樣本之固定樣本大小的整數。若成員「sampleSize」3450的值為非零，則省略成員「table array」3456。

成員「table array」的結構取決於樣本表的類型。在ChunkOffsetTable結構3408、SampleSizeTable結構3413及SyncSampleTable結構3416的情形中，該成員代表32位元整數序列。

在SampleToChunkTable結構3406的情形中，表陣列代表SampleToChunk資料結構3460之陣列。如圖34D所示，各SampleToChunk資料結構3460由三整數成員組成，即「firstChunk」3463、「numSamples」3466及「description」3468。

在TimeToSampleTable結構3410的情形中，表陣列代表TimeToSample資料結構3470之陣列。如圖34E所示，各TimeToSample資料結構3470由兩整數成員組成，即「count」3473與「duration」3476。

在SampleDescriptionTable結構3418的情形中，表陣列代表SampleDescription資料結構3480之陣列。如圖34F所示，各SampleDescription資料結構3480由以下六成員組成：

1. size 3483：一整數，其指示該SampleDescription結構

3480所代表的樣本描述表項目中的位元組數；

2. dataFormat 3486：一四字元識別項(「mp4s」、「mp4a」或「mp4v」)，其指定該SampleDescription結構3480所代表的mp4*原子982的類型；

3. dataReference 3488：一整數，其代表該Sample Description結構3480所代表的mp4*原子982的dataReference性質；

4. esdVersion 3490：一整數，其指示該SampleDescription結構3480所代表的mp4*原子982中，包含的「esds」原子986的版本性質；

5. esdSize 3493：一整數，其指示該SampleDescription結構3480所代表的mp4*原子982中，包含的「esds」原子986的位元組數；

6. esdData 3496：包含ES_Descr物件結構990之副本的位元組陣列之位址，該結構包含於該SampleDescription結構3480所代表的mp4*原子982所包含的「esds」原子986中。

根據mp4二進制檔案建立QtInfo結構

根據mp4二進制檔案建立QtInfo資料結構的過程，以基於數量「maxNumTracks」的特定值，新建立空QtInfo資料結構開始。數量「maxNumTracks」的值可為預定常數，如128，其足夠大，可超過特定mp4二進制檔案中的實際磁軌數，或可藉由打開mp4二進制檔案，計算該檔案中包含的trak原子數來決定。新建立空QtInfo資料結構後，對mp4檔案進行解譯，如圖35所示。該過程包括以下步驟：

1. 操作3500，打開mp4二進制檔案(以讀取)。

2. 操作3510，自mp4檔案中讀取一32位元整數，並將該整數值指派給數量「atomSize」。然後自該mp4檔案中讀取第二32位元整數，並將該整數值指派給數量「atomId」。

3. 操作3520，比較數量「atomId」之值與「mdat」值。

4. 操作3525，若數量「atomId」之值對應於「mdat」，則執行程序「處理mdat原子」。

5. 操作3530，比較數量「atomId」之值與「moov」值。

6. 操作3535，若數量「atomId」之值對應於「moov」，則執行程序「處理moov原子」。

7. 操作3540，比較數量「atomId」之值與「udta」值。

8. 操作3545，若數量「atomId」之值對應於「udta」，則執行程序「處理udta原子」，之後為下述操作3570。

9. 否則，操作3560，跨越mp4檔案內的atomSize-8位元組。即，自該mp4檔案中讀取該位元組數，但不用所得值進行任何操作。

10. 操作3570，測試mp4二進制檔案700的目前檔案位置。

11. 若mp4二進制檔案700的目前檔案位置對應於檔案結束條件(讀取的位元組數等於完整mp4檔案的位元組數)，則操作3580完成本過程。否則，重複操作3510至操作3570。

如圖36所示，程序「處理mdat原子」操作3530包括以下步驟：

1. 操作3600，建立新MediaData結構「newMediaData」3160。

2. 操作3610，將該新MediaData結構3160的位址，指派給QtInfo結構3100中mdat陣列3113的項目numMdatAtoms。

3. 操作3620，將numMdatAtoms 3110的值遞增一。

4. 操作3630，將數量「atomSize」的值指派給新MediaData結構3160中的成員「size」3163。

5. 操作3640，將目前檔案指標的值(-8)指派給新MediaData結構3160中的成員「start」3166。

6. 操作3650，建立具有(atomSize-8)位元組的新位元組陣列，並將該陣列的位址指派給新MediaData結構3160中的成員「MediaData」3168。

7. 操作3660，自該mp4檔案中讀取(atomSize-8)位元組，並將結果複製於新位元組陣列中，該陣列由新MediaData結構3160中的成員「MediaData」3168指定。

如圖37所示，程序「處理moov原子」操作3550由以下步驟組成：

1. 操作3700，將零值指派給QtInfo結構3100中的成員「numTracks」3119。

2. 操作3710，自mp4檔案中讀取一32位元整數，並將該整數值指派給數量「atomSize」。然後自該mp4檔案中讀取第二32位元整數，並將該整數值指派給數量「atomId」。

3. 操作3720，比較數量「atomId」之值與「mvhd」值。

4. 操作3725，若數量「atomId」之值對應於「mvhd」，則執行程序「處理mvhd原子」。然後，以下述操作3770繼續。

5. 操作3730，比較數量「atomId」之值與「iods」值。

4. 將該值「0」指派給UserData結構3280的成員「numCprt」3284。

5. 重複程序「讀取版權原子」，直至到達udta原子784的末端。自mp4檔案從udta原子784之起端，讀取的資料位元組數等於成員「size」3282的值時，即滿足到達udta原子784之末端的條件。

程序「讀取版權原子」由以下步驟組成：

1. 自mp4檔案中讀取一32位元整數，並將該整數值指派給數量「atomSize」。然後自該mp4檔案中讀取第二32位元整數，並將該整數值指派給數量「atomId」。

2. 比較數量「atomId」之值與「cprt」值。

3. 若數量「atomId」之值不對應於「cprt」，則跨越mp4檔案中的atomSize-8位元組資料，並以程序「處理使用者資料原子」之步驟5繼續。否則，執行下列步驟。

4. 建立一新cprt結構「newCprt」3290。

5. 將該新cprt結構的位址，指派給UserData結構3280之cprt陣列成員3286中的項目numCprt。

6. 將數量「atomSize」之值指派給該新cprt結構3290的成員「size」(3292)。

7. 自mp4檔案中讀取一32位元整數，並將該整數值指派給新cprt結構3290的成員「version & Flags」3294。

8. 自mp4檔案中讀取一16位元整數，並將該整數值指派給新cprt結構3290的成員「language」3296。

9. 自mp4檔案中讀取一空終止位元組序列，並將此等位

元組複製至一新字串數量「newNotice」。

10. 將新字串數量「newNotice」之位址，指派給該新cprt結構3290的成員「notice」3298。

11. 將UserData結構3280的成員numCprt 3284之值遞增1。

程序「處理mvhd原子」操作3725由以下步驟組成：

1. 建立一新MoovHeader資料結構3170。

2. 將該新MoovHeader資料結構3170的位址，指派給QtInfo結構3100中的成員MoovHeader 3116。

3. 將數量「atomSize」之值指派給該新MoovHeader資料結構3170中的成員「size」3173。

4. 自mp4檔案中讀取一32位元整數，並將結果指派給MoovHeader結構3170中的成員「version & Flags」3176。

5. 自mp4檔案中讀取一32位元整數，並將結果指派給MoovHeader結構3170中的成員「creationTime」3178。

6. 自mp4檔案中讀取一32位元整數，並將結果指派給MoovHeader結構3170中的成員「modificationTime」3180。

7. 自mp4檔案中讀取一32位元整數，並將結果指派給MoovHeader結構3170中的成員「timeScale」3183。

8. 自mp4檔案中讀取一32位元整數，並將結果指派給MoovHeader結構3170中的成員「duration」3186。

9. 讀取mp4檔案中的76位元組，並忽略所得值。

10. 自mp4檔案中讀取一32位元整數，並將結果指派給MoovHeader結構3170中的成員「nextTrackID」3188。

程序「處理iods原子」由以下步驟組成：

1. 建立一新Mp4fInitObjectDescr資料結構3130。

2. 將該新Mp4fInitObjectDescr資料結構3130的位址，指派給QtInfo結構3100中的成員Mp4fInitObjectDescr 3106。

3. 將「atomSize」之值指派給Mp4fInitObjectDescr結構3130中的成員「size」3133。

4. 自mp4檔案中讀取一32位元整數812、816，並將結果指派給Mp4fInitObjectDescr結構3130中的成員「version」3136。

5. 自mp4檔案中讀取一8位元整數828，並驗證該結果具有預期值「16」(MP4_IOD_Tag)。

6. 自mp4檔案中讀取一8位元整數832，並將結果指派給數量「objSize」。

7. 若objSize之值大於127，則重複步驟6。

8. 自mp4檔案中讀取一16位元整數836至848，並將結果指派給數量「odidandflags」。

9. 將值「odidandflags」836的高階10位元值，指派給Mp4fInitObjectDescr結構3130中的成員「ObjectDescriptor ID」3138。

10. 若邏輯操作的結果((「odidandflags」之值)與32)為零，則將值「0」指派給Mp4fInitObjectDescr結構3130中的成員「urlFlag」3140。否則，將值「1」指派給Mp4fInitObjectDescr結構3130中的成員「urlFlag」3140。

11. 若邏輯操作的結果((「odidandflags」之值)與16)為零，則將值「0」指派給Mp4fInitObjectDescr結構3130中的成員

「inlineProfiles」3146。否則，將值「1」指派給Mp4fInitObjectDescr結構3130中的成員「inlineProfiles」3146。

12. 將值「2」指派給Mp4fInitObjectDescr結構3130中的成員「maxNumEsIdIncs」3150。

13. 將零值指派給Mp4fInitObjectDescr結構3130中的成員numEsIdIncs 3153。

14. 建立一maxNumEsIdIncs位址值之陣列。

15. 將該陣列之位址，指派給Mp4fInitObjectDescr結構3130中的成員「EsIdInc array」3156。

16. 若成員urlFlag之值為「1」，則執行程序「處理URL串」。否則(urlFlag之值為「0」)，執行程序「處理設定檔等級指示符」。

程序「處理URL串」由以下步驟組成：

1. 自mp4檔案讀取一空終止位元組序列；

2. 將此等位元組保存於新位元組陣列中；

3. 將該新位元組陣列之位址，指派給Mp4fInitObjectDescr結構3130中的成員urlString 3143。

程序「處理設定檔等級指示符」由以下步驟組成：

1. 將零值指派給Mp4fInitObjectDescr結構3130中的數量urlString 3143。

2. 自mp4檔案中讀取五個設定檔等級指示值852至868，並將各值指派給Mp4fInitObjectDescr結構3130的對應一成員。此等設定檔等級值各由一單一位元組組成，且此等5值

由Mp4fInitObjectDescr結構3130的「ProfileLevels」3146部分代表。

3. 重複程序「處理EsIdInc物件」，直至到達iods原子800的末端。自mp4檔案從iods原子800之起端，讀取的資料位元組數，等於iods原子800的size 804、3133時，即滿足到達iods原子800之末端的條件。

程序「處理EsIdInc物件」由以下步驟組成：

1. 自mp4檔案讀取一位元組物件標籤880。

2. 驗證該物件標籤之值具有預期值(14，「EsIdIncTag」)。

3. 建立一新EsIdInc結構。該結構包含兩成員值「size」與「trackID」。

4. 將該新EsIdInc結構之位址，指派給Mp4fInitObject Descr結構3130中EsIdInc陣列3156中的項目numEsIdIncs。

5. 讀取包括剩餘EsIdInc物件的位元組數884，並將結果指派給新EsIdInc結構之成員「size」。

6. 自mp4檔案888中讀取一32位元整數，並將結果指派給新EsIdInc結構之成員「trackID」。

7. 將成員numEsIdIncs 3153之值遞增一。

如圖38所示，程序「處理trak原子」操作3745由以下步驟組成：

1. 操作3800，建立新trak資料結構3200。將該新trak結構的位址，指派給目前QtInfo結構3100之trak陣列成員3120中的項目numTracks。將數量「AtomSize」之值指派給該新trak結構3200的成員「size」3202。

2. 操作3810，自mp4檔案中讀取一32位元整數，並將該整數值指派給數量「atomSize」。然後自該mp4檔案中讀取第二32位元整數，並將該整數值指派給數量「atomId」。

3. 操作3820，比較數量「atomId」之值與「tkhd」值。

4. 操作3825，若數量「atomId」之值對應於「tkhd」，則執行程序「處理磁軌標頭(tkhd)原子」。然後，以操作3870繼續。

5. 操作3830，比較數量「atomId」之值與「mdia」值。

6. 操作3835，若數量「atomId」之值對應於「mdia」，則執行程序「處理磁軌媒體(mdia)原子」。然後，以操作3870繼續。

7. 操作3840，比較數量「atomId」之值與「edts」值。

8. 操作3845，若數量「atomId」之值對應於「edts」，則執行程序「處理磁軌編輯(edts)原子」。然後，以操作3870繼續。

9. 操作3850，比較數量「atomId」之值與「tref」值。

10. 操作3855，若數量「atomId」之值對應於「tref」，則執行程序「處理磁軌參考(tref)原子」。然後，以操作3870繼續。

11. 否則操作3860，跨越mp4檔案中的atomSize-8位元組。即，自mp4檔案中讀取atomSize-8位元組，並忽略所得值。

12. 操作3870，檢查到達trak原子900末端之情況。自mp4檔案從trak原子900之起端，讀取的資料位元組數等於目前trak結構3200的成員「size」3202之值時，即滿足到達trak

原子900之末端的條件。

13. 若已到達trak原子900之末端，則操作3880將numTracks之值遞增1。否則，重複操作3810至操作3870。

程序「處理磁軌標頭原子」由以下步驟組成：

1. 建立一新TrackHeader資料結構3212。

2. 將該新的TrackHeader資料結構3212的位址，指派給目前trak結構3200中的成員TrackHeader 3204。

3. 將數量「atomSize」之值指派給TrackHeader結構3212中的成員「size」3214。

4. 自mp4檔案中讀取一32位元整數，並將結果指派給TrackHeader結構3212中的成員「version & Flags」3216。

5. 自mp4檔案中讀取一32位元整數，並將結果指派給TrackHeader結構3212中的成員「creationTime」3218。

6. 自mp4檔案中讀取一32位元整數，並將結果指派給TrackHeader結構3212中的成員「modifiedTime」3220。

7. 自mp4檔案中讀取一32位元整數，並將該值指派給數量「trackID」。將數量「trackID」之值指派給TrackHeader結構3212的成員「trackID」3224。

8. 讀取mp4檔案中的4位元組，並忽略所得值。

9. 自mp4檔案中讀取一32位元整數，並將結果指派給TrackHeader結構3212中的成員「duration」3222。

10. 讀取mp4檔案中的60位元組，並忽略所得值。

如圖39所示，程序「處理磁軌媒體原子」由以下步驟組成：

1. 操作3900，建立新TrackMedia資料結構3226。將該TrackMedia結構3226的位址，指派給目前trak結構3200中的成員TrackMedia 3206。將「atomSize」之值指派給新TrackMedia結構3226中的成員「size」3228。

2. 操作3910，自mp4檔案中讀取一32位元整數，並將該整數值指派給數量「atomSize」。然後自該mp4檔案中讀取第二32位元整數，並將該整數值指派給數量「atomId」。

3. 操作3920，比較數量「atomId」之值與「mdhd」值。

4. 操作3925，若數量「atomId」之值對應於「mdhd」，則執行程序「處理媒體標頭(mdhd)原子」。然後，以下述操作3970繼續。

5. 操作3930，比較數量「atomId」之值與「hdlr」值。

6. 操作3935，若數量「atomId」之值對應於「hdlr」，則執行程序「處理處置器(hdlr)原子」。然後，以下述操作3970繼續。

7. 操作3940，比較數量「atomId」之值與「minf」值。

8. 操作3945，若數量「atomId」之值對應於「minf」，則執行程序「處理媒體資訊(minf)原子」。然後，以下述操作3970繼續。

9. 否則，操作3960，跨越mp4檔案中的atomSize-8位元組。即，自該mp4檔案中讀取該位元組數，但不用所得值進行任何操作。

10. 操作3970，檢查到達mdia原子912末端之情況。自mp4檔案從mdia原子912之起端，讀取的資料位元組數等於目前

TrackMedia結構3226的成員「size」3228之值時，即滿足到達mdia原子912之末端的條件。

11. 操作3980，若已到達mdia原子912之末端，則該程序完成，以操作3970繼續處理。否則，重複操作3910至操作3970。

程序「處理磁軌編輯原子」包括重複以下步驟，直至到達edts原子945的資料末端。自mp4檔案從edts原子945之起端，讀取的資料位元組數等於edts原子的大小時，即滿足到達edts原子945之末端的條件。期望在edts原子945中找到單一elst原子948。

1. 建立一新TrackEdit結構3236。

2. 將該新TrackEdit結構3236的位址，指派給目前trak結構3200中的成員TrackEdit 3208。

3. 將「atomSize」之值指派給新TrackEdit結構3236中的成員「size」3238。

4. 自mp4檔案中讀取一32位元整數，並將結果指派給TrackEdit結構3236中的成員「version & Flags」3240。

5. 自mp4檔案中讀取一32位元整數，並將該整數值指派給數量「atomSize」。然後自該mp4檔案中讀取第二32位元整數，並將該整數值指派給數量「atomId」。

6. 比較數量「atomId」之值與「elst」值。

7. 若數量「atomId」之值對應於「elst」，則執行程序「處理編輯列表原子」。否則，跨越mp4檔案中的atomSize-8位元組。即，自mp4檔案中讀取atomSize-8位元組，並忽略所得

值。

8. 檢查到達edts(磁軌編輯)原子945之末端的情形。自mp4檔案從edts原子945之起端，讀取的資料位元組數等於TrackEdit結構3236的成員「size」3238之值時，即滿足到達edts原子945之末端的條件。

9. 若已到達edts原子945之末端，則該程序完成。否則，重複步驟5至8。

程序「處理編輯列表原子」由以下步驟組成：

1. 建立一新EditList結構3244。

2. 將該新EditList結構3244的位址，指派給目前TrackEdit結構3236的成員「EditList」3242。

3. 將「atomSize」之值指派給新EditList結構3244中的成員「size」3246。

4. 自mp4檔案中讀取一32位元整數，並將結果指派給EditList結構3244中的成員「version & Flags」3248。

5. 自mp4檔案中讀取一32位元整數，並將該值指派給數量「numSegments」。

6. 將數量「numSegments」之值指派給EditList結構3244的成員「numSegments」3250。

7. 建立一numSegmentsEditSegment結構3256之陣列，並將該陣列之位址指派給EditList結構3244的成員「EditSegment array」。

8. 將零值指派給數量「segment」。

9. 重複以下步驟(10至13)numSegments次(直至數量

「segment」之值等於數量「numSegments」之值)：

10. 自mp4檔案中讀取一32位元整數，並將結果指派給EditSegment結構3252陣列中項目「segment」的成員「startTime」3258。

11. 自mp4檔案中讀取一32位元整數，並將結果指派給EditSegment結構3252陣列中項目「segment」的成員「duration」3260。

12. 自mp4檔案中讀取一32位元整數，並將結果指派給EditSegment結構3252陣列中項目「segment」的成員「relativeRate」3262。

13. 將數量「segment」之值遞增1。

程序「處理磁軌參考原子」由以下步驟組成：期望在tref原子940中找到單一mpod原子942。

1. 建立一新TrackReference結構3264。

2. 將該新TrackReference結構3264的位址，指派給目前trak結構3200的成員「TrackReference」。

3. 將該atomSize之值指派給該新TrackReference結構3264的成員「size」3266。

4. 重複以下步驟，直至到達tref原子940的資料末端。自mp4檔案從tref原子940之起端，讀取的資料位元組數等於tref原子940的大小時，即滿足到達tref原子940之末端的條件。

5. 自mp4檔案中讀取一32位元整數，並將該整數值指派給數量「atomSize」。然後自該mp4檔案中讀取第二32位元

整數，並將該整數值指派給數量「atomId」。

6. 比較數量「atomId」之值與「mpod」值。

7. 若數量「atomId」之值對應於「mpod」，則執行程序「處理mpod原子」。否則，跨越mp4檔案中的atomSize-8位元組。即，自mp4檔案中讀取atomSize-8位元組，並忽略所得值。

8. 檢查到達tref(磁軌參考)原子940之末端的情形。自mp4檔案從tref原子940之起端，讀取的資料位元組數等於TrackReference結構3264的成員「size」3266之值時，即滿足到達tref原子940之末端的條件。

9. 若已到達tref原子940之末端，則該程序完成。否則，重複步驟5至8。

程序「處理mpod原子」由以下步驟組成：

1. 建立一新Mpod資料結構3270。

2. 將該新Mpod資料結構3270的位址，指派給目前TrackReference資料結構3264的成員「Mpod」3268。

3. 將該atomSize之值指派給該新mpod資料結構的成員「size」3272。

4. 自Mpod成員「size」之值中減去8，結果除以四，並將所得結果指派給數量「numEntries」。

5. 將「numEntries」之值指派給新Mpod資料結構3270的成員「numEntries」3274。

6. 建立具有「numEntries」項目之一整數陣列，並將該陣列之位址指派給新Mpod資料結構3270的成員「trackID

數值指派給數量「atomSize」。接著自mp4檔案讀取第二32位元整數並將該整數值指派給數量「atomId」。

3. 操作4020，比較數量「atomId」值與「dinf」值。

4. 操作4025，若數量「atomId」值對應「dinf」則執行程序「處理媒體資料資訊(dinf)原子」。然後以操作4080繼續。

5. 操作4030，比較數量「atomId」值與「stbl」值。

6. 操作4035，若數量「atomId」值對應「stbl」則執行程序「處理樣本表原子」。然後以操作4080繼續。

7. 操作4040，比較數量「atomId」值與「nmhd」值。

8. 操作4045，若數量「atomId」值對應「nmhd」則執行程序「處理動畫專家群第四階媒體標頭原子」。然後以操作4080繼續。

9. 操作4050，比較數量「atomId」值與「smhd」值。

10. 操作4055，若數量「atomId」值對應「smhd」則執行程序「處理聲音媒體標頭原子」。然後以操作4080繼續。

11. 操作4060，比較數量「atomId」值與「vmhd」值。

12. 操作4065，若數量「atomId」值對應「vmhd」則執行程序「處理視頻媒體標頭原子」。然後以操作4080繼續。

13. 否則操作4070，跨越mp4檔案內atomSize-8位元組。即自mp4檔案讀取atomSize-8位元組，並忽略所得值。

14. 操作4080，檢查到達minf原子920末端之情形。當自mp4檔案從minf原子920之起端讀取的資料位元組數等於MediaInfo資料結構3333內成員「size」3336之值時，即滿足

到達minf原子920末端的條件。

15. 操作4090，若已經到達minf原子920末端則完成該程序。否則重複操作4010至操作4080。

程序「處理媒體資料資訊(dinf)原子」由以下步驟組成：

1. 建立新MediaDataInfo結構3350。

2. 將該新MediaDataInfo結構3350之位址指派給目前MediaInfo結構3333內成員「MediaDataInfo」3338。

3. 將數量「atomSize」值指派給MediaDataInfo結構3350內成員「size」3353。

4. 自mp4檔案讀取一32位元整數並將結果指派給MediaDataInfo結構3350內成員「version & Flags」3356。

5. 重複以下步驟直至到達dinf原子924之資料的末端。當自mp4檔案從dinf原子924之起端讀取的資料位元組數等於MediaDataInfo結構3350內成員「size」之值時即滿足到達dinf原子924末端的條件。

6. 自mp4檔案讀取一32位元整數並將該整數值指派給數量「atomSize」。

7. 自mp4檔案讀取一32位元整數並將該整數值指派給數量「atomId」。

8. 若數量「atomId」值對應「dref」則執行程序「處理資料參考表原子」。否則跨越mp4檔案內atomSize-8位元組。即自mp4檔案讀取atomSize-8位元組並忽略所得值。

9. 檢查到達dinf原子924末端之情形。若已經到達dinf原子924之末端，該程序結束。否則重複步驟2至9。

程序「處理聲音媒體標頭原子」由以下步驟組成：

1. 建立新SoundMediaHeader資料結構。SoundMediaHeader資料結構包含成員「size」、「version & Flags」及「balance」。

2. 將該新SoundMediaHeader資料結構之位址指派給目前MediaInfo資料結構3333內成員MediaInfoHeader 3343。

3. 將數量「atomSize」值指派給該新SoundMediaHeader資料結構之成員「size」。

4. 自mp4檔案讀取一32位元整數並將結果指派給新SoundMediaHeader結構內成員「version & Flags」。

5. 自mp4檔案讀取一32位元整數並將結果指派給目前SoundMediaHeader結構內成員「balance」。

6. 跨越聲音媒體標頭(smhd)原子936內包含的任何剩餘資料(若atomSize大於16)。

程序「處理視頻媒體標頭原子」由以下步驟組成：

1. 建立新VideoMediaHeader資料結構。VideoMediaHeader資料結構包含成員「size」、「version & Flags」、「mode」、「opRed」「opGreen」及「opBlue」。

2. 將該新VideoMediaHeader資料結構之位址指派給目前MediaInfo資料結構3333內成員MediaInfoHeader 3343。

3. 將atomSize值指派給該新VideoMediaHeader資料結構之成員「size」。

4. 自mp4檔案讀取一32位元整數並將結果指派給新VideoMediaHeader結構內成員「version & Flags」。

程序「處理大塊偏移表(stco)原子」。然後以操作4190繼續。

7. 操作4140，比較數量「atomId」值與「stts」值。

8. 操作4145，若數量「atomId」值對應「stts」則執行程序「處理time to sample table (stts)原子」。然後以操作4190繼續。

9. 操作4150，比較數量「atomId」值與「stsz」值。

10. 操作4155，若數量「atomId」值對應「stsz」則執行程序「處理樣本大小表(stsz)原子」。然後以操作4190繼續。

11. 操作4160，比較數量「atomId」值與「stss」值。

12. 操作4165，若數量「atomId」值對應「stss」則執行程序「處理同步樣本表(stss)原子」。然後以操作4190繼續。

13. 操作4170，比較數量「atomId」值與「stsd」值。

14. 操作4175，若數量「atomId」值對應「stsd」則執行程序「處理sample description table (stsd)原子」。然後以操作4190繼續。

15. 否則操作4180，跨越mp4檔案內atomSize-8位元組。即自mp4檔案讀取atomSize-8位元組並忽略所得值。

16. 操作4190，檢查到達stbl原子950末端之情形。當自mp4檔案從stbl原子950之起端讀取的資料位元組數等於SampleTables資料結構3400內成員「size」3403之值時即滿足到達stbl原子950末端的條件。

17. 操作4195，若已經到達stbl原子950之末端，該程序結束。否則重複操作4110至操作4190。

程序「處理sample to chunk table (stsc)原子」由以下步驟

2. 將該新樣本表資料結構3420之位址指派給目前SampleTables資料結構3400內成員TimeToSampleTable 3410。

3. 將數量「atomSize」值指派給新樣本表資料結構3420之成員「size」3423。

4. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「version & Flags」3426。

5. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「numEntries」3428。

6. 建立具有numEntries項目之位址陣列，並將該陣列位址指派給新樣本表資料結構3420之成員「table array」3430。

7. 將零值指派給數量「entry」。

8. 重複以下步驟numEntries次(直至數量「entry」值等於成員「numEntries」3428值)。

9. 建立新TimeToSample資料結構3470。

10. 將該新TimeToSample資料結構3470之位址指派給樣本表資料結構3420之成員「table array」3430內的項目「entry」。

11. 建立新TimeToSample資料結構3470。

12. 將該新TimeToSample資料結構3470之位址指派給樣本表資料結構3420之陣列「table array」3430內的項目「entry」。

13. 自mp4檔案讀取一32位元整數並將結果指派給新TimeToSample資料結構3470內成員「count」3473。

14. 自mp4檔案讀取一32位元整數並將結果指派給新TimeToSample資料結構3470內成員「duration」3476。

15. 將數量「entry」值遞增1。

程序「處理樣本大小表(stsz)原子」由以下步驟組成：

1. 建立新SampleSizeTable資料結構3440。

2. 將該新SampleSizeTable結構3440之位址指派給目前SampleTables資料結構3400內成員SampleSizeTable 3413。

3. 將數量「atomSize」值指派給新SampleSizeTable資料結構3440之成員「size」3443。

4. 自mp4檔案讀取一32位元整數並將結果指派給新SampleSizeTable資料結構3440內成員「version & Flags」3446。

5. 自mp4檔案讀取一32位元整數並將結果指派給新SampleSizeTable資料結構3440內成員「sampleSize」3450。若新SampleSizeTable資料結構3440內成員樣本大小3450之值為零，則執行以下步驟，否則將零值指派給新SampleSizeTable資料結構3440內成員「numEntries」3453及「table array」3456。

6. 自mp4檔案讀取一32位元整數並將結果指派給新SampleSizeTable資料結構3440內成員「numEntries」3453。

7. 建立具有numEntries項目之整數陣列，並將該陣列位址指派給新SampleSizeTable資料結構3440之成員「table array」3456。

8. 將零值指派給數量「entry」。

9. 重複以下步驟numEntries次(直至數量「entry」值等於成員「numEntries」3453值)。

10. 自mp4檔案讀取一32位元整數並將結果指派給SampleSizeTable資料結構3440內成員「table array」3456之項目「entry」。

11. 將數量「entry」值遞增1。

程序「處理同步樣本表(stss)原子」由以下步驟組成：

1. 建立新樣本表資料結構3420。

2. 將該新樣本表資料結構3420之位址指派給目前SampleTables資料結構3400內成員SyncSampleTable 3416。

3. 將數量「atomSize」值指派給新樣本表資料結構3420之成員「size」3423。

4. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「version & Flags」3426。

5. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「numEntries」3428。

6. 建立具有numEntries項目之整數陣列，並將該陣列位址指派給新樣本表資料結構3420之成員「table array」3430。

7. 將零值指派給數量「entry」。

8. 重複以下步驟numEntries次(直至數量「entry」值等於成員「numEntries」3428值)。

9. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內陣列「table array」3430的項目「entry」。

10. 將數量「entry」值遞增1。

程序「處理樣本描述表(stsd)原子」由以下步驟組成：

1. 建立新樣本表資料結構3420。

2. 將該新樣本表資料結構3420之位址指派給目前SampleTables資料結構3400內成員SampleDescriptionTable 3418。

3. 將數量「atomSize」值指派給新樣本表資料結構3420之成員「size」3423。

4. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「version & Flags」3426。

5. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「numEntries」3428。

6. 建立具有numEntries項目之位址陣列，並將該陣列位址指派給新樣本表資料結構3420之成員「table array」3430。

7. 將零值指派給數量「entry」。

8. 重複以下步驟numEntries次(直至數量「entry」值等於成員「numEntries」3428值)。

9. 建立新SampleDescription資料結構3480。

10. 將該新SampleDescription資料結構3480之位址指派給新樣本表資料結構3420之成員「table array」3430內的項目「entry」。

11. 自mp4檔案讀取一32位元整數並將結果指派給SampleDescription資料結構3480內成員「size」3483。

12. 自mp4檔案讀取一32位元整數並將結果指派給SampleDescription資料結構3480內成員「dataFormat」3486。

料3168以mp4bifs文件2260、2800的形式代表。與物件描述符資料流(若有)相關之媒體資料3168在mp4file文件2300內以xml形式2500代表。

如圖42所示，該步驟按以下程序完成：

1. 在操作4200，將零值指派給數量「iMdat」、「numVisual」及「numAudio」。

2. 操作4210比較數量「iMdat」值與QtInfo結構3100之成員「numMdatAtoms」3110值。

3. 在操作4220，若數量「iMdat」值等於成員「numMdatAtoms」之值，該步驟結束。否則繼續以下程序。

4. 操作4230將QtInfo結構3100之成員「mdat陣列」3113內的項目「iMdat」值指派給代表mdat資料結構3160位址之數量「mdatAtom」。

5. 操作4240將mdat資料結構「mdatAtom」3160之成員「start」3166值指派給數量「mdatStart」。

6. 操作4250將mdatAtom成員「size」3163值及數量「mdatStart」值之和指派給數量「nextMdat」。

7. 操作4260建立新「mdat」元件2310並將其附加至mp4file文件2300之頂階層元件。其將數量「iMdat」值指派給新mdat元件之屬性「mdatID」，將數量「mdatSize」值指派給新mdat元件之屬性「size」，以及將mdat資料結構「mdatAtom」之成員「MediaData」值指派給數量「mediaData」。

8. 操作4270執行程序「處理媒體資料」。

9. 操作4280將數量「iMdat」值遞增一，接著重複操作4210。

程序「處理媒體資料」

如圖43所示，程序「處理媒體資料」4270由以下步驟組成：

1. 操作4300，將零值指派給數量「iTrack」。

2. 操作4310，比較數量「iTrack」值與QtInfo結構3100之成員「numTracks」3118值。

3. 操作4315，若數量「iTrack」值等於QtInfo結構3100之成員「numTracks」3118值則該程序結束。

4. 操作4320，將QtInfo結構3100之成員「trak陣列」3120內項目「iTrack」值指派給trak資料結構3200「trakAtom」之位址。將資料結構「trakAtom」3200內成員TrackMedia 3206內成員MediaInfo 3234內，成員SampleTables 3340內成員ChunkOffsetTable 3408內的成員「table array」3430所指定之整數陣列內第一項目(entry0)值指派給數量「chunkOffset」。

5. 操作4330，若數量「chunkOffset」值大於零、大於或等於數量「mdatStart」值以及小於數量「nextMdat」值則執行以下步驟。否則，將數量「iTrack」值遞增一(操作4390)。

6. 操作4340，執行程序「找出資料流類型」。

7. 操作4350，比較數量「streamType」值與「1」值。

8. 操作4355，若數量「streamType」值係「1」則執行程序「OdsmDataToXml」。

9. 操作4360，比較數量「streamType」值與「3」值。

10. 操作4365，若數量「streamType」值係「3」則執行程序「SdsmDataToXml」。

11. 操作4370，比較數量「streamType」值與「4」值。

12. 操作4360，若數量「streamType」值係「4」則執行程序「VisualDataToXml」然後將「numVisual」值遞增一。

13. 操作4380，比較數量「streamType」值與「5」值。

14. 操作4385，若數量「streamType」值係「5」則執行程序「AudioDataToXml」然後將「numAudio」值遞增一。

15. 操作4390，將數量「iTrack」值遞增一。

程序「找出資料流類型」

如圖44所示，程序「找出資料流類型」由以下步驟組成：

1. 操作4400，將資料結構「trakAtom」內成員TrackMedia 3206內，成員MediaInfo 3234內成員SampleTables 3340內的成員SampleDescrTable 3418值指派給代表樣本表結構3420位址之數量「stsdAtom」。

2. 操作4410，將由數量「stsdAtom」值指定之樣本表結構3420的成員「table array」3430內，第一項目(entry0)所指定SampleDesctiption資料結構3480內成員「esdData」3496值指派給代表位元組陣列之數量「esdData」。

3. 操作4420，將由數量「stsdAtom」值指定之樣本表資料結構3420的成員「table array」內，第一項目(entry 0)所指定SampleDescription資料結構3480內成員「esdSize」值指派給數量「esdSize」。

ChunkOffsetTable、SampleToChunkTable等等係trak結構3200 trakAtom 4320之成員「TrackMedia」3206代表的TrackMedia資料結構3226之成員「MediaInfo」3234所代表的MediaInfo資料結構3333之成員「Sample Tables」3340代表的SampleTables結構3400之成員。

4. 操作4715將零值指派給數量「sttsIndex」。包含於樣本表結構3420TimeToSampleTable 3410內之表陣列3430內第一項目值，指派給代表TimeToSample結構3470位址的數量sttsEntry。TimeToSample結構sttsEntry之成員「count」值指派給數量「count」。TimeToSample結構sttsEntry之成員「duration」值指派給數量「duration」。

5. 操作4720將零值指派給數量「stscIndex」。包含於樣本表結構3420 SampleToChunkTable 3406內之表陣列3430內第一項目值指派給代表SampleToChunkTable結構3460位址的數量stscEntry。SampleToChunk結構stscEntry之成員「firstChunk」值減一指派給整數數量「firstChunk」。

6. 操作4725將零值指派給整數數量「time」、「sample」及「chunk」。

7. 操作4730比較數量「chunk」值與數量「numChunks」值。

8. 在操作4735，若數量「chunk」值等於數量「numChunks」值，程序「OdsmDataToXml」結束。

9. 在操作4740，若數量「chunk」值不等於數量「numChunks」值，值「8」加入mdat結構3160mdatAtom 4230

之成員「start」值，結果從包含於樣本表結構3420 ChunkOffsetTable 3408內之表陣列3430內的項目「chunk」值減去，其結果指派給整數數量「offset」。

10. 操作4745比較數量「chunk」值與數量「firstChunk」值。

11. 在操作4750，若數量「chunk」值等於數量「firstChunk」值，SampleToChunk結構3460 stscEntry內成員「numSamples」3466值指派給數量「numSamples」。數量「stscIndex」值遞增1。若數量「stscIndex」值小於數量「numStscEntries」值，包含於樣本表結構3420 SampleToChunkTable 3406內之表陣列3430內的項目「stscIndex」值指派給SampleToChunk結構3460 stscEntry，並且SampleToChunk結構stscEntry之成員「firstChunk」值減一指派給整數數量「firstChunk」。否則數量「numChunks」值指派給數量「firstChunk」。

12. 操作4755藉由加總包含於SampleSizeTable結構3440 SampleSizeTable 3413內之整數陣列3456內的項目「樣本」至「樣本+numSamples-1」決定數量「chunkSize」值。

13. 操作4760建立新「odsmChunk」元件並附加至建立於操作4705之新「odsm」元件。數量「offset」值指派給新「odsmChunk」元件之屬性「offset」。數量「chunkSize」值指派給新「odsmChunk」元件之屬性「size」。

14. 操作4770執行程序「解碼odsm大塊」，如圖48所示。

15. 操作4780將數量「chunk」值遞增1，接著重複操作4730。

程序「解碼odsm大塊」

如圖48所示，程序「解碼odsm大塊」由以下步驟組成：

1. 比較數量「chunkSize」值與零之操作4800。

2. 若數量「chunkSize」值不大於零，程序「解碼odsm大塊」結束以及程序「OdsmDataToXml」在操作4780恢復的操作4805。

3. 若數量「chunkSize」值大於零，SampleSizeTable結構3440 SampleSizeTable 3413內成員表3456內項目「sample」值指派給整數數量「sampleSize」的操作4810。

4. 操作4820，其中新odsmSample元件建立並附加至以上4760建立之odsmChunk元件。數量「offset」值指派給新odsmSample元件之屬性「offset」。數量「sampleSize」值指派給新odsmSample元件之屬性「size」。數量「time」值指派給新odsmSample元件之屬性「time」。

5. 操作4825，其中數量「time」值遞增數量「duration」值。數量「count」值遞減值「1」。

6. 操作4830，其中數量「count」值與零比較。

7. 若數量「count」值不大於零，數量「sttsIndex」值遞增值「1」之操作4835。包含於樣本表結構3420 TimeToSampleTable 3410內之表陣列3430內項目「sttsIndex」值指派給TimeToSample結構3470 sttsEntry。TimeToSample結構sttsEntry之成員「count」值指派給數量「count」。TimeToSamplesttsEntry之成員「duration」值指派給數量「duration」。

8. 操作4840，其中數量「chunkSize」值遞減數量「sampleSize」值。

9. 操作4845，其中數量「sampleSize」值與零比較。

10. 若數量「sampleSize」值不大於零，數量「sample」值遞增值「1」並且以操作4800繼續處理之操作4850。

11. 若數量「sampleSize」值大於零，將位元組「OdsmData」陣列內項目「offset」值指派給數量「tag」的操作4860。數量「offset」值遞增值「1」並且數量「sampleSize」值遞減值「1」。

12. 操作4865，其中執行程序「得到標籤大小」。此與上述程序「得到標籤大小」等效，除了

(a)位元組OdsmData陣列替代陣列「esdData」使用，

(b)數量「offset」替代數量「esdPos」使用

(c)每次數量「offset」值(esdPos)遞增一(操作4660、4690)則數量「sampleSize」值遞減值「1」。

13. 操作4870，其中數量「tag」值與動畫專家群第四階系統規格內定義之數量「ObjectDescrUpdateTag」值比較。

14. 若數量「tag」值等於數量「ObjectDescrUpdateTag」值，則執行程序「decodeOdUpdate」之操作4875。

15. 否則數量「tag」值與動畫專家群第四階系統規格內定義之數量「ObjectDescrRemoveTag」值比較的操作4880。

16. 若數量「tag」值等於數量「ObjectDescrRemoveTag」值，則執行程序「decodeOdRemove」之操作4885。

17. 否則數量「offset」值遞增數量「tagSize」值的操作

大小」等效，除了(a)位元組OdsmData陣列替代「esdData」陣列使用，以及(b)數量「offset」替代數量「esdPos」使用。將數量「tagSize」值指派給數量「odSize」。

7. 操作4950將數量「residualSize」值遞減數量「odSize」值。

8. 操作4960比較數量「odTag」值與動畫專家群第四階系統規格內定義之「MP4_OD_Tag」值。

9. 在操作4965，若數量「odTag」值不等於「MP4_OD_Tag」，將數量「offset」值遞增數量「tagSize」值，然後重複操作4920。

10. 在操作4970，若數量「odTag」值等於「MP4_OD_Tag」2108值，建立新「ObjectDescriptor」元件2550並將其附加至目前ObjectDescriptorUpdate元件2540。

11. 操作4980決定數量「ObjectDescriptorID」2124值並將結果指派給新ObjectDescriptor元件2550之屬性「ODID」。

數量「ObjectDescriptorID」值可藉由以下操作決定：

(a)將位元組「OdsmData」陣列內項目偏移值指派給整數數量「temp1」，

(b)將數量「offset」值遞增一併將數量「odSize」值遞減一，

(c)將位元組「OdsmData」陣列內項目偏移值指派給整數數量「temp2」，

(d)將數量「offset」值遞增一併將數量「odSize」值遞減一，

(e)將數量「temp1」值乘以4(向左移動2位元)，

(f)將數量「temp2」值除以64(向右移動6位元)，

(g)將「temp1」及「temp2」值之和指派給整數數量「ObjectDescriptorID」。

12. 操作4990執行下述程序「decodeEsIdRef」。

程序「解碼EsIdRef」

如圖50所示，程序「decodeEsIdRef」由以下步驟組成：

1. 操作5000比較數量「odSize」值與「1」值。

2. 在操作5010，若數量「odSize」值小於「1」，該程序結束。

3. 操作5020將位元組「OdsmData」陣列內項目「offset」值指派給數量「esdTag」，將數量「offset」值遞增「1」並將數量「odSize」值遞減「1」。

4. 操作5030使用程序「得到標籤大小」獲得數量「tagSize」值(numBytes 2180)。此與上述程序「得到標籤大小」等效，除了(a)位元組OdsmData陣列替代「esdData」陣列使用，以及(b)數量「offset」替代數量「esdPos」使用。將數量「tagSize」值指派給數量「esdSize」。

5. 操作5040將數量「odSize」值遞減數量「esdSize」值。

6. 操作5050比較數量「esdTag」值與動畫專家群第四階系統規格內定義之「ES_ID_RefTag」值。

7. 在操作5060，若數量「esdTag」值不等於「ES_ID_RefTag」值，將數量「offset」值遞增數量「esdSize」值，然後重複操作5000。

8. 在操作5070，若數量「esdTag」值等於「ES_ID_ RefTag」2170值，建立新「EsIdRef」元件2560並將其附加至目前ObjectDescriptor元件2550。

9. 操作5080決定數量「ES_ID」2190值並將結果指派給新EsIdRef元件2560之屬性「EsId」。

數量「ES_ID」值可藉由以下操作決定：

(a)將位元組「OdsmData」陣列內項目偏移值指派給整數數量「temp1」，

(b)將數量「offset」值遞增一，

(c)將位元組「OdsmData」陣列內項目偏移值指派給整數數量「temp2」，

(d)將數量「offset」值遞增一，

(e)將數量「temp1」值乘以256(向左移動8位元)，

(f)將「temp1」及「temp2」值之和指派給整數數量「ES_ID」。

程序「解碼OdRemove」

若用於odsm命令1960之命令標籤4860值等於「ObjectDescrRemoveTag」4880、2050值，則執行4885以下步驟：

1. 建立新ObjectDescrRemove元件2570並將其附加至目前odsmSample元件2520。

2. 使用程序「得到標籤大小」獲得數量「tagSize」值(numBytes 2060)。該與上述程序「得到標籤大小」等效，除了(a)位元組OdsmData陣列替代「esdData」陣列使用，以及(b)數量「offset」替代數量「esdPos」使用。

3. 將數量「tagSize」值乘以8，將結果除以10，其結果指派給數量「numOdIds」。

4. 建立新空串數量「OdIdList」。

5. 將位元組「OdsmData」陣列視為以項目「offset」開始之位元序列。從該位元序列擷取numOdId連續10位元整數值(objectDescriptorId 2070)。

6. 為每個該等objectDescriptorId值2070建立字串代表。將每個該等字串值連接至字串數量「OdIdList」，使用單一空白空間字元分離連續值。

7. 將字串數量「OdIdList」之所得值指派給新ObjectDescrRemove元件2570之屬性「ODID」。

程序「VisualDataToXml」

程序「VisualDataToXml」4375建立包括包含於mdat結構「mdatAtom」4230內的視覺媒體資料流的大塊之xml代表。包含於此等大塊內的媒體資料亦複製到外部媒體資料檔案。如圖51所示，該程序由以下操作組成：

1. 在操作5100，將mdat結構「mdatAtom」4230之成員「MediaData」3168值指派給代表位元組陣列位址的數量「MediaData」。mdat結構「mdatAtom」4230之成員「size」3163值指派給整數數量「MediaSize」。

2. 在操作5105，新「mediaFile」元件2480建立並附加至目前mdat元件2400、4260。每個該類「mediaFile」元件2480具有兩個屬性「trackID」及「name」。屬性「name」值指定將包含媒體資料大塊(其包括該資料流)的媒體資料檔案

2230之名稱。

包含於trak結構「trakAtom」4320內之TrackHeader結構3212、3204的成員trackID 3224值指派給新「mediaFile」元件2480之屬性「trackID」值。

「name」屬性值藉由連接三個字元串、檔案名稱根、序列號及檔案類型後置決定。檔案名稱根可藉由輸入mp4檔案名稱副本決定，除最後出現之「.」(點或句點)字元及全部隨後字元外。序列號可自視覺目標序列號(「numVisual」)值字串代表形成。

檔案類型後置可取決於包含於媒體資料流內的數量objectType及/或檔案格式指示值。例如，若數量objectType值係「32」，檔案類型後置可指定為「.m4v」(動畫專家群第四階視頻)。

3. 在操作5110，新媒體資料檔案為寫入打開。該新媒體資料檔案名稱藉由建立於操作5105內之mediaFile元件2480的「name」屬性值決定。

在「動畫專家群第四階視頻」(objectType 1044為「32」)情形中，初始資料區塊必須置於新媒體資料檔案之開始。該資料區塊包含於「解碼器特定資訊」區塊1072、1076內，其包含於用於具有trackID值3224(具有數量「iTrack」指定之值)的trak結構3200之SampleDescription資料結構3480內成員「esdData」3496指定的位元組陣列內。SampleDescription資料結構3480藉由用於該trak結構之SampleDescription資料結構3400內SampleDescriptionTable資料結

構3418之成員表3430內第一項目(項目0)決定「解碼器特定資訊」區塊1072、1076包含於「解碼器組態描述符」資料區塊1024、1032內，其包含於該esdData資料區塊3496內。

解碼器組態描述符資料1032可使用上述程序「找出解碼器組態描述符」定位。解碼器特定資訊資料區塊1072、1076位於緊跟decoderConfigDescriptorTag 1036、解碼器組態描述符大小值1040及解碼器組態描述符資料1044至1068之14位元組後的解碼器組態描述符1032內。解碼器特定資訊資料區塊1076由一位元組decoderSpecificInfoTag 1080(值「5」)、解碼器特定資訊資料區塊大小值(numBytes 1084)及解碼器特定資訊資料之numBytes位元組組成。該numBytes 1084位元組序列複製到輸出動畫專家群第四階視頻檔案2230。

4. 在操作5115，樣本表結構3420 ChunkOffsetTable 3408之成員numEntries 3428值指派給數量「numChunks」。樣本表結構3420 SampleToChunkTable 3406之成員numEntries 3428值指派給數量「numStscEntries」。樣本表結構ChunkOffsetTable、SampleToChunkTable等等係trak結構3200 trakAtom 4230之成員「TrackMedia」3206代表的TrackMedia資料結構3226之成員MediaInfo 3234代表的MediaInfo資料結構3333之成員「SampleTables」3340代表的SampleTables結構3400之成員。

5. 在操作5120，TimeToSample結構sttsEntry之成員「count」值指派給數量「count」。

6. 在操作5125，零值指派給數量「stscIndex」。包含於樣本表結構3420 SampleToChunkTable 3406內之表陣列3430內第一項目值指派給代表SampleToChunk結構3460位址的數量 stscEntry。SampleToChunk結構 stscEntry之成員「firstChunk」值減一指派給整數數量「firstChunk」。

7. 在操作5130，零值指派給整數數量「chunk」。

8. 在操作5140，比較數量「chunk」值與數量「numChunks」值。

9. 在操作5145，若數量「chunk」值等於數量「numChunks」值，程序「VisualDataToXml」結束。

10. 在操作5150，若數量「chunk」值不等於數量「numChunks」值，值「8」加入mdat結構3160 mdatAtom 4230之成員「start」值，結果從包含於樣本表結構3420 ChunkOffsetTable 3408內之表陣列3430內的項目「chunk」值減去，其結果指派給整數數量「offset」。

11. 在操作5160，比較數量「chunk」值與數量「firstChunk」值。

12. 在操作5165，若數量「chunk」值等於數量「firstChunk」值，SampleToChunk結構3460 stscEntry內成員「numSamples」3466值指派給數量「numSamples」。數量「stscIndex」值遞增1。若數量「stscIndex」值小於數量「numStscEntries」值，包含於樣本表結構3420 SampleToChunkTable 3406內之表陣列3430內的項目「stscIndex」值指派給SampleToChunk結構3460 stscEntry，並且SampleToChunk結構stscEntry之成員

「firstChunk」值減一指派給整數數量「firstChunk」。否則數量「numChunks」值指派給數量「firstChunk」。

13. 在操作5170，數量「chunkSize」值藉由加總包含於SampleSizeTable結構3440 SampleSizeTable 3413內之整數陣列3456內的項目「sample」至「樣本+numSamples-1」決定。

14. 在操作5180，新「chunk」元件建立並附加至操作5105內建立的新「mediaFile」元件。數量「offset」值指派給新「chunk」元件之屬性「offset」。數量「chunkSize」值指派給新「chunk」元件之屬性「size」。

15. 在操作5185，自位元組陣列MediaData之起端在偏移位元組開始的媒體資料之chunkSize位元組複製到操作5110打開的輸出媒體資料檔案。

16. 在操作5190，數量「chunk」值遞增1，接著重複操作5140。

程序「AudioDataToXml」

程序「AudioDataToXml」4385建立包括包含於mdat結構「mdatAtom」4230內的聲頻媒體資料流的大塊之xml代表。包含於此等大塊內的媒體資料亦複製到外部媒體資料檔案。如圖51所示，該程序與上述程序「VisualDataToXml」等效，除以下差異外：

用於形成「name」屬性值之序列號可自聲頻目標序列號「numAudio」之字串代表形成。

檔案類型後置可取決於包含於媒體資料流內的數量objectType及/或檔案格式指示值。例如，若數量objectType

值係「107」，檔案類型後置可指定為「.mp3」(動畫專家群第一階3層聲頻)。

在操作5145，若數量「chunk」值等於數量「numChunks」值，程序「AudioDataToXml」結束。

某些類型聲頻資料，例如動畫專家群第二階AAC(先進聲頻編碼)可需要超出本文說明的額外處理。執行該類處理之裝置可由熟習技術人士根據對應媒體特定檔案結構規格設計。本文未涵蓋用以處理每種類型聲頻資料的特定裝置。

程序「SdsmDataToXml」

程序「SdsmDataToXml」4365建立包含於目前mdat結構「mdatAtom」4230內的sdsm(BIFS)資料流(場景描述資料流)之xml代表。該程序建立之sdsm的xml代表由兩部分組成：

(a)sdsm元件2440，其包含包括sdsm之媒體資料大塊2450的概要。該sdsm元件2440包含於目前mdat元件4260內，其形成mp4file文件2300之部分。

(b)sdsm媒體資料樣本1100、1110之xml代表建立於mp4bifs文件2260、2800內。

如圖52所示，該程序由以下操作組成：

1. 在操作5200，將mdat結構「mdatAtom」4230之成員「MediaData」3168值指派給代表位元組陣列位址的數量「SdsmData」。mdat結構「mdatAtom」4230之成員「size」3163值指派給整數數量「SdsmSize」。

2. 在操作5205，新「sdsm」元件2410、2440建立並附加至目前mdat元件2400、4260。每個該類「sdsm」元件2440

具有兩個屬性「trackID」及「xmlFile」。屬性「xmlFile」指定可用以代表結果mp4bifs文件2260、2800的選擇性xml輸出檔案之名稱。該選擇性xml輸出檔案可用於診斷目的，但本發明之操作(用於根據mp4二進制檔案建立XMT-A檔案)不需要。

包含於trak結構「trakAtom」4320內之TrackHeader結構3212、3204的成員trackID 3224值指派給新「sdsm」元件2440之屬性「trackID」值。

「xmlFile」屬性值藉由連接兩個字元串、檔案名稱根及檔案類型後置決定。檔案名稱根可藉由輸入mp4檔案名稱副本決定，除最後出現之「.」(點或句點)字元及全部隨後字元外。檔案類型後置係「Bifs.xml」。

3. 在操作5210，新「bifsConfig」元件2810建立並附加至mp4bifs文件2260、2800之頂部元件。下述程序「設定bifsConfig屬性」接著用於設定「bifsConfig」元件2810之屬性。

4. 在操作5215，樣本表結構3420 ChunkOffsetTable 3408之成員numEntries 3428值指派給數量「numChunks」。樣本表結構3420 SampleToChunkTable 3406之成員numEntries 3428值指派給數量「numStscEntries」。樣本表結構ChunkOffsetTable、SampleToChunkTable等等係trak結構3200 trakAtom4320之成員「TrackMedia」3206代表的TrackMedia資料結構3226之成員MediaInfo 3234代表的MediaInfo資料結構3333之成員「SampleTables」3340代表的SampleTables

結構3400之成員。

5. 在操作5220，零值指派給數量「sttsIndex」。包含於樣本表結構3420 TimeToSampleTable 3410內之表陣列3430內第一項目值指派給代表TimeToSample結構3470位址的數量sttsEntry。TimeToSample結構sttsEntry之成員「count」值指派給數量「count」。TimeToSample結構sttsEntry之成員「duration」值指派給數量「duration」。

6. 在操作5225，零值指派給數量「stscIndex」。包含於樣本表結構3420 SampleToChunkTable 3406內之表陣列3430內第一項目值指派給代表SampleToChunk結構3460位址的數量stscEntry。SampleToChunk結構stscEntry之成員「firstChunk」值減一指派給整數數量「firstChunk」。

7. 在操作5230，零值指派給整數數量「chunk」。

8. 在操作5240，比較數量「chunk」值與數量「numChunks」值。

9. 在操作5245，若數量「chunk」值等於數量「numChunks」值，程序「SdsmDataToXml」結束。

10. 在操作5250，若數量「chunk」值不等於數量「numChunks」值，值「8」加入mdat結構3160 mdatAtom 4230之成員「start」值，結果從包含於樣本表結構3420 ChunkOffsetTable 3408內之表陣列3430內的項目「chunk」值減去，其結果指派給整數數量「offset」。

11. 在操作5260，比較數量「chunk」值與數量「firstChunk」值。

12. 在操作5265，若數量「chunk」值等於數量「firstChunk」值，SampleToChunk結構3460 stscEntry內成員「numSamples」3466值指派給數量「numSamples」。數量「stscIndex」值遞增1。若數量「stscIndex」值小於數量「numStscEntries」值，包含於樣本表結構3420 SampleToChunkTable 3406內之表陣列3430內的項目「stscIndex」值指派給SampleToChunk結構3460 stscEntry，並且SampleToChunk結構stscEntry之成員「firstChunk」值減一指派給整數數量「firstChunk」。否則數量「numChunks」值指派給數量「firstChunk」。

13. 在操作5270，數量「chunkSize」值藉由加總包含於SampleSizeTable結構3440 SampleSizeTable 3413內之整數陣列3456內的項目「sample」至「sample+numSamples-1」決定。

14. 在操作5275，具有numSamples項目之時間戳記的表根據包含於相關trak結構3200內的TimeToSampleTable資料結構3410、TrackEdit資料結構3208、3236及MediaHeader資料結構3230內表陣列成員3430內之項目內容建立。該表包含用於目前大塊內每個樣本的一個項目，該表內每個項目代表用於該資料流內全部前述樣本的持續時間值3476之和加上選擇性TrackEdit資料結構3208、3236內指定的任何初始延遲(若有)。熟習技術人士可建立用於自可用資訊產生該一表之裝置。

15. 在操作5280，新「chunk」元件建立並附加至步驟2內建立的新「sdsm」元件。數量「offset」值指派給新「chunk」

元件之屬性「offset」。數量「chunkSize」值指派給新「chunk」元件之屬性「size」。

16. 在操作5285，如圖53所示，執行程序「解碼sdsm大塊」。

17. 在操作5290，數量「chunk」值遞增1，接著重複操作5240。

程序設定bifsConfig屬性

mp4bifs bifsConfig元件2810具有七個屬性「nodeIdBits」、「routeIdBits」、「commandFrame」、「useBIFSv2Config」、「protoIdbits」、「use3DMeshCoding」及「usePredictive MFField」。此等屬性值藉由用於目前trak資料結構「trakAtom」4320之解碼器特定資訊資料1072的內容決定。該資料區塊包含於SampleTables資料結構3400(包含於目前trackAtom資料結構3400、4320內)之成員SampleDescription Table 3418的成員「table array」3430內項目0所指定之SampleDesctiption資料結構3480的esdData成員1000、3496內解碼器組態描述符1024、1032內。

若數量「objectType」4460值係「2」，則解碼器特定資訊資料代表「版本2」BIFS組態。該情形中，解碼器特定資訊區塊前三個位元組具有以下內容：

[uraaaaab bbbbpppp pcxxxxxx]

其中，

u=用於「use 3DMeshCoding」之1位元值，

r=用於「usePredictiveMFField」之1位元值，

aaaaa=用於「nodeIdBits」之5位元無符號整數值，

bbbbb=用於「routeIdBits」之5位元無符號整數值，

ppppp=用於「protoIdbits」之5位元無符號整數值，

c=用於「commandFrame」之1位元值。

aaaaa、bbbbb及ppppp指定的值指派給指示屬性。

「aaaaa」指定之無符號整數值指派給mp4bifs bifsConfig元件2810之「nodeIdBits」屬性。「bbbbb」指定之無符號整數值指派給mp4bifs bifsConfig元件2810之「routeIdBits」屬性。「ppppp」指定之無符號整數值指派給mp4bifs bifsConfig元件2810之「protoIdbits」屬性。

若「u」、「r」或「c」值為「0」，則值「假」指派給mp4bifs bifsConfig元件2810的對應屬性。若「u」、「r」或「c」值為「1」，則值「真」指派給mp4bifs bifsConfig元件2810的對應屬性。

若數量「objectType」4460值非「2」，則解碼器特定資訊資料代表「版本1」BIFS組態。該情形中，解碼器特定資訊區塊前兩個位元組具有以下內容：

[aaaaabbb bbcxxxxxl

其中，

aaaaa=用於「nodeIdBits」之5位元無符號整數值，

bbbbb=用於「routeIdBits」之5位元無符號整數值，

c=用於「commandFrame」之1位元值。

該情形中，「protoIdbits」、「use3DMeshCoding」及「usePredictiveMFField」值並不需要。

「aaaaa」指定之無符號整數值指派給mp4bifs bifsConfig元件2810之「nodeIdBits」屬性。「bbbbb」指定之無符號整數值指派給mp4bifs bifsConfig元件2810之「routeIdBits」屬性。

若「c」值為「0」，則值「假」指派給mp4bifs bifsConfig元件2810的「commandFrame」屬性。否則(若「c」值為「1」)，則值「真」指派給mp4bifs bifsConfig元件2810的「commandFrame」屬性。

程序「解碼sdsm大塊」

如圖53所示，程序「解碼sdsm大塊」由以下步驟組成：

1. 在操作5300，數量「offset」5250值指派給整數數量「sdsmPos」。數量「chunkSize」5270值加入數量「offset」值，結果指派給整數數量「sdsmEnd」。數量「sdsmPos」值指定位元組陣列「SdsmData」5200內位元組之位置。

相關整數數量「sdsmBit」指定由數量sdsmPos值指定之位元組內位元的位置。開始，零值指派給相關數量「sdsmBit」。隨後，對於從sdsm位元流擷取的每個位元，sdsmBit值遞增1。無論何時sdsmBit超過7，「sdsmBit」值遞減8且「sdsmPos」值遞增1。

2. 在操作5305，零值指派給整數數量「sample」且值「假」指派給布林數量「frameOpen」。

3. 在操作5310，比較數量「sdsmPos」值與數量「sdsmEnd」值。

4. 在操作5315，若數量「sdsmPos」值不小於數量

「sdsmEnd」值，該程序結束。

5. 在操作5320，若數量「sdsmPos」值小於數量「sdsmEnd」值，數量「frameOpen」值與值「假」比較。

6. 在操作5325，若數量「frameOpen」值等於值「假」，新「commandFrame」元件2820建立並附加至mp4bifs文件2800。時間戳記5275表內項目「sample」值指派給新commandFrame元件之「time」屬性。值「真」指派給布林數量「frameOpen」。

7. 在操作5330，始於由「sdsmPos」及「sdsmBit」值指定之點的sdsm位元流之下兩個位元指派給數量「cmdCode」。數量「sdsmBit」值遞增2，且數量「cmdCode」值與0比較。

8. 在操作5335，若數量「cmdCode」值等於「0」1206，執行程序「解碼插入命令」。

9. 操作5340，其中數量「cmdCode」值與「1」比較。

10. 在操作5345，若數量「cmdCode」值等於「1」1226，執行程序「解碼刪除命令」。

11. 操作5350，其中數量「cmdCode」值與「2」比較。

12. 在操作5355，若數量「cmdCode」值等於「2」1244，執行程序「解碼替換命令」。

13. 在操作5360，若「cmdCode」值非「0」、「1」或「2」，其必須係「3」，因為該值由資料兩個位元決定。因此，該步驟結果總是「是」。

14. 在操作5365，若數量「cmdCode」值等於「3」1280，執行程序「解碼ReplaceScene命令」。

程序「解碼ReplaceScene命令」包含建立新「ReplaceScene」bifsCommand元件2930並將其附加至目前「commandFrame」元件2830，操作5310。接著解譯緊跟場景替換cmdCode1280之BIFS場景資料結構1290、1600，根據需要導前sdsmBit及sdsmByte值。

「reserved」組件1610指定值藉由遞增sdsmBit值跨越。BIFS場景結構1600的USENAMES 1620及protoList 1630組件指定值決定並指派給新「ReplaceScene」元件之對應屬性。

程序「解碼SFNode」接著用以解譯BIFS場景結構1600之SFTopNode組件1640。新「ReplaceScene」bifsCommand元件2930用作結果TopNode BIFS節點元件2940之母元件。

一位元值「hasRoutes」1650指派給布林值「hasRoutes」。若「hasRoutes」值係「1」(「真」)，接著解譯以下Routes組件1660、1800、1830。新「Routes」元件2950建立並附加至新「ReplaceScene」元件2930。新「Routes」元件2950建立並附加至每個Route結構1810或1850(包含於Routes組件1660內)之新「Routes」元件2950。決定由每個Routes結構1860之routeID 1870、routeName 1875、outNodeID 1880、outFieldRef 1885、inNodeID 1890及inFieldRef 1895組件指定值並指派給每個新「Routes」元件2960的對應屬性。

15. 在操作5370，程序「解碼插入命令」、「解碼刪除命令」、「解碼替換命令」或「解碼ReplaceScene命令」完成後，sdsm位元流內下一位元指派給布林數量「continue」1130或1140。數量「continue」所得值係「1」(「真」)1130，該程

序繼續操作5310。

16. 若數量「continue」值係「0」(「假」)1140，sdsm位元流導前至下一位元組之起端的操作5375。即，若數量sdsmBit值係零則不作處理。否則，「sdsmBit」值設定為零且sdsmByte值遞增一。該步驟在sdsm命令框1110末端跨越「padding bits」1150。

17. 在操作5380，布林數量「frameOpen」值設定為「假」，數量「sample」值遞增一，且該程序繼續操作5310。

程序「解碼插入命令」

如圖54所示，程序「解碼插入命令」由以下步驟組成：

1. 在操作5400，sdsm位元流1210下兩個位元指派給數量「paramType」。

2. 操作5410，其中數量「paramType」值與「0」比較。

3. 在操作5420，若數量「paramType」值係「0」1308，執行程序「解碼InsertNode cmd」。

程序「解碼InsertNode cmd」包含建立新「InsertNode」bifsCommand元件2840並將其附加至目前母元件2830，操作5310。決定由nodeID 1312、insertionPosition 1316及Node Insertion命令1300之位置組件1320指定的值並指派給新「InsertNode」.bifsCommand元件之對應屬性。

程序「解碼SFNode」接著用以解譯Node Insertion命令1300之SFNode組件1324。

4. 在操作5430，數量「paramType」值與「1」比較。

5. 在操作5440，若數量「paramType」值係1，則遇到錯

誤。該值在有效sdsm資料流內不允許。解碼sdsm位元流之處理過程必須終止。

6. 在操作5450，數量「paramType」值與「2」比較。

7. 在操作5460，若數量「paramType」值係「2」1336，執行程序「解碼InsertIdxValue cmd」(Idx；索引)。

程序「解碼InsertIdxValue cmd」包含建立新「InsertIndexed Value」bifsCommand元件2840並將其附加至目前母元件2830，5310。決定由nodeID 1340、inFieldID 1344、insertionPosition 1348及Indexed Value Insertion命令1328之位置1320組件指定的值並指派給新「InsertIndexed Value」bifsCommand元件之對應屬性。

由組件「field value」1356代表之資料類型(布林、整數、字串等等)藉由數量「nodeID」、「inFieldID」值及動畫專家群第四階系統規格定義之表來決定。若用於一欄值之資料類型係「SFNode」，則程序「解碼SFNode」用於解譯索引值插入命令1328之「field value」組件1356，以及包含於「field value」組件內之BIFS節點名稱指派給新「InsertIndexed Value」元件的「value」屬性。否則，由「field value」組件1356代表之數值或字串值指派給新「InsertIndexed Value」bifsCommand元件的「value」屬性。

8. 在操作5470，若「paramType」值非「0」、「1」或「2」，其必須係「3」1368，因為該值由兩個資料位元決定。因此，該步驟結果總是「是」。

9. 在操作5480，執行程序「解碼InsertRoute cmd」。

程序「解碼InsertRoute cmd」包含建立新「InsertRoute」bifsCommand元件2840並將其附加至目前母元件2830，操作5310。決定由Route Insertion命令1360之isUpdateable 1372、routeID 1376、departureNodeID 1380、departureFieldID 1384、arrivalNodeID 1388及arrivalFieldID 1392組件指定的值並指派給新「InsertRoute」bifsCommand元件之對應屬性。

10. 在操作5490，程序「解碼InsertNode cmd」、「解碼InsertIdxValue cmd」或「解碼InsertRoute cmd」完成後，對sdsm位元流繼續處理數量「morecommands」5370值。

程序「解碼刪除命令」

如圖55所示，程序「解碼刪除命令」由以下步驟組成：

1. 在操作5500，sdsm位元流下兩個位元指派給數量「paramType」。

2. 在操作5510，數量「paramType」值與「0」比較。

3. 在操作5520，若數量「paramType」值係「0」1412，執行程序「解碼DeleteNode cmd」。

在操作5310，程序「解碼DeleteNode cmd」包含建立新「DeleteNode」bifsCommand元件2840並將其附加至目前母元件2830。決定由NodeDeletion命令1400之nodeID 1418組件指定的值並指派給新「DeleteNode」bifsCommand元件之「nodeID」屬性。

4. 在操作5530，數量「paramType」值與「1」比較。

5. 在操作5540，若數量「paramType」值係1，則遇到錯誤。該值在有效sdsm資料流內不允許。解碼sdsm位元流之

處理過程必須終止。

6. 在操作5550，數量「paramType」值與「2」比較。

7. 在操作5560，若數量「paramType」值係「2」1436，執行程序「解碼DeleteIdxValue cmd」。

在操作5310，程序「解碼DeleteIdxValue cmd」包含建立新「DeleteIndexed Value」bifsCommand元件2840並將其附加至目前母元件2830。決定由NodeDeletion命令1424之nodeID 1442、inFieldID 1448、deletionPosition 1454及位置1460組件指定的值並指派給新「DeleteIndexed Value」bifsCommand元件之對應屬性。

8. 在操作5570，若「paramType」值非「0」、「1」或「2」，其必須係「3」1478，因為該值由兩個資料位元決定。因此，該步驟結果總是「是」。

9. 在操作5580，執行程序「解碼DeleteRoute cmd」。

在操作5310，程序「解碼DeleteRoute cmd」包含建立新「DeleteRoute」bifsCommand元件2840並將其附加至目前母元件2830。決定由Route Deletion命令1466之routeID 1484組件指定的值並指派給新「DeleteRoute」bifsCommand元件之「routeID」屬性。

10. 在操作5590，程序「解碼InsertNode cmd」、「解碼InsertIdxValue cmd」或「解碼InsertRoute cmd」完成後，對sdsm位元流繼續處理數量「moreCommands」5370值。

程序「解碼替換命令」

如圖56所示，程序「解碼替換命令」由以下步驟組成：

1. 在操作5600，sdsm位元流下兩個位元指派給數量「paramType」。

2. 在操作5610，數量「paramType」值與「0」比較。

3. 在操作5620，若數量「paramType」值係「0」1508，執行程序「解碼ReplaceNode cmd」。

在操作5310，程序「解碼ReplaceNode cmd」包含建立新「ReplaceNode」 bifsCommand元件2840並將其附加至目前母元件2830。決定由Node Replacement命令1500之nodeID 1510組件指定的值並指派給新「ReplaceNode」bifsCommand元件之「nodeID」屬性。

程序「解碼SFNode」接著用以解譯Node Replacement命令1500之SFNode組件1514。

4. 在操作5630，數量「paramType」值與「1」比較。

5. 在操作5640，若數量「paramType」值係「1」1538，執行程序「解碼ReplaceField cmd」。

在操作5310，程序「解碼ReplaceField cmd」包含建立新「ReplaceField」 bifsCommand元件2840並將其附加至目前母元件2830。決定由Field Replacement命令1520之nodeID 1530及inFieldID 1534組件指定的值並指派給新「ReplaceField」bifsCommand元件之對應屬性。

由組件「field value」1538代表之資料類型(布林、整數、字串等等)藉由數量「nodeID」、「inFieldID」值及動畫專家群第四階系統規格定義之表來決定。若用於一欄位值之資料類型係「SFNode」，則程序「解碼SFNode」用於解譯Field

Replacement命令1520之「field value」組件1538，以及包含於「field value」組件內之BIFS節點名稱指派給新「ReplaceField」元件的「value」屬性。否則，由「field value」組件1538代表之數值或字串值指派給新「ReplaceField」bifsCommand元件的「value」屬性。

6. 在操作5650，數量「paramType」值與「2」比較。

7. 在操作5660，若數量「paramType」值係「2」1548，執行程序「解碼ReplaceIdxValue cmd」。

在操作5310，程序「解碼ReplaceIdxValue cmd」包含建立新「ReplaceIndexed Value」bifsCommand元件2840並將其附加至目前母元件2830。決定由Indexed Value Replacement命令1540之nodeID 1550、inFieldID 1554、replacementPosition 1558及位置1560組件指定的值並指派給新「ReplaceIndexed Value」bifsCommand元件之對應屬性。

由組件「field value」1564代表之資料類型(布林、整數、字串等等)藉由數量「nodeID」、「inFieldID」值及動畫專家群第四階系統規格定義之表來決定。若用於一欄位值之資料類型係「SFNode」，則程序「解碼SFNode」用於解譯Indexed Value Replacement命令1540之「field value」組件1564，以及包含於「field value」組件內之BIFS節點名稱指派給新「ReplaceIndexedValue」元件的「value」屬性。否則，由「field value」組件1564代表之數值或字串值指派給新「ReplaceIndexedValue」bifsCommand元件的「value」屬性。

8. 在操作5670，若「paramType」值非「0」、「1」或「2」，

其必須係「3」1578，因為該值由兩個資料位元決定。因此，該步驟結果總是「是」。

9. 在操作5680，執行程序「解碼ReplaceRoute cmd」。

在操作5310，程序「解碼ReplaceRoute cmd」包含建立新「ReplaceRoute」bifsCommand元件2840並將其附加至目前「commandFrame」元件2830。決定由Route Replacement命令 1570 之 routeID 1580 、 departureNodeID 1584 、departureFieldID 1588、arrivalNodeID 1590及arrivalFieldID 1594組件指定的值並指派給新「ReplaceRoute」bifsCommand元件之對應屬性。

10. 在操作5690，程序「解碼InsertNode cmd」、「解碼InsertIdxValue cmd」或「解碼InsertRoute cmd」完成後，在操作5370對sdsm位元流繼續處理數量「moreCommands」值。

程序「解碼SFNode」

圖17中總結SFNode資料結構1700、1710、1730之結構。每個SFNode資料結構包含於母資料結構內。母資料結構可係BIFS命令1120或另一SFNode結構。任一情形中，mp4bifs文件中會有對應母元件。BIFS命令母結構情形中，mp4bifs母元件將係bifsCommand元件2910。SFNode母結構情形中，mp4bifs母元件將係BIFS Node元件。

如圖57所示，程序「解碼SFNode」由以下步驟組成：

1. 在操作5700，sdsm位元流內下一位元指派給布林數量「isReused」1704、1712、1732且所得值與值「真」(1)比較。

2. 在操作5710，若「isReused」值係「真」(1)1704，則

新「ReusedNode」元件建立並附加至目前mp4bifs母元件。接著解譯「nodeIDRef」組件1708值且結果整數值指派給新「ReusedNode」元件之「nodeRef」屬性。用以指定值「nodeIDRef」之位元數藉由mp4bifs文件2800之「bifsConfig」元件2810的「nodeIdBits」屬性決定。該完成SFNode結構之處理。

3. 在操作5720，若「isReused」值係「0」(「假」)1712、1732，「localNodeType」1714、1734值從sdsm位元流擷取。代表該值之位元數藉由母結構之特性及動畫專家群第四階系統規格內定義之表決定。

母結構特性、「localNodeType」值及動畫專家群第四階系統規格內定義之表係用以決定用於該SFNode結構代表之特定類型BIFS節點的「nodeNumber」性質整數值。整數數量「nodeNumber」值及動畫專家群第四階系統規格內定義之表係用以決定字串數量「nodeName」值，其描述該SFNode結構代表之BIFS節點類型。具有由數量「nodeName」值指定之元件名稱的新BIFS節點元件建立並附加至mp4bifs母元件。

4. 在操作5730，sdsm位元流內下一位元指派給布林數量「isUpdateable」1716、1736且所得值與值「真」(1)比較。

5. 在操作5740，若「isUpdateable」值係「1」，則數量「nodeID」從sdsm位元流擷取。用以指定「nodeID」值之位元數藉由mp4bifs文件2800之「bifsConfig」元件2810的「nodeIdBits」屬性決定。所得值指派給操作5720建立之新

BIFS節點元件的「nodeID」屬性。

6. 在操作5750，相關「ReplaceScene」元件之「USENAMES」屬性的布林值與值「真」(1)比較。

7. 在操作5760，若「USENAMES」值係「真」，則空終止文數字元串1720、1740從SFNode結構擷取，結果字串值指派給操作5720建立之新BIFS節點元件的「name」屬性。

8. 在操作5770，sdsm位元流內下一位元指派給布林數量「maskAccess」1722、1742且所得值與值「真」(1)比較。

9. 在操作5780，若「maskAccess」值係1(「真」)，則SFNode結構具有如圖17B所示之形式。該情形中，遮罩存取位元1722後緊跟遮罩位元1726序列。遮罩位元數由數量「nFields」1728值代表。數量「nFields」值藉由操作5720之數量「nodeNumber」值及動畫專家群第四階系統規格定義之表決定。解譯「maskAccess」1722值後數量「fieldIndex」設定為零。數量「fieldIndex」值在解譯每個遮罩位元1726後遞增一。具有「1」值的每個遮罩位元1726後緊跟由數量「nodeNumber」值及數量「fieldIndex」值指定的性質欄位值，

每個性質欄位值藉由程序「解碼欄位值」決定。

10. 在操作5790，若「maskAccess」值係0「假」，則SFNode結構具有如圖17C所示之形式。該情形中，遮罩存取位元1722後緊跟endFlag位元1744。若endFlag位元值係「1」1750，則已經到達SFNode結構末端。若endFlag位元值係「0」1744，endFlag位元後緊跟fieldRef值1746及性質欄位(由數量

「nodeNumber」值及數量「fieldIndex」值指定)值1748。

每個性質欄位值藉由程序「解碼欄位值」決定。

程序「解碼欄位值」

每個SFNode結構之每個性質欄位具有特定欄位資料類型性質及特定欄位名稱串。欄位資料類型性質可係以下之一：布林、整數、浮點、向量2d、向量3d、旋轉、顏色、時間、字串、URL、節點或緩衝。每個性質欄位亦定義為SFField(單值欄位)或MFField(多值欄位)。

欄位資料類型性質、fieldName串及每個性質欄位之MFField/SFField性質藉由SFNode結構的「nodeNumber」數量5720值、性質欄位的「fieldIndex」或「fieldRef」數量值及動畫專家群第四階系統規格內定義之表決定。

SFField性質欄位情形中，用於代表欄位值之位元序列藉由欄位資料類型性質及動畫專家群第四階系統規格提供之資訊決定。

在MFField性質欄位情形中，欄位值藉由一組零或更多SFField值決定。包括MFField性質欄位之該組SFField值具有圖17D或圖17E說明的形式。任一情形中，第一位元1762、1782必須為零。下一位元指定布林數量「isList」1766、1786值。

若數量「isList」值係「1」，MFField性質欄位具有圖17D所示列表形式。該情形中，isList位元1766後緊跟endFlag位元1770。若endFlag位元值係「0」1770，endFlag位元後緊跟依據目前欄位資料類型格式化的SFField值1774。每個該

類SFField值1774後緊跟另一endFlag位元1770。具有值「1」1772之endFlag位元終止列表形式MFField。

若數量「isList」值係「1」，MFField性質欄位具有圖17D所示列表形式。該情形中，isList位元1766後緊跟5位元整數值「numBits」1790。「numBits」值後緊跟由numBits位元組成之整數值「nFields」1792。「nFields」值後緊跟nFields SFField值，各依據目前欄位資料類型格式化。

每個情形(SFField、列表形式MFField或向量形式MFField)中，每個SFField值轉換為字元串。每個該類字元串附加至具有藉由空白空間分隔的連續附加值之屬性串。在欄位資料類型整數、浮點、向量2d、向量3d、旋轉、顏色、時間情形中，標準裝置用以代表每個數值作為字元串。在欄位資料類型布林情形中，值「0」及「1」由字元串「假」及「真」代表。在欄位資料類型串及URL情形中，每個SFField值附以引號。

在欄位資料類型「節點」情形中，每個SFField值代表一SFNode結構1700、1710或1720。每個該類SFNode結構稱為一「子節點」。每個該類子節點藉由遞迴使用程序「解碼SFNode」解譯。為每個該類子節點決定之字串數量「nodeName」值附加至目前屬性串。此外，具有由字串數量「nodeName」值指定之元件名稱的新xml元件建立並附加至建立於程序「解碼SFNode」之步驟3的BIFS節點元件。

在欄位資料類型「緩衝」情形中，每個SFField值代表一sdsm BIFS命令1120。每個該類BIFS命令使用程序「解碼sdsm

大塊」之操作5330至操作5365解譯，目前節點元件用作母元件。與每個BIFS命令(例如「插入節點」)相關之命令名稱串附加至目前屬性串。此外，具有由命令名稱指定之元件名稱的新xml元件建立並附加至建立於程序「解碼SFNode」之步驟3的BIFS節點元件。

每個欄位資料值之解譯完成後，結果屬性串指派給對應BIFS節點元件之欄位名稱屬性。即，結果屬性串指派給具有由欄位名稱串(用於指定BIFS節點之指定性質欄位)提供之屬性名稱的屬性之屬性值。

形成QtInfo資料結構內代表之moov原子的xml代表。

根據QtInfo資料結構3100建立mp4file文件2300之第二步驟係建立包含於mp4二進制檔案700內的moov原子712、754之xml代表。如圖58所示，該步驟由以下操作組成：

1. 在操作5800，建立新「moov」元件2320並將其附加至mp4file文件2300頂部階層元件。

2. 在操作5805，處理QtInfo資料結構3100之MoovHeader成員3116。如圖31D所示，MoovHeader資料結構3170包括數量size 3173、version & Flags 3176、creationTime 3178、modifiedTime 3180、timeScale 3183、duration 3186及nextTrackID 3188值。每個此等值指派給新「moov」元件2320之類似名稱屬性。

3. 在操作5810，執行程序「處理Mp4fInitObjectDescr」。

4. 在操作5820，若成員UserData 3123值非零，執行程序「處理UserData資料結構」。

5. 在操作5825，將零值指派給整數數量「iTrack」。

6. 在操作5830，數量「iTrack」值與QtInfo成員「numTracks」3118值比較。

7. 在操作5835，若數量「iTrack」值等於QtInfo成員「numTracks」值，該步驟完成。在該點，建立中間xml檔案2250及2260之處理過程已完成。建立最終XMT-A文件之處理過程繼續「根據中間xml文件之XMT-A xml文件的建立」。

8. 在操作5840，若數量「iTrack」值不等於QtInfo成員「numTracks」值，新trak元件2350建立並將該附加至建立於操作5800之新moov元件2320。此係「目前trak元件」。

QtInfo成員「trak陣列」3120內項目「iTrack」指派給代表trak結構3200之數量「trakAtom」。由值「trakAtom」代表之trak結構係「目前trak結構」。

9. 在操作5850，執行程序「處理TrakHeader資料結構」。

10. 在操作5860，執行程序「處理TrakMedia資料結構」。

11. 在操作5870，若目前trak結構之TrackEdit成員3208非零，執行程序「處理TrackEdit資料結構」。

12. 在操作5880，若目前trak結構之TrackReference成員3210非零，執行程序「處理TrackReference資料結構」。

13. 在操作5890，將數量「iTrack」值遞增一，然後重複數量「iTrack」值與數量「numTracks」5830值之比較。

程序「處理Mp4InitObjectDescr」

程序「處理Mp4InitObjectDescr」由以下步驟組成：

1. 新「mp4fiods」元件2330建立並附加至「moov」元件2320。

2. Mp4fInitobjectDescr成員「ObjectDescrID」3138之整數值指派給新「mp4fiods」元件2360的「ObjectDescriptorID」屬性2370。

3. 成員「inlineProfiles」3146之布林值指派給新mp4fiods元件2360的「includeInlineProfilesFlag」屬性。

4. 若成員「urlFlag」3140之布林值係「真」，由成員「urlString」代表之字元串指派給「mp4fiods」元件2360之「url」屬性，完成Mp4fInitObjectDescr資料結構3130之處理。若成員「urlFlag」3140布林值係「假」，空字串指派給mp4fiods元件2360之「url」屬性。

5. 整數值根據Mp4fInitObjectDescr資料結構3130的「ProfileLevels」成員3148之對應成員(ODProfileLevel、sceneProfileLevel、audioProfileLevel、visualProfileLove1及graphicsProfileLevel)指派給mp4fiods元件2360之「ODProfile LevelIndication」、「sceneProfileLevelIndication」、「audio ProfileLevelIndication」、「visualProfileLevelIndication」及「graphicsProfileLevelIndication」屬性。

6. 新EsIdInc元件2380為每個numEsIdIncs 3153項目建立於Mp4fInitObjectDescr資料結構3130內的EsIdInc陣列3156內。每個新EsIdInc元件2380附加至mp4fiods元件2360。EsIdInc陣列3156內每個項目包括整數成員「trackID」。成員「trackID」值指派給對應新EsIdInc元件2380之「trackID」

屬性2390。

程序「處理UserData資料結構」

程序「處理UserData資料結構」由以下步驟組成：

1. 建立新「udta」元件2340並將其附加至新「moov」元件2320。

若UserData資料結構3280內成員numCprt 3284值大於零，為成員cprt陣列3286內每個項目重複以下步驟：

2. 建立新「cprt」元件並將其附加至新「udta」元件2340。

3. 將cprt陣列3286內選擇項目代表之cprt結構3290的version & Flags成員3294值指派給新「cprt」元件之「version」及「flags」屬性。

4. 將「cprt」結構3290內成員「language」3296值指派給新「cprt」元件之「language」屬性。

5. 根據cprt資料結構3290內成員「notice」3298建立新xml本文節點，並將該新本文節點附加至「cprt」元件。

程序「處理TrakHeader資料結構」

程序「處理TrakHeader資料結構」由以下步驟組成：

1. 將目前trak結構TrackHeader成員的成員「version」3216值指派給目前「trak」元件2600之「version」屬性。

2. 將目前trak結構TrackHeader成員的成員「flags」3216值指派給目前「trak」元件2600之「flags」屬性。

3. 將目前「trak」結構TrackHeader成員的成員「creationTime」3218值指派給目前「trak」元件2600之「creationTime」屬性。

4. 將目前「trak」結構TrackHeader成員的成員「modifiedTime」3220值指派給目前「trak」元件2600之「modifiedTime」屬性。

5. 將目前「trak」結構TrackHeader成員的成員「duration」3222值指派給目前「trak」元件2600之「duration」屬性。

6. 將目前「trak」結構TrackHeader成員的成員「trackID」3224值指派給目前「trak」元件2600之「trackID」屬性。

程序「處理TrakMedia資料結構」

程序「處理TrakMedia資料結構」由以下步驟組成：

1. 建立新「mdia」元件2604並將其附加至目前「trak」元件2600。

2. 執行程序「處理MediaHeader資料結構」。

3. 執行程序「處理Handler資料結構」。

4. 執行程序「處理MediaInfo資料結構」。

程序「處理TrackEdit資料結構」

程序「處理TrackEdit資料結構」由以下步驟組成：

1. 建立新edts元件2644並將其附加至目前trak元件。

若目前TrackEdit資料結構3208、3236之成員EditList 3242非零，執行以下額外步驟：

2. 建立新elst元件2648並將其附加至新edts元件2644。

3. 將目前TrackEdit資料結構3208、3236之EditList成員3242代表的EditList資料結構3244之成員version & Flags 3248值指派給新「elst」元件2648的「version」及「flags」屬性。

4. 將零值指派給數量「segment」以及將EditList資料結構3244內成員「numSegments」3250值指派給數量「numSegments」。數量「numSegments」值期望為「2」。

5. 為EditList資料結構3244之EditSegment陣列成員3252內每個numSegments項目重複以下步驟。每個此等項目由一Editsegment結構3256組成。

6. 建立新「segment」元件並將其附加至新「elst」元件2648。

7. 將EditSegment結構3256之成員「startTime」3258值指派給新「segment」元件之「startTime」屬性。

8. 將EditSegment結構3256之成員「duration」3260值指派給新「segment」元件之「duration」屬性。

9. 將EditSegment結構3256之成員「relativeRate」3262值指派給新「segment」元件之「relativeRate」屬性。

程序「處理TrackReference資料結構」

程序「處理TrackReference資料結構」由以下步驟組成：

1. 建立新「tref」元件2636並將其附加至目前trak元件2600。

若目前TrackReference資料結構3210、3264之成員Mpod3268非零，執行以下額外步驟：

2. 建立新「mpod」元件2640並將其附加至新「tref」元件2636。

3. 將零值指派給數量「entry」，將Mpod結構3268、3270內成員「numEntries」3274值指派給數量「numEntries」，以

及建立名為「trackIdList」的新字元串(作為空字串)。

4. 若數量「numEntries」值大於零，為Mpod結構3270內成員trackID陣列3276的每個numEntries項目重複以下三個步驟：

a.若數量「entry」值大於零，將空白空間附加至字元串「trackIdList」。

b.建立由trackID陣列3276內項目「entry」代表之整數值的字元串代表，以及將結果附加至字元串「trackIdList」。

c.將數量「entry」值遞增「1」。

5. 將字元串「trackIdList」值指派給mpod元件2640之trackID屬性。

程序「處理MediaHeader資料結構」

程序「處理MediaHeader資料結構」包含下列步驟：

1. 將MediaHeader結構3230、3300之成員「version」3306的值指派給目前「mdia」元件2604之「version」屬性。

2. 將MediaHeader結構3230、3300之成員「flags」3306的值指派給目前「mdia」元件2604之「flags」屬性。

3. 將MediaHeader結構3230、3300之成員「creationTime」3308的值指派給目前「mdia」元件2604之「creationTime」屬性。

4. 將MediaHeader結構3230、3300之成員「modifiedTime」3310的值指派給目前「mdia」元件2604之「modifiedTime」屬性。

5. 將MediaHeader結構3230、3300之成員「timeScale」3313

的值指派給目前「mdia」元件2604之「timeScale」屬性。

6. 將MediaHeader結構3230、3300之成員「duration」3316的值指派給目前「mdia」元件2604之「duration」屬性。

7. 將MediaHeader結構3230、3300之成員「language」3318的值指派給目前「mdia」元件2604之「language」屬性。

程序「處理Handler資料結構」

程序「處理Handler資料結構」包含下列步驟：

1. 建立一新「hdlr」元件2608且將其附加至目前「mdia」元件2604。

2. 將Handler結構3232、3320之成員「version」3326的值指派給新「hdlr」元件2608之「version」屬性。

3. 將Handler結構3232、3320之成員「flags」3326的值指派給新「hdlr」元件2608之「flags」屬性。

4. 將Handler結構3232、3320之成員「handlerType」3328的值指派給新「hdlr」元件2608之「handlerType」屬性。

5. 將Handler資料結構3232、3320之成員「name」3330的值指派給新「hdlr」元件2608之「name」屬性。

程序「處理MediaInfo資料結構」

程序「處理MediaInfo資料結構」包含下列步驟：

1. 建立一新「minf」元件2612且將其附加至目前「mdia」元件2604。

2. 執行程序「處理MediaDataInfo資料結構」。

3. 執行程序「處理MediaInfoHeader資料結構」。

4. 執行程序「處理SampleTables資料結構」。

程序「處理MediaDataInfo資料結構」

程序「處理MediaDataInfo資料結構」包含下列步驟：

1. 建立一新「dinf」元件2616且將其附加至目前「minf」元件2612。

2. 若MediaDataInfo結構3338、3350之成員DataRefTable 3358非零，則執行下列步驟：

3. 建立一新「dref」元件2620且將其附加至新「dinf」元件2616。

4. 將MediaDataInfo結構3338、3350中之成員DataRefTable 3358所代表的DataRefTable結構3360之成員「version」3366的值指派給新「dref」元件2620之「version」屬性。

5. 將DataRefTable結構3360之成員「flags」3366的值指派給新「dref」元件2620之「flags」屬性。

6. 對DataRefTable結構3360中之成員dataRef陣列3370中的各項目重複下列步驟：

7. 建立一新「urlData」元件2624且將其附加至新「dref」元件2620。

8. 將DataRef陣列3370中之該選擇項目所代表的dataRef結構3380之成員「version」3386的值指派給新「urlData」元件2624之「version」屬性。

9. 將dataRef結構3380之成員「flags」3386的值指派給新「urlData」元件2624之「flags」屬性。

10. 若dataRef結構3380之成員「flags」3386的值非「1」，

且成員「dataRefType」3388之值為「url」(即u-r-l-space)，則將成員「dataRefData」3390解譯為一字元串，且將該結果指派給新「urlData」元件2624之「location」屬性。

程序「處理MediaInfoHeader資料結構」

一給定trak結構中包含的MediaInfoHeader資料結構可為三類型之一，取決於與該trak結構相關的該媒體資料之資料流類型。

若目前「trak」結構係與一場景描述資料流(scene description stream；sdsm)或物件描述符資料流(object descriptor stream；odsm)相關，則MediaInfoHeader結構3343必定為一「Mpeg4MediaHeader」資料結構。此結構具有成員「size」、「version」及「flags」。在此情形中，程序「處理MediaInfoHeader資料結構」包含下列步驟：

1. 建立一新「nmhd」元件2632且將其附加至目前「minf」元件2616。

2. 將MediaInfoHeader結構3343之成員「version」的值指派給新「nmhd」元件2632之「version」屬性。

3. 將MediaInfoHeader結構3343之成員「flags」的值指派給新「nmhd」元件2632之「flags」屬性。

若目前「trak」結構係與一聲頻資料流相關，則MediaInfoHeader結構3343必定為一「SoundMediaHeader」資料結構。此資料結構具有成員「size」、「version」、「flags」及「balance」。在此情形中，程序「處理MediaInfoHeader資料結構」包含下列步驟：

1. 建立一新「smhd」元件2632且將其附加至目前「minf」元件。

2. 將MediaInfoHeader結構3343之成員「version」的值指派給新「smhd」元件2632之「version」屬性。

3. 將MediaInfoHeader結構3343之成員「flags」的值指派給新「smhd」元件2632之「flags」屬性。

4. 將MediaInfoHeader結構3343之成員「balance」的值指派給新「smhd」元件2632之「balance」屬性。

若目前「trak」結構係與一視覺資料流相關，則MediaInfoHeader結構3343必定為一「VideoMediaHeader」資料結構。此資料結構具有成員「size」、「version」、「flags」、「graphMode」、「opRed」、「opGreen」及「opBlue」。在此情形中，程序「處理MediaInfoHeader資料結構」包含下列步驟：

1. 建立一新「vmhd」元件2632且將其附加至目前「minf」元件。

2. 將MediaInfoHeader結構3343之成員「version」的值指派給新「vmhd」元件2632之「version」屬性。

3. 將MediaInfoHeader結構3343之成員「flags」的值指派給新「vmhd」元件2632之「flags」屬性。

4. 將MediaInfoHeader結構3343之成員「graphMode」的值指派給新「vmhd」元件2632之「transferMode」屬性。

5. 將MediaInfoHeader結構3343之成員「opRed」的值指派給新「vmhd」元件2632之「opColorRed」屬性。

6. 將MediaInfoHeader結構3343之成員「opGreen」的值指派給新「vmhd」元件2632之「opColorGreen」屬性。

7. 將MediaInfoHeader結構3343之成員「opBlue」的值指派給新「vmhd」元件2632之「opColorBlue」屬性。

程序「處理SampleTables資料結構」

程序「處理SampleTables資料結構」包含下列步驟：

1. 建立一新「stbl」元件2628且將其附加至目前「minf」元件2612。

2. 執行程序「處理SampleToChunkTable資料結構」。

3. 執行程序「處理ChunkOffsetTable資料結構」。

4. 執行程序「處理TimeToSampleTable資料結構」。

5. 執行程序「處理SampleSizeTable資料結構」。

6. 若SampleTables 3400之成員SyncSampleTable 3416的值非零，則執行程序「處理SyncSampleTable資料結構」。

7. 執行程序「處理SampleDescrTable資料結構」。

程序「處理SampleToChunkTable資料結構」

一SampleToChunkTable資料結構包含一樣本表資料結構3420，其中成員「table array」中之各項目包含一SampleToChunk結構3460。程序「處理SampleToChunkTable資料結構」包含下列步驟：

1. 建立一新「stsc」元件2656且將其附加至目前「stbl」元件2628、2652。

2. 將SampleTables成員SampleToChunkTable 3406指派給樣本表結構3420「SampleToChunkTable」。

3. 將SampleToChunkTable成員「version」3426的值指派給新「stsc」元件2656之「version」屬性。

4. 將SampleToChunkTable成員「flags」3426的值指派給新「stsc」元件2656之「flags」屬性。

5. 將SampleToChunkTable成員「numEntries」3428的值指派給該數量「numEntries」。將零值指派給該數量「entry」，且重複下列步驟，直到該數量「entry」之值等於該數量「numEntries」之值為止：

6. 建立一新「sampleToChunk」元件且將其附加至新「stsc」元件2656。

7. 將SampleToChunkTable成員「table array」3430中的項目「entry」指派給SampleToChunk結構3460「SampleToChunk TableEntry」。

8. 將SampleToChunkTableEntry成員「firstChunk」3463的值指派給新「sampleToChunk」元件之「firstChunk」屬性。

9. 將SampleToChunkTableEntry成員「numSamples」3466的值指派給新「sampleToChunk」元件之「numSamples」屬性。

10. 將SampleToChunkTableEntry成員「description」3468的值指派給新「sampleToChunk」元件之「sampleDesc」屬性。

11. 將該數量「entry」之值遞增「1」。

程序「處理ChunkOffsetTable資料結構」

一ChunkOffsetTable資料結構3408包含一樣本表資料結

構3420，其中成員「table array」中之各項目包含一整數序列。程序「處理ChunkOffsetTable資料結構」包含下列步驟：

1. 建立一新「stco」元件2664且將其附加至目前「stbl」元件2652。

2. 將SampleTables成員ChunkOffsetTable 3408指派給樣本表結構3420「ChunkOffsetTable」。

3. 將ChunkOffsetTable成員「version」的值指派給新「stco」元件2664之「version」屬性。

4. 將ChunkOffsetTable成員「flags」的值指派給新「stco」元件2664之「flags」屬性。

5. 將ChunkOffsetTable成員「numEntries」3428的值指派給該數量「numEntries」。將零值指派給該數量「entry」，且重複下列步驟，直到該數量「entry」之值等於該數量「numEntries」之值為止：

6. 建立一新「chunkOffset」元件且將其附加至新「stco」元件2664。

7. 將成員「table array」中之項目「entry」的值指派給該數量「offset」。

8. 將該數量「offset」的值指派給新「chunkOffset」元件之「offset」屬性。

9. 比較該數量「offset」的值與QtInfo成員「mdat array」3113中各項目3160之成員「start」3166及「size」3163的值，以決定與該指定偏移值相關的mdat原子706之索引「mdatId」的值。該數量「mdatId」之值係藉由「mdat array」3113中

該項目的索引來決定，對於該項目，該數量「offset」之值係大於成員「start」3166之值，而小於成員「start」3166與「size」3163之和。

10. 將mdat陣列項目「mdatId」之成員「start」3163的值指派給該數量「mdatStart」。將「mdatStart」之值增加「8」，從該數量「offset」之值中減去該結果，且將其結果指派給該數量「mdatOffset」。

11. 將該數量「mdatId」的值指派給新「chunkOffset」元件之「mdatId」屬性。

12. 將該數量「mdatOffset」的值指派給新「chunkOffset」元件之「mdatOffset」屬性。

13. 將該數量「entry」之值遞增「1」。

程序「處理TimeToSampleTable資料結構」

一TimeToSampleTable資料結構3410包含一樣本表資料結構3420，其中成員「table array」中之各項目包含一TimeToSample結構3470。程序「處理TimeToSampleTable資料結構」包含下列步驟：

1. 建立一新「stts」元件2660且將其附加至目前「stbl」元件。

2. 將SampleTables成員TimeToSampleTable 3410指派給樣本表結構3420「TimeToSampleTable」。

3. 將TimeToSampleTable成員「version」3426的值指派給新「stts」元件2660之「version」屬性。

4. 將TimeToSampleTable成員「flags」3426的值指派給新

「stts」元件2660之「flags」屬性。

5. 將TimeToSampleTable成員「numEntries」3428的值指派給該數量「numEntries」。將零值指派給該數量「entry」，且重複下列步驟，直到該數量「entry」之值等於該數量「numEntries」之值為止：

6. 建立一新「timeToSample」元件且將其附加至新「stts」元件2660。

7. 將TimeToSampleTable成員「table array」3430中的項目「entry」指派給TimeToSample結構3470「TimeToSample TableEntry」。

8. 將TimeToSampleTableEntry成員「count」3473的值指派給新「timeToSample」元件之「count」屬性。

9. 將TimeToSampleTableEntry成員「duration」3476的值指派給新「timeToSample」元件之「duration」屬性。

10. 將該數量「entry」之值遞增「1」。

程序「處理SampleSizeTable資料結構」

程序「處理SampleSizeTable資料結構」包含下列步驟：

1. 建立一新「stsz」元件2668且將其附加至目前「stbl」元件2652。

2. 將SampleTables成員SampleSizeTable 3413指派給SampleSizeTable結構3440「SampleSizeTable」。

3. 將SampleSizeTable成員「version」3446的值指派給新「stsz」元件2668之「version」屬性。

4. 將SampleSizeTable成員「flags」3446的值指派給新

「stsz」元件2668之「flags」屬性。

5. 將SampleSizeTable成員「numEntries」3453的值指派給該數量「numEntries」。

6. 將該數量「numEntries」的值指派給新「stsz」元件2668之「numSamples」屬性。

7. 若SampleSizeTable 3413、3440成員「sampleSize」3450的值非零，則將SampleSizeTable 3413、3440成員「sampleSize」3450的值指派給新「stsz」元件之「sizeForAll」屬性。在此情形中，此處理過程完成。

8. 若SampleSizeTable 3413、3440成員「sampleSize」3450的值為零，則將零值指派給該數量「entry」，且重複下列步驟，直到該數量「entry」之值等於該數量「numEntries」之值為止：

9. 建立一新「sampleSize」元件且將其附加至新「stsz」元件2668。

10. 將SampleSizeTable成員「table array」3456中項目「entry」的值指派給新「sampleSize」元件之「size」屬性。

11. 將該數量「entry」之值遞增「1」。

程序「處理SyncSampleTable資料結構」

一SyncSampleTable資料結構3416包含一樣本表資料結構3420，其中成員「table array」中之各項目包含一整數序列。程序「處理SyncSampleTable資料結構」包含下列步驟：

1. 建立一新「stss」元件2672且將其附加至目前「stbl」元件2652。

2. 將SampleTables成員SyncSampleTable 3416指派給樣本表結構3420「SyncSampleTable」。

3. 將SyncSampleTable成員「version」3426的值指派給新「stss」元件2672之「version」屬性。

4. 將SyncSampleTable成員「flags」3426的值指派給新「stss」元件2672之「flags」屬性。

5. 將SyncSampleTable成員「numEntries」3428的值指派給該數量「numEntries」。將零值指派給該數量「entry」，且重複下列步驟，直到該數量「entry」之值等於該數量「numEntries」之值為止：

6. 建立一新「syncSample」元件且將其附加至新「stss」元件2672。

7. 將SyncSampleTable成員「table array」3430中項目「entry」的值指派給新「syncSample」元件之「sample Number」屬性。

8. 將該數量「entry」之值遞增「1」。

程序「處理SampleDescrTable資料結構」

一SampleDescrTable資料結構3418包含一樣本表資料結構3420，其中成員「table array」中之各項目包含一SampleDescription結構3480。程序「處理SampleDescrTable資料結構」包含下列步驟：

1. 建立一新「stsd」元件2676且將其附加至目前「stbl」元件2652。

2. 將SampleTables成員SampleDescrTable 3418指派給樣

本表結構3420「SampleDescrTable」。

3. 將SampleDescrTable成員「version」3426的值指派給新「stsd」元件2676之「version」屬性。

4. 將SampleDescrTable成員「flags」3426的值指派給新「stsd」元件2676之「flags」屬性。

5. 將SampleDescrTable成員「numEntries」3428的值指派給該數量「numEntries」。將零值指派給該數量「entry」，且重複下列步驟，直到該數量「entry」之值等於該數量「numEntries」之值為止：

6. 將SampleDescrTable成員「table array」3430中的項目「entry」指派給樣本描述結構3480「SampleDescription」。

7. 若SampleDescription成員「dataFormat」的值對應於「mp4a」，則建立一新「mp4a」元件且將其附加至新「stsd」元件2676。若SampleDescription成員「dataFormat」的值對應於「mp4v」，則建立一新「mp4v」元件且將其附加至新「stsd」元件2676。若SampleDescription成員「dataFormat」的值對應於「mp4s」，則建立一新「mp4s」元件且將其附加至新「stsd」元件2676。所得之元件(「mp4a」、「mp4v」或「mp4s」)將稱作「mp4*」元件2680。

8. 將SampleDescription成員「dataReference」3488的值指派給新「mp4*」元件2680之「dataRefIndex」屬性。

9. 建立一新「esds」元件2684且將其附加至新「mp4*」元件2680。

10. 將SampleDescription成員「esdVersion」3490的值指

派給新「esds」元件2684之「version」屬性。

11. 將SampleDescription成員「esdData」3496的值指派給該數量「esdData」，其代表一位元組陣列之位址，且將零值指派給該數量「esdPos」。

12. 執行程序「解碼esdData陣列」。

13. 將該數量「entry」之值遞增「1」。

程序「解碼esdData陣列」

程序「解碼esdData陣列」包含下列步驟：

1. 將陣列「esdData」中之項目「esdPos」的值指派給該數量「EsDescTag」，且將該數量「esdPos」之值遞增「1」。

2. 利用程序「得到標籤大小」以獲得該數量「numBytes」之值。

3. 若該數量「EsDescTag」之值等於常數「ES_DescrTag」(3)之值，則執行下列步驟。否則退出。

4. 建立一新「ES_Descr」元件2688、2700且將其附加至目前「esds」元件2684。

5. 將該數量「numBytes」的值指派給新「ES_Descr」元件2688、2700之「size」屬性。

6. 將陣列「esdData」中之項目「esdPos」與「esdPos+1」的值合併成一16位元整數，將該結果指派給新「ES_Descr」元件2688之「ES_ID」屬性2700，且將該數量「esdPos」之值遞增「2」。

7. 將陣列「esdData」中之項目「esdPos」的值指派給該數量「flagsAndPriority」，然後將此值之高階5位元指派給

該數量「priority」，將低階3位元指派給該數量「flags」，且將該數量「esdPos」之值遞增「1」。

8. 將該數量「priority」的值指派給新「ES_Descr」元件2688、2700之「priority」屬性。

9. 將該數量「flags」的值指派給新「ES_Descr」元件2688、2700之「flags」屬性。

10. 若該數量「flags」之低階位元為「1」，則將值「真」指派給新「ES_Descr」元件2688之「OCRstreamFlag」屬性。

11. 重複下列步驟，直到該數量「numBytes」之值小於「1」為止：

12. 將陣列「esdData」中之項目「esdPos」的值指派給該數量「descrTag」，將該數量「numBytes」之值遞減「1」，且將該數量「esdPos」之值遞增「1」。

13. 利用程序「得到標籤大小」以決定該數量「descrSize」之值，且將該數量「numBytes」的值遞減用以指定此數量之值的位元組數。

14. 將該數量「numBytes」的值遞減該數量「descrSize」之值。

15. 若該數量「descrTag」之值等於常數「DecoderConfig DescrTag」(4)之值，則執行程序「處理解碼器組態描述符」。

16. 若該數量「descrTag」之值等於常數「SLConfigDescrTag」(6)之值，則執行程序「處理sl組態描述符」。

17. 否則(即該數量「descrTag」之值既不等於「Decoder

ConfigDescrTag」，亦不等於「SLConfigDescrTag」)，將該數量「esdPos」的值遞增該數量「infoSize」之值。

程序「處理解碼器組態描述符」

程序「處理解碼器組態描述符」包含下列步驟：

1. 建立一新「DecoderConfigDescriptor」元件2710且將其附加至目前「ES_Descr」元件2700。

2. 將該數量「descrSize」的值指派給新「DecoderConfig Descriptor」元件2710之「size」屬性。

3. 將陣列「esdData」中之項目「esdPos」的值指派給該數量「objectType」及新「DecoderConfigDescriptor」元件2710之「objectType」屬性，且將該數量「esdPos」之值遞增「1」。

4. 將陣列「esdData」中之項目「esdPos」的值指派給該數量「streamType」，且將該數量「esdPos」之值遞增「1」。

5. 若該數量「streamType」之第二位元(「2」位元)非零，則將值「真」指派給新「DecoderConfigDescriptor」元件2710之「upStream」屬性。

6. 將該數量「streamType」之值除以4(右移二位元)且將該結果指派給該數量「streamType」及新「DecoderConfig Descriptor」元件2710之「streamType」屬性。

7. 將陣列「esdData」中之下三項目合併成一24位元值，將該結果指派給新「DecoderConfigDescriptor」元件2710之「bufferSize」屬性，且將該數量「esdPos」之值遞增「3」。

8. 將陣列「esdData」中之下四項目合併成一32位元值，將該結果指派給新「DecoderConfigDescriptor」元件2710之

「maxBitrate」屬性，且將該數量「esdPos」之值遞增「4」。

9. 將陣列「esdData」中之下四項目合併成一32位元值，將該結果指派給新「DecoderConfigDescriptor」元件2710之「avgBitrate」屬性，且將該數量「esdPos」之值遞增「4」。

10. 將該數量「descrSize」之值遞減「13」。

11. 若該數量「descrSize」之值大於零，則重複下列步驟，直到該數量「descrSize」的值小於1為止：

12. 將陣列「esdData」中之下一項目指派給該數量「infoTag」，且將該數量「descrSize」之值遞減「1」。

13. 利用程序「得到標籤大小」以決定該標籤大小的值，其跟隨指派給該數量「infoTag」之值，且將該結果指派給該數量「infoSize」，且將該數量「descrSize」的值遞減用以指定此數量之值的位元組數。

14. 將該數量「descrSize」的值遞減該數量「infoSize」之值。

15. 若該數量「infoTag」之值等於常數「DecoderSpecific InfoTag」(5)之值，則執行程序「處理解碼器特定資訊」。否則，將該數量「esdPos」的值遞增該數量「infoSize」之值。

<u>程序「處理sl組態描述符」</u>

程序「處理sl組態描述符」包含下列步驟：

1. 建立一新「SLConfigDescriptor」元件2760且將其附加至目前「ES_Descr」元件2700。

2. 將陣列「esdData」中之項目「esdPos」的值指派給新

「SLConfigDescriptor」元件2760之「predefined」屬性，且將該數量「esdPos」之值遞增「1」。

程序「處理解碼器特定資訊」

跟隨一「DecoderSpecificInfoTag」之陣列「esdData」中的項目之內容取決於該等數量「streamType」及「objectType」的值。

若該數量「streamType」之值為「3」，則執行程序「處理BIFS組態」。

若該數量「streamType」之值為「4」，且該數量「objectType」之值為「108」，則執行程序「處理JPEG組態」。

若該數量「streamType」之值為「4」，且該數量「objectType」之值非「108」，則執行程序「處理視覺組態」。

若該數量「streamType」之值為「5」，則執行程序「處理聲頻組態」。

程序「處理BIFS組態」

程序「處理BIFS組態」包含下列步驟。該等步驟將「esdData」陣列中從項目「esdPos」開始的「infoSize」項目序列當作一位元序列。

1. 建立一新「BIFS_DecoderConfig」元件2720且將其附加至目前「DecoderConfigDescriptor」元件2710。

2. 若該數量「objectType」之值為「2」，則從陣列「esdData」中擷取下列數量之值(從項目「esdPos」開始)，且將所得之各值指派給新BIFS_DecoderConfig元件之具有類似名稱的屬性：

a. use3DMeshCoding(1位元)，

b. usePredictiveMFField(1位元)，

c. numNodeIdBits(5位元)，

d. numRouteIdBits(5位元)，

e. numPROTOIDbits(5位元)，

3. 否則(即該數量「objectType」之值非「2」)，從陣列「esdData」中擷取下列數量之值(從項目「esdPos」開始)，且將所得之各值指派給新BIFS_DecoderConfig元件之具有類似名稱的屬性：

a. numNodeIdBits(5位元)，

b. numRouteIdBits(5位元)，

4. 擷取陣列「esdData」中下一位元之值。若該結果為「0」，則將值「假」指派給新「BIFS_DecoderConfig」元件之「isCommandStream」屬性，且跳至步驟8。若該結果為「1」，則將值「真」指派給「isCommandStream」屬性，且執行下列步驟。

5. 擷取陣列「esdData」中下一位元之值。若該結果為「0」，則將值「假」指派給新「BIFS_DecoderConfig」元件之「pixel metric」屬性。否則，將值「真」指派給「pixel metric」屬性。

6. 擷取陣列「esdData」中下一位元之值。若該結果為「0」，則將值「假」指派給新「BIFS_DecoderConfig」元件之「hasSize」屬性，且跳至步驟8。若該結果為「1」，則將值「真」指派給「hasSize」屬性，且執行下列步驟。

7. 從陣列「esdData」中擷取下列數量之值(從項目「esdPos」開始)，且將所得之各值指派給新BIFS_DecoderConfig元件之具有類似名稱的屬性：

a. pixelWidth(16位元)，

b. pixelHeight(16位元)。

8. 將該數量「esdPos」的值遞增該數量「infoSize」之值。

程序「處理JPEG組態」

程序「處理JPEG組態」包含下列步驟：

1. 建立一新「JPEG_DecoderConfig」元件2730且將其附加至目前「DecoderConfigDescriptor」元件2710。

2. 將該數量「infoSize」的值指派給新「JPEG_DecoderConfig」元件2730之「size」屬性。

3. 將陣列「esdData」中之下二項目合併成一16位元值，將該結果指派給新「JPEG_DecoderConfig」元件2730之「headerLength」屬性，且將該數量「esdPos」之值遞增「2」。

4. 將陣列「esdData」中之下二項目合併成一16位元值，將該結果指派給新「JPEG_DecoderConfig」元件2730之「Xdensity」屬性，且將該數量「esdPos」之值遞增「2」。

5. 將陣列「esdData」中之下二項目合併成一16位元值，將該結果指派給新「JPEG_DecoderConfig」元件2730之「Ydensity」屬性，且將該數量「esdPos」之值遞增「2」。

6. 將陣列「esdData」中之項目「esdPos」的值指派給新「JPEG_DecoderConfig」元件2730之「numComponents」屬性，且將該數量「esdPos」之值遞增「1」。

程序「處理視覺組態」

程序「處理視覺組態」包含下列步驟：

1. 建立一新「VisualConfig」元件2740且將其附加至目前「DecoderConfigDescriptor」元件2710。

2. 將該數量「infoSize」的值指派給新「VisualConfig」元件2740之「size」屬性。

3. 將該數量「esdPos」的值遞增該數量「infoSize」之值。

程序「處理聲頻組態」

程序「處理聲頻組態」包含下列步驟：

1. 建立一新「AudioConfig」元件2750且將其附加至目前「DecoderConfigDescriptor」元件2710。

2. 將該數量「infoSize」的值指派給新「AudioConfig」元件2750之「size」屬性。

3. 將該數量「esdPos」的值遞增該數量「infoSize」之值。

基於該對中間xml文件建立一XMT-A xml文件

如圖59所示，建立一XMT-A文件100之過程包含下列步驟：

1. 於操作5900，使用標準xml裝置以建立一新的空XMT-A文件100。

2. 於操作5920，執行程序「建立XMT-A Header」。

3. 於操作5940，使用標準xml裝置以建立一新的空XMT-A Body元件120且將其附加至新XMT-A文件100。

4. 於操作5960，執行程序「建立OD命令之par元件」。

5. 於操作5980，執行程序「建立BIFS命令之par元件」。

步驟2、4及5將在下文描述。

為準備該等步驟，附屬於該mp4file文件之「moov」元件的「timeScale」屬性之值係指派給該數量「timeScale」。

程序「建立XMT-A Header元件」

如圖60所示，程序「建立XMT-A Header」包含下列步驟：

1. 於操作6000，使用標準xml裝置以建立一新Header元件110且將其附加至該XMT-A文件100。

2. 於操作6010，執行程序「建立InitialObjectDescriptor」。

3. 於操作6020，執行程序「建立Profiles元件」。

4. 於操作6030，執行程序「建立Descr元件」。

5. 於操作6040，使用標準xml裝置以選擇附屬於mp4file文件2300中之mp4fiods元件2330、2360的各元件。

6. 於操作6050，對附屬於mp4fiods元件2330、2360之各EsIdInc元件2380執行程序「處理EsIdInc元件」。

步驟2中之程序「建立InitialObjectDescriptor」包含下列操作：

a. 使用標準xml裝置以建立一新InitialObjectDescriptor元件130且將其附加至新Header元件110。

b. 使用標準xml裝置以定位包含於mp4file文件2300中之moov元件2320。

c. 使用標準xml裝置以定位包含於moov元件2320中之mp4fiods元件2330、2360。

d. 建立一新字串，其包含該前置「IODID:」，然後是mp4iods元件2360之objectDescriptorID屬性2370之值，且將

此字串之值指派給InitialObjectDescriptor元件130的ODID屬性。

步驟3中之程序「建立Profiles元件」包含下列操作：

a. 使用標準xml裝置以建立一新Profiles元件150且將其附加至新InitialObjectDescriptor元件130。

b. 將mp4fiods元件2360之includeInlineProfilesFlag屬性的值指派給Profiles元件150之includeInlineProfilesLevelFlag屬性。

c. 基於mp4fiods元件2360之對應屬性的值，將值指派給Profiles元件150之下列屬性：

i. sceneProfileLevelIndication

ii. ODProfileLevelIndication

iii. audioProfileLevelIndication

iv. visualProfileLevelIndication

v. graphicsProfileLevelIndication

指派給Profiles元件150之該等屬性的該等值可包含文數字串，其代表mp4fiods元件2360之對應屬性所代表的特定數值。例如，字串「無」可用以代表數值「-1」或「255」。動畫專家群第四階系統規格中定義的字串值可取代其他允許用於mp4fiods元件2360之屬性的數值。

步驟4中之程序「建立Descr元件」包含下列操作：

a. 使用標準xml裝置以建立一新Descr元件160且將其附加至新InitialObjectDescriptor元件130。

b. 使用標準xml裝置以建立一新esDescr元件170且將其

附加至新Descr元件160。

操作5包含使用標準xml裝置以選擇附屬於mp4file文件2300中之mp4fiods元件2320、2360的各元件2380。此步驟所選擇的該等元件預期係EsIdInc元件2380。對此步驟所識別的各EsIdInc元件2380執行程序「處理EsIdInc元件」。

程序「處理EsIdInc元件」

對附屬於mp4file文件2300中之mp4fiods元件2360的各EsIdInc元件2380執行下列操作：

1. 於操作6100，將EsIdInc元件2380之trackID屬性2390的值指派給該數量「trackID」。

2. 於操作6110，使用標準xml裝置以找出附屬於mp4file文件2300中之moov元件2320的trak元件2350，其中該trak元件2350之trackID屬性匹配該數量「trackID」之值。

3. 於操作6120，使用標準xml裝置以找出包含於前一步驟所識別的trak元件2350、2600中之ES_Descr元件2688。

4. 於操作6130，使用標準xml裝置以建立一新ES_Descriptor元件180、190且將其附加至esDescr元件170。

5. 於操作6140，比較步驟「2」中決定之該字串數量「trackType」的值與字串「sdsm」。

6. 於操作6150，若該字串數量「trackType」之值為「sdsm」，則執行程序「建立場景描述資料流之DecoderConfigDescriptor」。

7. 於操作6160，執行程序「建立BIFSConfig」。

8. 於操作6170，若該字串數量「trackType」之值非

「sdsm」，則比較該字串數量「trackType」的值與字串「odsm」。

9. 於操作6180，若該字串數量「trackType」之值為「odsm」，則執行程序「建立物件描述符資料流之DecoderConfigDescriptor」。

10. 於操作6190，執行程序「從tref元件中取得trak之mpodTrackIDs」。

步驟2包含下列於該選擇trak元件得到識別之後執行的操作：

a. 使用標準xml裝置以找出附屬於該指定trak元件2600之hdlr元件2608。

b. 將此hdlr元件之「name」屬性的值指派給字串數量「trackName」。

c. 將此hdlr元件之「handlerType」屬性的值指派給字串數量「trackType」。此數量之值預期係「sdsm」或「odsm」。

d. 將字串「IOD」與該數量「trackName」的值及字串「ESID:1」連接以建立一新字串數量「ES_ID」，且將該結果指派給新ES_Descriptor元件180、190之「ES_ID」屬性。

e. 建立一新decConfigDescr元件646且將其附加至ES_Descriptor元件180、190、640。

f. 建立一新DecoderConfigDescriptor元件650且將其附加至decConfigDescr元件646。

g. 建立一新slConfigDescr元件660且將其附加至ES_Descriptor元件640。

h. 建立一新SLConfigDescriptor元件666且將其附加至slConfigDescr元件660。

i. 建立一新「predefined」元件且將其附加至SLConfigDescriptor元件666。

j. 將值「2」指派給「predefined」元件之「value」屬性。

建立場景描述資料流之DecoderConfigDescriptor

1. 使用標準xml裝置以找出mp4file文件2300中之該選擇trak元件2600中包含的ES_Descr元件2688、2700中包含之DecoderConfigDescriptor元件2710。

2. 將DecoderConfigDescriptor元件2710之「avgBitrate」及「maxBitrate」屬性的值指派給XMT-A文件100中之新DecoderConfigDescriptor元件650的對應屬性。

3. 將DecoderConfigDescriptor元件2710之「bufferSize」屬性的值指派給新DecoderConfigDescriptor元件650之「bufferSizeDB」屬性。

4. 基於mp4file文件2300中的DecoderConfigDescriptor元件2710之「streamType」屬性的數值，將一字串值指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「streamType」屬性。mp4file文件2300中的DecoderConfigDescriptor元件2710之「streamType」屬性的值預期與圖62A之第一行6200中的該等項目之一匹配。將圖62A之第二行6220中的對應項目指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「streamType」屬性。

在此情形中，mp4file文件2300中的DecoderConfig

Descriptor元件2710之streamType屬性的值預期係「3」，且將值「SceneDescription」指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「streamType」屬性。

5. 基於mp4file文件2300中的DecoderConfigDescriptor元件2710之「objectType」屬性的數值，將一字串值指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「objectTypeIndication」屬性。mp4file文件2300中的DecoderConfigDescriptor元件2710之「objectType」屬性的值預期與圖62B之第一行6240中的該等項目之一匹配。將圖62B之第二行6260中的對應項目指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「objectTypeIndication」屬性。

6. 建立一新decSpecificInfo元件656、680且將其附加至DecoderConfigDescriptor元件650。

<u>建立BIFSConfig</u>

程序「建立BIFS Config」包含下列步驟：

1. 建立一新BIFSConfig元件686且將其附加至新decSpecificInfo元件680。

2. 使用標準xml裝置以找出附屬於DecoderConfigDescriptor元件2710之BIFS_DecoderConfig元件2720。

3. 將BIFS_DecoderConfig元件2720之「nodeIdBits」屬性的值指派給該數量nodeIdBits以及BIFSConfig元件686之「nodeIDbits」屬性。

4. 將BIFS_DecoderConfig元件2720之「routeIdBits」屬性

的值指派給BIFSConfig元件686之「routeIDbits」屬性。

5. 若該數量「v2Config」之值為「真」，則執行以下三步驟：

a. 將BIFS_DecoderConfig元件2720之「protoIdBits」屬性的值指派給BIFSConfig元件686之「PROTOIDbits」屬性。

b. 將BIFS_DecoderConfig元件2720之「use3DMesh Coding」屬性的值指派給BIFSConfig元件686之「use3DMesh Coding」屬性。

c. 將BIFS_DecoderConfig元件2720之「usePredictive MFField」屬性的值指派給BIFSConfig元件686之「usePredictiveMFField」屬性。

6. 若BIFS_DecoderConfig元件2710之「commandStream」屬性的值為「真」，則執行下列步驟：

a. 建立一新「commandStream」元件690且將其附加至BIFSConfig元件686。

b. 將BIFS_DecoderConfig元件2710之「pixelMetric」屬性的值指派給commandStream元件686之「pixelMetric」屬性。

c. 建立一新「size」元件696且將其附加至commandStream元件690。

d. 將BIFS_DecoderConfig元件2710之「pixelHeight」屬性的值指派給「size」元件696之「pixelHeight」屬性。

e. 將BIFS_DecoderConfig元件2710之「pixelWidth」屬性的值指派給「size」元件696之「pixelWidth」屬性。

建立物件描述符資料流之DecoderConfigDescriptor

1. 使用標準xml裝置以找出mp4file文件2300中之該選擇trak元件2600中包含的ES_Descr元件2688、2700中包含之DecoderConfigDescriptor元件2710。

2. 將DecoderConfigDescriptor元件2710之「avgBitrate」及「maxBitrate」屬性的值指派給XMT-A文件中之新DecoderConfigDescriptor元件650的對應屬性。

3. 將mp4file文件中之trak原子中的DecoderConfigDescriptor元件之「bufferSize」屬性的值指派給新DecoderConfigDescriptor元件650之「bufferSizeDB」屬性。

4. 基於mp4file文件2300中的DecoderConfigDescriptor元件2710之「streamType」屬性的數值，將一字串值指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「streamType」屬性。mp4file文件2300中的DecoderConfigDescriptor元件2710之「streamType」屬性的值預期與圖62A之第一行6200中的該等項目之一匹配。將圖62A之第二行6220中的對應項目指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「streamType」屬性。

在此情形中，mp4file文件2300中的DecoderConfigDescriptor元件2710之streamType屬性的值預期係「1」，且將值「ObjectDescriptor」指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「streamType」屬性。

5. 基於mp4file文件2300中的DecoderConfigDescriptor元件2710之「objectType」屬性的數值，將一字串值指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之

「objectTypeIndication」屬性。mp4file文件2300中的DecoderConfigDescriptor元件2710之「objectType」屬性的值預期與圖62B之第一行6240中的該等項目之一匹配。將圖62B之第二行6260中的對應項目指派給XMT-A文件100中的新DecoderConfigDescriptor元件650之「objectTypeIndication」屬性。

從tref元件中取得trak之mpodTrackIDs

程序「從tref元件中取得trak之mpodTrackIDs」包含下列步驟：

1. 使用標準xml裝置以找出附屬於該選擇trak元件2600之tref元件2636。

2. 使用標準xml裝置以找出附屬於該選擇tref元件2636之mpod元件2640。

3. 將「mpod」元件2640之「trackID」屬性的值指派給該數量「mpodTrackIds」。

程序「建立OD命令之par元件」

代表Object Descriptor命令元件之XMT-A「par」元件140係源自代表物件描述符資料流(odsm)之mp4file文件2300的元件2460。XMT-A文件100中代表的各Object Descriptor命令亦可包含ES_Descriptor元件630，其各係源自mp4file文件2300中之一trak元件2350、2600所包含的一ES_Descriptor元件2688。

為準備建立該等「par」元件，利用該字串數量「mpodTrackIds」以建立一trackId值陣列「pMpodTrackId」。

若該對應的mp4檔案存在一物件描述符資料流(odsm)，則mpodTrackIds之值係於建立XMT-A Header元件110期間指派。若未指派一值給此數量，則不存在物件描述符資料流，不會產生「par」元件以代表Object Descriptor命令元件，且此步驟完成。若對應的mp4檔案包含除sdsm及odsm之外的一或多媒體資料流，則該數量「mpodTrackIds」之值將包含一整數trackID值序列，且陣列「pMpodTrackId」之各元件將代表此序列之對應成員。

如圖63所示，用以建立代表Object Descriptor命令之「par」元件的程序包含下列步驟：

1. 於操作6300，使用標準xml裝置以找出mp4file文件2300中之所有odsm元件2420。為此，首先使用標準xml裝置以選擇mp4file文件中之各mdat元件2310。然後使用標準xml裝置以選擇各mdat元件2400中之各odsm元件2420。

2. 於操作6310，使用標準xml裝置以選擇各odsm元件2460中之各odsmChunk元件2470。

3. 於操作6320，使用標準xml裝置以選擇各odsmChunk元件2500中之各odsmSample元件2510。

4. 於操作6330，使用標準xml裝置以建立一新的空「par」元件140且將其附加至XMT-A Body元件120。

5. 於操作6340，將odsmSample元件2510、2520之「time」屬性2515、2525屬性的值除以該數量「timeScale」之值，且將該結果指派給新「par」元件140之「begin」屬性。

6. 於操作6350，使用標準xml裝置以選擇各odsmSample

元件2520中包含的各odsm-command元件2530。

7. 於操作6360，將各odsm-command元件2530之元件名稱與字串「ObjectDescrUpdate」相比較。

8. 於操作6370，若一odsm-command元件2530之元件名稱係「ObjectDescrUpdate」，則執行程序「建立ObjectDescrUpdate元件」。

9. 於操作6380，若一odsm-command元件2530之元件名稱非「ObjectDescrUpdate」，則將該odsm-command元件2530之元件名稱與字串「ObjectDescrRemove」相比較。

10. 於操作6390，若一odsm-command元件2530之元件名稱係「ObjectDescrRemove」，則執行程序「建立ObjectDescrRemove元件」。

建立ObjectDescriptorUpdate元件

程序建立ObjectDescriptorUpdate元件包含下列步驟：

1. 使用標準xml裝置以建立一新「ObjectDescriptorUpdate」元件220且將其附加至目前「par」元件210。

2. 使用標準xml裝置以建立一新「OD」元件230且將其附加至目前「ObjectDescriptorUpdate」元件220。

3. 使用標準xml裝置以找出附屬於ObjectDescr Update odsm-command元件2540之「ObjectDescriptor」元件2550。

4. 使用標準xml裝置以建立一新「ObjectDescriptor」元件240、600且將其附加至目前「OD」元件230。

5. 將ObjectDescriptor元件2550之ODID屬性2555的值指派給該字串數量「ODID」。

6. 將該字串數量「ODID」值與前置「ODID:」連接，且將該結果指派給新ObjectDescriptor元件240、606之objectDescriptorID屬性，其中「ODID」代表「object DescriptorID」。

7. 使用標準xml裝置以建立一新「Descr」元件610且將其附加至新「ObjectDescriptor」元件600。

8. 使用標準xml裝置以建立一新「esDescr」元件620且將其附加至新「Descr」元件610。

9. 使用標準xml裝置以建立一新「ES_Descriptor」元件630且將其附加至目前「esDescr」元件620。

10. 使用標準xml裝置以選擇附屬於ObjectDescriptor元件2550之「EsIdRef」元件2560。

11. 將「EsIdRef」元件2560之「EsId」屬性2565的值指派給該數量「EsId」。

12. 使用標準xml裝置以建立一新「decConfigDescr」元件646且將其附加至新「ES_Descriptor」元件630、640。

13. 使用標準xml裝置以建立一新「DecoderConfig Descriptor」元件650且將其附加至新「decConfigDescr」元件640。

14. 將陣列「pMpodTrackId」中之項目(EsId-1)的值指派給該數量「mpodTrackID」。例如，若「EsId」之值為「1」，則「mpodTrackID」的值係藉由陣列「pMpodTrackId」中項目「0」(第一項目)決定。

15. 使用標準xml裝置以選擇trak元件2350，其中trackID

屬性之值與該數量「mpodTrackID」的值匹配。

16. 使用標準xml裝置以選擇附屬於該選擇trak元件2350、2600之hdlr元件2608。

17. 將該選擇hdlr元件2608之「name」屬性的值指派給該字串數量「trackName」。

18. 將該選擇hdlr元件2608之「handlerType」屬性的值指派給該字串數量「trackType」。

19. 將字串「ESID:」與該數量「trackName」之值及該數量「EsId」之值連接以建立一新字串數量，且將該結果指派給新「ES_Descriptor」元件640之「ES_ID」屬性636。

20.使用標準xml裝置以建立一新「slConfigDescr」元件660且將其附加至目前「ES_Descriptor」元件640。

21. 使用標準xml裝置以建立一新「SLConfigDescriptor」元件666且將其附加至新「slConfigDescr」元件660。

22. 使用標準xml裝置以建立一新「predefined」元件且將其附加至新「SLConfigDescriptor」元件666。

23. 將值「2」指派給新「predefined」元件之「value」屬性。

24. 執行程序「建立媒體資料流之DecoderConfig Descriptor元件」。

25. 執行程序「建立StreamSource元件」。

建立媒體資料流之DecoderConfigDescriptor元件

程序「建立媒體資料流之DecoderConfigDescriptor元件」包含下列步驟：

1. 使用標準xml裝置以選擇程序「建立ObjectDescriptor Update元件」之步驟15中選擇之trak元件2350中包含的DecoderConfigDescriptor元件2710。

2. 將一字串值指派給程序「建立ObjectDescriptorUpdate元件」之步驟13中建立的新DecoderConfigDescriptor元件650之屬性「streamType」。此字串值係藉由圖62A之第二行6220中的一項目決定。此項目對應於該第一行6200上之該項目，其與該選擇DecoderConfigDescriptor 2710之「streamType」屬性的數值匹配。

3. 將一字串值指派給程序「建立ObjectDescriptorUpdate元件」之步驟13中建立的新DecoderConfigDescriptor元件650之屬性「objectTypeIndication」。此字串值係藉由圖62B之第二行6260中的一項目決定。此項目對應於該第一行6240上之該項目，其與該選擇DecoderConfigDescriptor 2710之「objectType」屬性的數值匹配。

4. 將該選擇DecoderConfigDescriptor元件2710之「avgBitrate」及「maxBitrate」屬性的值指派給XMT-A文件中之新DecoderConfigDescriptor元件650的對應屬性。

5. 將該選擇DecoderConfigDescriptor元件2710之「bufferSize」屬性的值指派給XMT-A文件中之新Decoder ConfigDescriptor元件650的「bufferSizeDB」屬性。

6. 若該選擇DecoderConfigDescriptor元件2710包含一附屬JPEG_DecoderConfig元件2730、VisualConfig元件2740或AudioConfig元件2750，則執行下列步驟：

a. 使用標準xml裝置以建立一新「decSpecificInfo」元件656且將其附加至新「DecoderConfigDescriptor」元件650。

b. 使用標準xml裝置以建立一新「DecoderSpecificInfo」元件且將其附加至目前「decSpecificInfo」元件656。

c. 將值「auto」指派給新DecoderSpecificInfo元件之「type」屬性。

建立StreamSource元件

程序「建立StreamSource元件」包含下列步驟：

1. 使用標準xml裝置以選擇包含於mp4file文件2300中之各mdat元件2310。

2. 使用標準xml裝置以選擇包含於各mdat元件2310中之各mediaFile元件2430、2480。

3. 將各選擇之mediaFile元件2480之「trackID」屬性的值指派給該數量「mdatTrackID」。

4. 將該數量「mdatTrackID」的值與該數量「mpodTrackID」之目前值相比較。

5. 若該數量「mdatTrackID」之值與該數量「mpodTrackID」之目前值匹配，則執行下列步驟：

a. 使用標準xml裝置以建立一新「StreamSource」元件670且將其附加至目前「ES_Descriptor」元件640。

b. 將該選擇之mediaFile元件2480之「name」屬性的值與前置「file:///」連接，且將該結果指派給新「StreamSource」元件670之「url」屬性。

建立ObjectDescriptorRemove元件

程序「建立ObjectDescriptorRemove元件」包含下列步驟：

1. 使用標準xml裝置以建立一新「ObjectDescriptor Remove」元件250且將其附加至目前「par」元件210。

2. 將odsm-sample元件2530中的「ObjectDescriptor Remove」元件2570之「ODID」屬性的值與前置「ODID:」連接，且將該結果指派給新「ObjectDescriptorRemove」元件250之「objectDescriptorId」屬性。XMT-A「ObjectDescriptor Remove」元件之「objectDescriptorId」屬性係表示為圖2B中之「ODID」。

程序「建立BIFS命令之par元件」

如圖64所示，程序「建立BIFS命令之par元件」包含下列步驟：

1. 於操作6400，使用標準xml裝置以選擇mp4bifs文件2800中之bifsConfig元件2810。將「nodeIdBits」屬性的值指派給該數量「numNodeIdBits」屬性。建立名稱為「pNodeNumber」之一整數陣列及名稱為「pNodeIdName」之一字串陣列。該等陣列之各陣列中的元件數目係指定為2的「numNodeIdBits」次冪。

2. 於操作6410，使用標準xml裝置以選擇mp4bifs文件2800中之各「commandFrame」元件2820。

3. 於操作6420，使用標準xml裝置以建立一新「par」元件140且將其附加至XMT-A Body元件120。

4. 於操作6430，將commandFrame元件2820之「time」屬性的值指派給「par」元件140之「begin」屬性。

5. 於操作6440，利用目前commandFrame元件2820執行程序「構造節點數及節點名稱陣列」。

6. 於操作6450，利用目前commandFrame元件2820執行程序「將mp4bifscommandFrame元件轉換成XMT-A元件」。

構造節點數及節點名稱陣列

圖65概述程序「構造節點數及節點名稱陣列」。此程序包含下列步驟：

1. 於操作6500，使用標準xml裝置以選擇目前mp4bifs commandFrame元件2830所擁有的各附屬元件。該等附屬元件各對應一bifsCommand元件2840、2910。

2. 於操作6510，使用標準xml裝置以選擇目前bifsCommand元件所擁有的各附屬元件。若一選擇之附屬元件對應於一bifsCommand元件，則此程序遞迴式重複。否則，該選擇之附屬元件必定對應於一mp4bifs Node元件，且執行下列步驟。

3. 於操作6520，將該選擇之mp4bifs Node元件2920、3000的元件名稱與字串「ReusedNode」相比較。若該選擇之mp4bifs Node元件2920、3000的元件名稱為「ReusedNode」，則繼續選擇(若有)下一mp4bifs Node元件6510、6590。

4. 於操作6530，若該選擇之mp4bifs Node元件2920、3000的元件名稱非「ReusedNode」，則使用標準xml裝置以決定對該選擇之mp4bifs Node元件2920、3000的「NodeId」屬性是否已指定一值。

5. 於操作6540，若對該選擇之mp4bifs Node元件2920、

3000的「NodeId」屬性已指定一值，則將該「NodeId」屬性的數值指派給該整數數量「nodeId」。

6. 於步驟6550，將該選擇之mp4bifs Node元件2920、3000的名稱與動畫專家群第四階規格說明中定義的一節點名稱陣列之各項目相比較。將該動畫專家群第四階Node名稱陣列中之匹配項目的索引指派給該整數數量nodeNumber。將該整數數量nodeNumber的值指派給陣列pNodeNumber中之項目nodeId。

7. 於操作6560，使用標準xml裝置以決定對該選擇之mp4bifs Node元件2920、3000的「name」屬性是否已指定一值。

8. 於操作6570，若對該選擇之mp4bifs Node元件2920、3000的「name」屬性已指定一值，則將此「name」屬性的值指派給陣列pNodeIdName中之項目nodeId。

9. 於操作6580，使用標準xml裝置以選擇該選擇之mp4bifs Node元件2920、3000所擁有的各附屬元件3030、3070。對於對應一bifsCommand元件3070之各附屬元件，遞迴式重複步驟2之6510。

10. 於步驟6590，對於對應一mp4bifs Node元件3030之各附屬元件，遞迴式重複步驟3之6520。

<u>將mp4bifs commandFrame元件轉換成XMT-A元件</u>

以下程序係用以將一mp4bifs commandFrame元件2830之內容轉換成XMT-A BIFS命令元件300、310、320。在此情形中，「目前mp4bifs元件」係於6410選擇之mp4bifs command

Frame元件2830，且「XMT-A母元件」係於6420建立之XMT-A「par」元件。

使用該同一程序將一mp4bifs Conditional Node 3040轉換成XMT-A BIFS命令元件300、310、320。在此情形中，「目前mp4bifs元件」係附屬於一mp4bifs命令元件2910或一mp4bifs Node元件3000之一mp4bifs Conditional Node元件3040，且「XMT-A母元件」係一XMT-A命令元件300、310、320或一XMT-A CommandBuffer nodeField元件410。

1. 使用標準xml裝置以選擇附屬於目前mp4bifs元件之各mp4bifs元件。該等附屬元件各係以所述下列步驟之一處理：

2. 若該選擇之附屬元件的元件名稱為「InsertNode」，則執行程序「將一mp4bifs InsertNode元件轉換成XMT-A元件」。

3. 若該選擇之附屬元件的元件名稱為「InsertIndexedValue」，則執行程序「將一mp4bifs Insert IndexedValue元件轉換成XMT-A元件」。

4. 若該選擇之附屬元件的元件名稱為「InsertRoute」，則執行程序「將一mp4bifs InsertRoute元件轉換成XMT-A元件」。

5. 若該選擇之附屬元件的元件名稱為「DeleteNode」，則執行程序「將一mp4bifs DeleteNode元件轉換成XMT-A元件」。

6. 若該選擇之附屬元件的元件名稱為「DeleteIndexed Value」，則執行程序「將一mp4bifs DeleteIndexed Value元件

轉換成XMT-A元件」。

7. 若該選擇之附屬元件的元件名稱為「DeleteRoute」，則執行程序「將一mp4bifs DeleteRoute元件轉換成XMT-A元件」。

8. 若該選擇之附屬元件的元件名稱為「ReplaceNode」，則執行程序「將一mp4bifs ReplaceNode元件轉換成XMT-A元件」。

9. 若該選擇之附屬元件的元件名稱為「ReplaceIndexed Value」，則執行程序「將一mp4bifs ReplaceIndexed Value元件轉換成XMT-A元件」。

10. 若該選擇之附屬元件的元件名稱為「ReplaceField」，則執行程序「將一mp4bifs ReplaceField元件轉換成XMT-A元件」。

11. 若該選擇之附屬元件的元件名稱為「Replace Route」，則執行程序「將一mp4bifs ReplaceRoute元件轉換成XMT-A元件」。

12. 若該選擇之附屬元件的元件名稱為「ReplaceScene」，則執行程序「將一mp4bifs ReplaceScene元件轉換成XMT-A元件」。

將一mp4bifs InsertNode元件轉換成XMT-A元件

程序「將一mp4bifs InsertNode元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Insert」元件300且將其附加至XMT-A母元件。

2. 處理parentId屬性：

a. 將mp4bifs「InsertNode」元件之「parentId」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

c. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

d. 將該數量「nodeIdName」的值指派給XMT-A「Insert」元件300之「atNode」屬性。

3. 處理insertionPosition屬性：

a. 將mp4bifs「InsertNode」元件之「insertionPosition」屬性的值指派給該字串數量「position」。

b. 若該數量「position」之值為「0」，則將mp4bifs「InsertNode」元件之「position」屬性的值指派給XMT-A「Insert」元件300之「position」屬性。

c. 若該數量「position」之值為「2」，則將值「BEGIN」指派給XMT-A「Insert」元件300之「position」屬性。

d. 若該數量「position」之值為「3」，則將值「END」指派給XMT-A「Insert」元件300之「position」屬性。

4. 處理附屬mp4bifs Node元件：

a. 使用標準xml裝置以選擇目前bifsCommand元件2910所擁有的各附屬Node元件2920。

b. 對各選擇之附屬Node元件2920執行程序「將一mp4bifs

Node元件轉換成一XMT-A Node元件」。

將一mp4bifs InsertIndexed Value元件轉換成XMT-A元件

程序「將一mp4bifs InsertIndexed Value元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Insert」元件300且將其附加至目前XMT-A母元件。

2. 處理nodeId屬性：

a. 將mp4bifs「InsertNode」元件之「nodeId」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeNumber中之項目nodeId的值指派給該整數數量「nodeNumber」。

c. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

d. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

e. 將該數量「nodeIdName」的值指派給XMT-A「Insert」元件之「atNode」屬性。

3. 處理「inFieldName」及值屬性：

a. 將mp4bifs「InsertIndexed Value」元件之「inFieldName」屬性的值指派給XMT-A「Insert」元件之「atField」屬性。

b. 將mp4bifs「InsertIndexed Value」元件之「value」屬性的值指派給XMT-A「Insert」元件之「value」屬性。

4. 處理insertionPosition屬性：

a. 將mp4bifs「InsertIndexed Value」元件之「insertion Position」屬性的值指派給該字串數量「position」。

b. 若該數量「position」之值為「0」，則將mp4bifs「Insert Node」元件之「position」屬性的值指派給XMT-A「Insert」元件300之「position」屬性。

c. 若該數量「position」之值為「2」，則將值「BEGIN」指派給XMT-A「Insert」元件300之「position」屬性。

d. 若該數量「position」之值為「3」，則將值「END」指派給XMT-A「Insert」元件300之「position」屬性。

5. 處理附屬mp4bifs Node元件：

a. 使用標準xml裝置以選擇目前bifsCommand元件2910所擁有的各附屬Node元件2920。

b. 對各選擇之附屬Node元件2920執行程序「將一mp4bifs Node元件轉換成一XMT-A Node元件」。

將一mp4bifs InsertRoute元件轉換成XMT-A元件

程序「將一mp4bifs InsertRoute元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Insert」元件300且將其附加至目前XMT-A母元件。

2. 建立一新XMT-A「Route」元件340且將其附加至新XMT-A「Insert」元件300。

3. 處理routeId屬性：

a. 將mp4bifs「InsertRoute」元件之「routeId」屬性的值指派給該字串數量「iRoute」。

b. 將mp4bifs「InsertRoute」元件之「name」屬性的值指派給該字串數量「routeName」。

c. 若該字串數量「routeName」之值為非空值且至少包含一字元，則將該數量「routeName」指派給XMT-A「ROUTE」元件340之「DEF」屬性。否則，若該字串數量「iRoute」之值為非空值且至少包含一字元，則將該數量「iRoute」的值附加至前置「route:」，且將該結果指派給XMT-A「ROUTE」元件340之「DEF」屬性。

4. 基於mp4bifs「InsertRoute」元件，執行程序「將一mp4bifs Route元件之屬性轉換成一XMT-A ROUTE元件之屬性」。

將一mp4bifs DeleteNode元件轉換成XMT-A元件

程序「將一mp4bifs DeleteNode元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Delete」元件310且將其附加至目前XMT-A母元件。

2. 處理nodeId屬性：

a. 將mp4bifs「DeleteNode」元件之「nodeId」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

c. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

d. 將該數量「nodeIdName」的值指派給XMT-A「Delete」元件310之「atNode」屬性。

將一mp4bifs DeleteIndexed Value元件轉換成XMT-A元件

程序「將一mp4bifs DeleteIndexed Value元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Delete」元件310且將其附加至目前XMT-A母元件。

2. 處理nodeId屬性：

a. 將mp4bifs「DeleteIndexed Value」元件之「nodeId」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeNumber中之項目nodeId的值指派給該整數數量「nodeNumber」。

c. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

d. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

e. 將該數量「nodeIdName」的值指派給XMT-A「Delete」元件310之「atNode」屬性。

3. 處理inField屬性：

a. 將mp4bifs「DeleteIndexed Value」元件之「inField」屬性的值指派給該整數數量「field」。

b. 將用於具有索引「nodeNumber」之BIFS節點的動畫專家群第四階規格說明中定義之一欄位名稱陣列中的項目

「field」之值指派給該字串數量「fieldName」。

c. 將該數量「fieldName」的值指派給XMT-A「Delete」元件310之「atField」屬性。

4. 處理「deletionPosition」屬性：

a. 將mp4bifs「DeleteIndexed Value」元件之「deletionPosition」屬性的值指派給該字串數量「position」。

b. 若該數量「position」之值為「0」，則將mp4bifs「InsertNode」元件之「position」屬性的值指派給XMT-A「Delete」元件310之「position」屬性。

c. 若該數量「position」之值為「2」，則將值「BEGIN」指派給XMT-A「Delete」元件310之「position」屬性。

d. 若該數量「position」之值為「3」，則將值「END」指派給XMT-A「Delete」元件310之「position」屬性。

將一mp4bifs DeleteRoute元件轉換成XMT-A元件

程序「將一mp4bifs DeleteRoute元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Delete」元件310且將其附加至目前XMT-A母元件。

2. 將mp4bifs「DeleteRoute」元件之「routeId」屬性的值指派給該字串數量「routeId」。

3. 將該數量「routeId」之值附加至前置「route:」，且將該結果指派給XMT-A「Delete」元件310之「atRoute」屬性。

將一mp4bifs ReplaceNode元件轉換成XMT-A元件

程序「將一mp4bifs ReplaceNode元件轉換成XMT-A元件」

包含下列步驟：

1. 建立一新XMT-A「Replace」元件320且將其附加至目前XMT-A母元件。

2. 處理nodeId屬性：

a. 將mp4bifs「ReplaceNode」元件之「nodeId」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

c. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

d. 將該數量「nodeIdName」的值指派給XMT-A「Replace」元件320之「atNode」屬性。

3. 處理附屬mp4bifs Node元件：

a. 使用標準xml裝置以選擇目前bifsCommand元件2910所擁有的各附屬Node元件2920。

b. 對各選擇之附屬Node元件2920執行程序「將一mp4bifs Node元件轉換成一XMT-A Node元件」。

將一mp4bifs ReplaceIndexed Value元件轉換成XMT-A元件

程序「將一mp4bifs ReplaceIndexed Value元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Replace」元件320且將其附加至目前XMT-A母元件(圖3C)。

2. 處理nodeId屬性：

a. 將mp4bifs「ReplaceIndexed Value」元件之「nodeId」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeNumber中之項目nodeId的值指派給該整數數量「nodeNumber」。

c. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

d. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

e. 將該數量「nodeIdName」的值指派給XMT-A「Replace」元件320之「atNode」屬性。

3. 處理inFieldName及值屬性：

a. 將mp4bifs「ReplaceIndexed Value」元件之「inFieldName」屬性的值指派給XMT-A「Replace」元件之「atField」屬性。

b. 將mp4bifs「ReplaceIndexed Value」元件之「value」屬性的值指派給XMT-A「Replace」元件320之「value」屬性。

4. 處理replacementPosition屬性：

a. 將mp4bifs「ReplaceIndexed Value」元件之「replacement Position」屬性的值指派給該字串數量「position」。

b. 若該數量「position」之值為「0」，則將mp4bifs「InsertNode」元件之「position」屬性的值指派給XMT-A「Replace」元件320之「position」屬性。

c. 若該數量「position」之值為「2」，則將值「BEGIN」

指派給XMT-A「Replace」元件320之「position」屬性。

d. 若該數量「position」之值為「3」，則將值「END」指派給XMT-A「Replace」元件320之「position」屬性。

5. 處理附屬mp4bifs Node元件：

a. 使用標準xml裝置以選擇目前bifsCommand元件2910所擁有的各附屬Node元件2920。

b. 對各選擇之附屬Node元件2920執行程序「將一mp4bifs Node元件轉換成一XMT-A Node元件」。

將一mp4bifs ReplaceField元件轉換成XMT-A元件

程序「將一mp4bifs ReplaceField元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Replace」元件320且將其附加至目前XMT-A母元件。

2. 處理nodeId屬性：

a. 將mp4bifs「ReplaceField」元件之「nodeId」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeNumber中之項目nodeId的值指派給該整數數量「nodeNumber」。

c. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

d. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

e. 將該數量「nodeIdName」的值指派給XMT-A「Replace」

元件之「atNode」屬性。

3. 處理inFieldName及值屬性：

a. 將mp4bifs「ReplaceField」元件之「inFieldName」屬性的值指派給XMT-A「Replace」元件之「atField」屬性。

b. 將mp4bifs「ReplaceField」元件之「value」屬性的值指派給XMT-A「Replace」元件之「value」屬性。

4. 處理附屬mp4bifs Node元件：

a. 使用標準xml裝置以選擇目前bifsCommand元件2910所擁有的各附屬Node元件2920。

b. 對各選擇之附屬Node元件2920執行程序「將一mp4bifs Node元件轉換成一XMT-A Node元件」。

將一mp4bifs ReplaceRoute元件轉換成XMT-A元件

程序「將一mp4bifs ReplaceRoute元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Replace」元件320且將其附加至目前XMT-A母元件。

2. 將mp4bifs「ReplaceRoute」元件之「routeId」屬性的值指派給該字串數量「routeId」。

3. 將該數量「routeId」之值附加至前置「route:」，且將該結果指派給XMT-A「Replace」元件之「atRoute」屬性。

4. 基於mp4bifs「ReplaceRoute」元件，執行程序「將一mp4bifs Route元件之屬性轉換成一XMT-A ROUTE元件之屬性」。

將一mp4bifs ReplaceScene元件轉換成XMT-A元件

程序「將一mp4bifs ReplaceScene元件轉換成XMT-A元件」包含下列步驟：

1. 建立一新XMT-A「Replace」元件320且將其附加至目前XMT-A母元件。

2. 建立一新XMT-A「Scene」元件370且將其附加至新XMT-A「Replace」元件320。

3. 將ReplaceScene元件2930之「USENAMES」屬性的值指派給該字串數量「useNames」。若該數量useNames之值為「真」，則將該數量useNames的值指派給新XMT-A Scene元件370之「USENAMES」屬性。

4. 使用標準xml裝置以選擇ReplaceScene元件2930所擁有的各附屬元件。對各選擇之附屬元件執行下列操作：

5. 若所選擇之附屬元件之名稱為「Routes」2950，則執行下列步驟。否則，執行程序「將一mp4bifs Node元件轉換成一XMT-A Node元件」。

6. 使用標準xml裝置以選擇該選擇「Routes」元件2950所擁有的各附屬元件。各附屬元件預期係一「Route」元件2960。對各「Route」元件2960執行下列步驟：

7. 建立一新XMT-A「ROUTE」元件390且將其附加至新XMT-A「Scene」元件370。

8. 處理可選「routeId」屬性：

a. 將該選擇之「Route」元件2960之「routeId」屬性的值指派給該字串數量「iRoute」。

b. 將mp4bifs「Route」元件2960之「name」屬性的值指

派給該字串數量「routeName」。

c. 若該字串數量「routeName」之值為非空值且至少包含一字元，則將該數量「routeName」指派給XMT-A「ROUTE」元件390之「DEF」屬性。否則，若該字串數量「iRoute」之值為非空值且至少包含一字元，則將該數量「iRoute」的值附加至前置「route:」，且將該結果指派給XMT-A「ROUTE」元件390之「DEF」屬性。

9. 基於mp4bifs「Route」元件2960，執行程序「將一mp4bifs Route元件之屬性轉換成一XMT-A ROUTE元件之屬性」。

將一mp4bifs Route元件之屬性轉換成一XMT-A ROUTE元件之屬性。

程序「將一mp4bifs Route元件之屬性轉換成一XMT- A ROUTE元件之屬性」包含下列步驟：

此程序所稱的mp4bifs Route元件可為一mp4bifs「Route」元件2960、一mp4bifs「InsertRoute」bifsCommand元件2910或一mp4bifs「ReplaceRoute」bifsCommand元件2910。

1. 處理departureNode值：

a. 將mp4bifs Route元件之「departureNode」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeNumber中之項目nodeId的值指派給該整數數量「nodeNumber」。

c. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

d. 若該字串數量「nodeIdName」的值為空值或無內容(零

長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

e. 將該數量「nodeIdName」的值指派給XMT-A「ROUTE」元件360或390之「fromNode」屬性。

2. 處理departureField值：

a. 將mp4bifs Route元件之「departureField」屬性的值指派給該整數數量「field」。

b. 將用於具有索引「nodeNumber」之BIFS節點的動畫專家群第四階規格說明中定義之一欄位名稱陣列中的項目「field」之值指派給該字串數量「fieldName」。

c. 將該數量「fieldName」的值指派給XMT-A「ROUTE」元件360或390之「fromField」屬性。

3. 處理arrivalNode值：

a. 將mp4bifs Route元件之「arrivalNode」屬性的值指派給該整數數量「nodeId」。

b. 將陣列pNodeNumber中之項目nodeId的值指派給該整數數量「nodeNumber」。

c. 將陣列pNodeIdName中之項目nodeId的值指派給該字串數量「nodeIdName」。

d. 若該字串數量「nodeIdName」的值為空值或無內容(零長度)，則將該數量「nodeId」之值附加至前置「nodeId:」，且將該結果指派給該數量「nodeIdName」。

e. 將該數量「nodeIdName」的值指派給XMT-A「ROUTE」元件360或390之「toNode」屬性。

4. 處理arrivalField值：

a. 將mp4bifs Route元件之「arrivalField」屬性的值指派給該整數數量「欄位」。

b. 將用於具有索引「nodeNumber」之BIFS節點的動畫專家群第四階規格說明中定義之一欄位名稱陣列中的項目「field」之值指派給該字串數量「fieldName」。

c. 將該數量「fieldName」的值指派給XMT-A「ROUTE」元件360或390之「toField」屬性。

將一mp4bifs Node元件轉換成一XMT-A Node元件

各mp4bifs Node元件3000係藉由附屬於一特定XMT-A母元件之一新XMT-A Node元件400代表。此XMT-A母元件之身分係藉由該mp4bifs Node元件3000所屬的該mp4bifs元件來決定。

a. 若mp4bifs Node元件2920係附屬於一InsertNode bifs Command元件2910或InsertIndexedValue bifsCommand元件2910，則該XMT-A母元件係藉由該對應的XMT-A「Insert」元件300來定義。

b. 若mp4bifs Node元件2920係附屬於一ReplaceNode bifs Command元件2910、ReplaceField bifsCommand元件2910或ReplaceIndexedValue bifsCommand元件2910，則該XMT-A母元件係藉由該對應的XMT-A「Replace」元件320來定義。

c. 若mp4bifs Node元件2940係附屬於一ReplaceScene bifs Command元件2930，則該XMT-A母元件係藉由該對應的XMT-A「Scene」元件370來定義。

d. 若mp4bifs Node元件3030係附屬於另一mp4bifs Node元件3000或InsertIndexed Value bifsCommand元件2910，則該XMT-A母元件係藉由該對應的XMT-A「nodeField」元件410來定義。

若該mp4bifs Node元件的元件名稱為「ReusedNode」3080，則執行程序「為一mp4bifs ReusedNode元件建立XMT-A Node元件」。否則執行下列步驟。

1. 決定該整數數量「nodeNumber」之值。此值係藉由藉由動畫專家群第四階規格說明所定義的一節點名稱陣列中之項目數來決定，其中該指示項目之值與目前mp4bifs Node元件3000之NodeName匹配。

2. 建立一新XMT-A Node元件400且將其附加至XMT-A母元件，該XMT-A Node元件400具有與mp4bifs Node元件3000相同的元件名稱(NodeName)。此新XMT-A Node元件係定義為「目前XMT-A Node元件」。

3. 若需要，則決定該「DEF」屬性之值：

a. 將mp4bifs Node元件3000之「name」屬性3016的值指派給該字串數量「nodeIdName」。

b. 若該數量「nodeIdName」之值為非空值且此數量所代表的該字串至少具有一字元，則將該數量「nodeIdName」的值指派給新XMT-A Node元件400之「DEF」屬性。

c. 若該數量「nodeId Name」之值為空值或此數量所代表的該字串具有少於一的字元，則將mp4bifs Node元件3000之「NodeId」屬性3010的值指派給該字串數量「nodeId」。

說明無意徹底詳盡或限制本發明成為所揭示之精確形式，而且顯然可按照上述教導來修改及變化。例如，XMT-A檔案之精確定義可能隨著時間推移而變化或發展。同樣，動畫專家群第四階中間格式之精確定義可能隨著時間推移而變化或發展。本文所描述的本發明並不侷限於上文所引用文件中指定的特定定義。因此，本發明之原理亦可應用於其他無關的資料結構。動畫專家群第四階系統規格中亦定義SFNode資料結構之其他延伸形式。對於熟悉技術人士，延伸上述具體實施例中描述的本發明以涵蓋該等情形之裝置係顯而易見。因此，選擇且描述所揭露之具體實施例係為了最佳解釋本發明之原理及其實際應用，從而讓其他熟悉技術人士可在預計適合特定用途的各種具體實施例及各種變更中最佳地利用本發明。隨附申請專利範圍係要解釋為包含除先前技術所限制者之外的本發明之其他替代具體實施例。

【圖式簡單說明】

圖1A顯示本發明一項具體實施例所用之一示範性XMT-A文件。

圖1B顯示一示範性XMT-A初始物件描述符。

圖2A顯示一示範性XMT-A par元件。

圖2B顯示示範性XMT-A odsm命令元件。

圖3A顯示示範性XMT-A插入命令。

圖3B顯示一示範性XMT-A刪除命令。

圖3C顯示示範性XMT-A替換命令。

圖4顯示一示範性XMT-A BIFS節點元件。

圖5A顯示一示範性XMT-A BIFS節點。

圖5B顯示一示範性XMT-A再用BIFS節點。

圖6A顯示一示範性XMT-A ObjectDescriptor。

圖6B顯示一示範性XMT-A ES_Descriptor。

圖6C顯示sdsm之一示範性DecoderSpecificInfo(BIFS)。

圖7A顯示本發明一項具體實施例所產生之一示範性mp4二進制檔案。

圖7B顯示一示範性mdat原子。

圖7C顯示一示範性大塊。

圖7D顯示一示範性moov原子。

圖8A顯示一示範性mp4檔案iods原子。

圖8B顯示一示範性Mp4fInitObjectDescr。

圖8C顯示一示範性ES_ID_Inc。

圖9A顯示一示範性trak原子。

圖9B顯示一示範性樣本表原子。

圖10A顯示一示範性二進制ES descriptor。

圖10B顯示一示範性解碼器組態描述符。

圖10C顯示一示範性解碼器特定資訊描述符。

圖10D顯示一示範性二進制SL組態描述符。

圖11A顯示一示範性sdsm二進制大塊。

圖11B顯示一示範性sdsm命令框。

圖12A顯示一示範性BIFS插入命令。

圖12B顯示一示範性BIFS刪除命令。

圖12C顯示一示範性BIFS替換命令。

圖12D顯示一示範性BIFS場景替換命令。

圖13A顯示一示範性Node插入命令。

圖13B顯示一示範性Indexed Value插入命令。

圖13C顯示一示範性Route插入命令。

圖14A顯示一示範性Node刪除命令。

圖14B顯示一示範性Indexed Value刪除命令。

圖14C顯示一示範性Route刪除命令。

圖15A顯示一示範性Node替換命令。

圖15B顯示一示範性Field替換命令。

圖15C顯示一示範性Indexed Value替換命令。

圖15D顯示一示範性Route替換命令。

圖16顯示一示範性BIFS場景。

圖17A顯示一示範性SFNode(再用)。

圖17B顯示一示範性SFNode(遮罩節點)。

圖17C顯示一示範性SFNode(列表節點)。

圖17D顯示一示範性MFField(列表形式)。

圖17E顯示一示範性MFField(向量形式)。

圖18A顯示一示範性Routes(列表形式)。

圖18B顯示一示範性Routes(向量形式)。

圖18C顯示一示範性Route。

圖19A顯示一示範性odsm二進制大塊。

圖19B顯示一示範性odsm二進制樣本。

圖20A顯示一示範性ObjectDescriptorUpdate命令。

圖20B顯示一示範性ObjectDescriptorRemove命令。

圖21A顯示一示範性二進制物件描述符。

圖21B顯示一示範性二進制EsIdRef描述符。

圖22顯示本發明設想的一示範性動畫專家群第四階中間至XMT-A檔案轉換器。

圖23A顯示一示範性mp4file文件。

圖23B顯示一示範性mp4fiods元件。

圖24A顯示一示範性mdat元件。

圖24B顯示一示範性sdsm元件。

圖24C顯示一示範性odsm元件。

圖24D顯示一示範性mediaFile元件。

圖25A顯示一示範性odsmChunk元件。

圖25B顯示一示範性odsmSample元件。

圖25C顯示一示範性odsm-命令元件。

圖26A顯示一示範性trak元件。

圖26B顯示一示範性stbl元件。

圖27顯示一示範性ES_Descr。

圖28A顯示一示範性mp4bifs文件。

圖28B顯示一示範性mp4bifs commandFrame元件。

圖29A顯示一示範性mp4bifs bifsCommand元件。

圖29B顯示一示範性mp4bifs ReplaceScene元件。

圖30A顯示一示範性mp4bifs初始節點元件。

圖30B顯示一示範性mp4bifs條件節點元件。

圖30C顯示一示範性mp4bifs再用節點元件。

圖31A顯示一示範性QtInfo結構。

圖31B顯示一示範性Mp4fInitObjectDescr元件。

圖31C顯示一示範性mdat結構。

圖31D顯示一示範性MoovHeader元件。

圖32A顯示一示範性磁軌結構。

圖32B顯示一示範性TrackHeader結構。

圖32C顯示一示範性TrackMedia結構。

圖32D顯示一示範性TrackEdit結構。

圖32E顯示一示範性EditList結構。

圖32F顯示一示範性EditSegment結構。

圖32G顯示一示範性TrackReference結構。

圖32H顯示一示範性Mpod結構。

圖32I顯示一示範性UserData結構。

圖32J顯示一示範性cprt結構。

圖33A顯示一示範性MediaHeader結構。

圖33B顯示一示範性Handler結構。

圖33C顯示一示範性MediaInfo結構。

圖33D顯示一示範性MediaDataInfo結構。

圖33E顯示一示範性DataRefTable結構。

圖33F顯示一示範性dataRef結構。

圖34A顯示一示範性SampleTables結構。

圖34B顯示一示範性樣本表結構。

圖34C顯示一示範性SampleSizeTable結構。

圖34D顯示一示範性SampleToChunk結構。

圖34E顯示一示範性TimeToSample結構。

圖34F顯示一示範性SampleDescription結構。

圖35顯示一示範性mp4檔案解譯流程圖。

圖36顯示一示範性mdat原子處理流程圖。

圖37顯示一示範性moov原子處理流程圖。

圖38顯示一示範性trak原子處理流程圖。

圖39顯示一示範性磁軌媒體原子處理流程圖。

圖40顯示一示範性媒體資訊原子處理流程圖。

圖41顯示一示範性樣本表原子處理流程圖。

圖42顯示形成一QtInfo資料結構代表的媒體資料原子之xml代表的示範性流程圖。

圖43顯示一示範性媒體資料處理流程圖。

圖44顯示找出一stream type之示範性流程圖。

圖45顯示找出一解碼器組態描述符之示範性流程圖。

圖46顯示一示範性得到標籤大小流程圖。

圖47顯示自一odsm資料流建立一xml代表的示範性流程圖。

圖48顯示一示範性解碼odsm大塊流程圖。

圖49顯示一示範性解碼OdUpdate流程圖。

圖50顯示一示範性decodeEsIdRef程序流程圖。

圖51顯示建立媒體檔案及代表mdat原子的示範性流程圖。

圖52顯示一示範性SdsmDataToXml程序流程圖。

圖53顯示一示範性解碼sdsm大塊程序流程圖。

圖54顯示一示範性解碼插入命令程序流程圖。

圖55顯示一示範性解碼刪除命令程序流程圖。

圖56顯示一示範性解碼替換命令程序流程圖。

圖57顯示一示範性解碼SFNode程序流程圖。

圖58建立一moov原子之xml代表的示範性流程圖。

圖59顯示建立一XMT-A文件的示範性流程圖。

圖60顯示建立一XMT-A Header的示範性流程圖。

圖61顯示處理EsIdInc元件的示範性流程圖。

圖62A顯示一示範性streamType轉換表。

圖62B顯示一示範性objectType至objectTypeIndication表。

圖63顯示建立代表Object Descriptor命令之「par」元件的示範性流程程序。

圖64顯示建立代表commandFrame元件之「par」元件的示範性流程圖。

圖65顯示建立節點數目值及nodeId名稱表的示範性流程圖。

【圖式代表符號說明】

100	XMT-A檔案/XMT-A文件
110	Header元件
120	Body元件
130	InitialObjectDescriptor元件
140、200	「par」元件
150	Profiles元件
160	Descr元件

170	「esDescr」元件
180、190	「ES_Descriptor」子元件
210	「par-child」元件
220	ObjectDescriptorUpdate元件
230	OD子元件
240、600	ObjectDescriptor子元件
250	ObjectDescriptorRemove元件
300	Insert元件
310	Delete元件
320	Replace元件
330	「xmtaBifsNode」子元件
340	「ROUTE」子元件
350	「xmtaBifsNode」子元件
360	「ROUTE」子元件
370	「Scene」子元件
380	「xmtaTopNode」子元件
390	ROUTE子元件
400	xmtaBifsNode元件(的一般結構)
410	nodeField元件
420	xmtaBifsNode元件
500	NodeName標籤/初始節點元件
510	可選屬性「DEF」
520	欄位屬性
530	欄位值子元件/起始標籤

540	欄位值子元件/結束標籤
550	BIFS節點元件
560	結束標籤
570	NodeName標籤/再用節點元件
580	「USE」屬性
590	值
600	ObjectDescriptor元件
606	「ObjectDescriptorID」(ODID)屬性
610	Descr子元件
620	esDescr子元件
630	ES_Descriptor子元件
636	「ES_ID」屬性
640	ES_Descriptor元件(結構)
646	decConfigDescr子元件
650	DecoderConfigDescriptor子元件
656	decSpecificInfo子元件
660	slConfigDescr子元件
666	SLConfigDescriptor子元件
670	StreamSource子元件
680	decSpecificInfo元件
686	BIFSConfig子元件
690	commandStream元件
696	「size」元件
700	mp4二進制檔案

706	媒體資料原子
712、754	「moov」原子
718	mdat原子
724	原子大小值
730	四位元組atomId「mdat」
736	「chunks(大塊)」
742	大塊
748	「樣本」
760	原子大小值
766	四位元組atomId「moov」
772	「mvhd」(moov標頭)原子
778	「iods」(初始物件描述符)原子
784	「udta」(使用者資料)原子
790	「trak」原子
800	iods原子
804	原子大小值
808	四位元組atomId「iods」
812	8位元version(版本)值
816	24位元flag(旗標)值
820	Mp4fInitObjDescr資料結構
824	Mp4fInitObjDescr資料結構
828	位元組MP4_IODTAG值
832	位元組數
836	10位元ObjectDescriptorID

840、844	旗標位元
848	reserved(保留)位元
852、856、860、	設定檔等級指示值
864、868	
872、876	ES_ID_Inc物件
880	位元組ES_ID_IncTag值
884	位元組數
888	32位元ES_ID值
900	trak原子
903	原子大小值
906	四位元組atomId「trak」
910	「tkhd」(磁軌標頭)原子
912	「mdia」(媒體)原子
915	「mdhd」(媒體標頭)原子
918	「hdlr」(處置器)原子
920	「minf」(媒體資訊)原子
924	dinf原子
927	dref原子
933	「stbl」(樣本表)原子
936	媒體資訊標頭原子
940	「tkhd」(磁軌參考)原子
942	mpod原子
945	「edts」(編輯列表)原子
948	elst原子

950	stbl原子
954	原子大小值
957	四位元組atomId「stbl」
960	「stsc」(樣本至大塊表)原子
963	「stts」(時間至樣本表)原子
966	「stco」(大塊偏移表)原子
970	「stsz」(樣本大小表)原子
974	「stss」(同步樣本表)原子
978	「stsd」(樣本描述表)原子
982	「mp4*」原子
986	「esds」(基本流描述符)原子
990	動畫專家群第四階基本流描述符(Es_Descr)資料結構
1000	動畫專家群第四階基本流描述符
1004	位元組標籤(ES_DescrTag)
1008	位元組數指示
1012	16位元ES_ID值
1016	1位元旗標
1020	5位元資料流優先值
1024	「解碼器組態描述符」資料區塊
1028、1088	同步層組態描述符資料結構
1032	「解碼器組態描述符」資料區塊
1036	位元組標籤(DecoderConfigDescrTag)
1040	解碼器組態描述符尺寸值

1044	解碼器組態描述符資料
1048、1052、1056	資料值
1060、1064	
1068	解碼器組態描述符資料
1072、1076	解碼器特定資訊資料結構
1080	一位元組標籤(DecoderSpecificInfoTag)
1084	位元組數指示
1092	位元組標籤(SLConfigDescrTag)
1094	位元組數指示
1098	資料位元組
1100	sdsm二進制大塊/sdsm媒體資料樣本
1110	命令框
1120	BIFS命令
1130、1140	連續位元
1150	空填充位元
1200	BIFS插入命令
1206	兩位元插入碼
1210	兩位元參數類型碼
1216	插入命令資料
1220	BIFS刪除命令
1226	兩位元刪除碼
1230	兩位元參數類型碼
1236	刪除命令資料
1240	BIFS替換命令

1244 兩位元替換碼

1250 兩位元參數類型碼

1260 替換命令資料

1270 BIFS場景替換命令

1280 兩位元場景替換碼

1290 BIFS場景資料結構

1300 Node插入命令

1304 兩位元插入碼

1308 兩位元參數類型碼

1312 nodeID值

1316 兩位元插入位置碼

1320 8位元位置值

1324 SFNode資料結構/SFNode組件

1328 Indexed Value插入命令

1332 兩位元插入碼

1336 兩位元參數類型碼

1340 nodeID值

1344 inFieldID值

1348 兩位元插入位置碼

1352 8位元位置值

1356 欄位值資料結構/組件「fieldvalue」

1360 Route插入命令

1364 兩位元插入碼

1368 兩位元參數類型碼

1372 「isUpdateable」位元

1376 routeID

1380 departureNodeID值

1384 departureFieldID值

1388 arrivalNodeID值

1392 arrivalFieldID值

1400 Node刪除命令

1406 兩位元刪除碼

1412 兩位元參數類型碼

1418 nodeID值

1424 Indexed Value刪除命令

1430 兩位元刪除碼

1436 兩位元參數類型碼

1442 nodeID值

1448 inFieldID值

1454 兩位元刪除位置值

1460 位置組件

1466 Route刪除命令

1472 兩位元刪除碼

1478 兩位元參數類型碼

1484 RouteID值

1500 Node替換命令

1504 兩位元替換碼

1508 兩位元參數類型碼

1510 nodeID值

1514 SFNode組件

1520 Field替換命令

1524 兩位元替換碼

1528 兩位元參數類型碼

1530 nodeID值

1534 inFieldID值

1538 組件「field value」

1540 Indexed Value替換命令

1544 兩位元替換碼

1548 兩位元參數類型碼

1550 nodeID值

1554 inFieldID值

1558 兩位元替換位置碼

1560 8位元位置值

1564 「field value」組件

1570 Route替換命令

1574 兩位元替換碼

1578 兩位元參數類型碼

1580 routeID值

1584 departureNodeID值

1588 departureFieldID值

1590 arrivalNodeID值

1594 arrivalFieldID值

1600	BIFS場景資料結構
1610	6位元保留欄位/「reserved」組件
1620、1630	一位元旗標
1640	SFTopNode組件
1650	一位元旗標(hasRoutes)
1660	Routes組件
1700	SFNode資料結構
1704	布林數量「isReused」
1708	「nodeIDRef」組件
1710	SFNode資料結構
1712	布林數量「isReused」
1714、1734	本地節點類型值(localNodeType)
1716	布林數量「isUpdateable」
1718、1738	nodeID值
1720、1740	空終止字串(「名稱」)
1722	布林數量「maskAccess」
1726	遮罩位元
1728	二進制欄位值
1730	SFNode資料結構
1732	布林數量「isReused」
1736	布林數量「isUpdateable」
1740	空終止文數字元串
1742	布林數量「maskAccess」
1744、1750	一位元結束旗標

1746	fieldRef值
1748	二進制欄位值
1760	列表形式
1762、1782	一位元保留位元
1766	布林數量「isList」
1770、1772	一位元endFlag值
1774	SFField值
1780	向量形式
1782	第一位元
1786	布林數量「isList」
1790	5位元欄位(nBits)
1792	後續欄位計數值(nFields)
1796	nFields SFField結構
1800	Routes組件/列表形式
1805、1835	一位元列表旗標
1810	Routes結構
1815、1820	一位元moreRoutes旗標
1830	Routes組件/向量形式
1840	五位元nBits欄位
1845	後續numRoutes值
1850	Routes結構
1860	Routes結構
1865	一位元旗標(isUpdateable)
1870	routeID值

1875	空終止字串(routeName)
1880	outNodeID值
1885	outFieldRef值
1890	inNodeID值
1895	inFieldRef值
1900	odsm大塊
1920、1940	odsm樣本
1960	odsm命令
2000	ObjectDescriptorUpdate命令
2010	一位元組ObjectDescriptorUpdateTag
2020、2060	位元組數(numBytes)指示
2040	ObjectDescriptorRemove命令
2050	一位元組ObjectDescriptorRemoveTag
2060	位元組數(numBytes)指示
2070	ObjectDescriptorsId值
2080	填充位元
2108	一位元組MP4_OD_Tag
2116	numBytes值
2124	十位元ObjectDescriptorID值
2132	一位元URL_Flag值
2140	五位元保留欄位(0x1f)
2148	資料結構
2160	EsIdRef資料結構
2170	一位元組ES_ID_RefTag

2180	numBytes值
2190	16位元基本資料流ID(ES_ID)值
2200	處理過程
2210	動畫專家群第四階中間檔案/mp4二進制檔案
2220	XMT-A文件
2230	一組媒體資料檔案
2240	mp4檔案至中間文件轉換器
2245	一組中間文件
2250	mp4-file文件
2260	mp4-bifs文件
2270	中間文件至XMT-A轉換器
2300	mp4file文件
2310	媒體資料(mdat)元件
2320	moov元件
2330	mp4fiods元件
2340	使用者資料(udta)元件
2350	trak元件
2360	「mp4fiods」元件
2370	「ObjectDescrID」屬性
2380	EsIdInc元件
2390	「trackID」屬性
2400	mdat元件
2410	sdsm元件

2420	odsm元件
2430	mediaFile元件
2440	sdsm元件
2450、2490	大塊元件
2460	odsm元件
2470	odsmChunk元件
2480	mediaFile元件
2500	odsmChunk元件
2510	odsmSample元件
2515	time屬性
2520	odsmSample元件
2525	time屬性
2530	odsm命令元件
2540	ObjectDescrUpdate元件
2550	ObjectDescriptor元件
2555、2575	「ODID」屬性
2560	EsIdRef元件
2565	「EsId」屬性
2570	ObjectDescrRemove元件
2600	「trak」元件
2604	mdia元件
2608	hdlr元件
2612	minf元件
2616	dinf元件

2620	dref元件
2624	urlData
2628、2652	stbl元件
2632	媒體標頭元件
2636	tref(磁軌參考)元件
2640	「mpod」元件
2644	edts(編碼列表)元件
2648	elst元件
2656	stsc(樣本至大塊表)元件
2660	stts(時間至樣本表)元件
2664	stco(大塊偏移表)元件
2668	stsz(樣本大小表)元件
2672	stss(同步樣本表)元件
2676	stsd(樣本描述表)元件
2680	mp4*元件
2684	esds元件
2688	ES_Descr元件
2700	ES_Descr元件
2710	DecoderConfigDescriptor元件
2720	BIFS_DecoderConfig元件
2730	JPEG_DecoderConfig元件
2740	VisualConfig元件
2750	AudioConfig元件
2760	SLConfigDescriptor元件

2800	mp4-bifs文件
2810	「bifsConfig」元件
2820	commandFrame元件
2830	commandFrame元件
2840	bifsCommand元件
2910	bifsCommand元件
2920	mp4bifs Node元件
2930	ReplaceScene bifsCommand元件
2940	「TopNode」元件
2950、2960	附屬「Routes」元件
3000	mp4bifs初始節點元件
3010	NodeId屬性
3016	name屬性
3020	性質欄位屬性
3030	附屬節點元件
3040	mp4bifs Conditional Node元件
3050	NodeId屬性
3056	名稱屬性
3070	附屬bifsCommand元件
3080	ReusedNode元件
3090	nodeRef屬性
3100	QtInfo資料結構
3110	成員「numMdatAtoms」
3113	成員「mdat陣列」

3116 MoovHeader成員

3118 成員「numTracks」

3120 成員「trak陣列」

3123 成員UserData

3130 Mp4fInitObjectDescr資料結構

3138 成員「ObjectDescrID」

3140 成員「urlFlag」

3146 成員「inlineProfiles」

3156 EsIdInc陣列

3160 mdat資料結構

3163 成員「size」

3166 成員「start」

3168 成員「MediaData」

3170 MoovHeader資料結構

3200 trak結構

3202 成員「size」

3204 成員「Trackheader」

3206 成員「TrackMedia」

3208 TrackEdit成員

3210 TrackReference成員

3212 Trackheader結構

3214 成員「size」

3216 成員「version」

3218 成員「creationTime」

3220 成員「modifiedTime」

3222 成員「duration」

3224 成員「trackID」

3226 TrackMedia資料結構

3228 成員「size」

3230 成員「MediaHeader」

3232 成員「Handler」

3234 成員MediaInfo

3236 TrackEdit資料結構

3242 EditList成員

3244 EditList結構

3246 成員size

3248 成員version & Flags

3250 成員numSegments

3252 成員EditSegmentarray

3256 Editsegment結構

3258 成員「startTime」

3260 成員「duration」

3262 成員「relativeRate」

3264 TrackReference資料結構

3266 成員size

3268 成員Mpod

3270 Mpod結構

3272 成員size

3274	成員numEntries
3276	成員trackIDarray
3280	UserData資料結構
3282	成員size
3284	成員numCprt
3286	成員cprt array
3290	cprt結構
3292	成員size
3294	成員version & Flags
3296	成員language
3298	成員notice
3300	MediaHeader結構
3320	Handler結構
3323	成員「size」
3326	成員「version & Flags」
3328	成員「handlerType」
3330	成員「name」
3333	MediaInfo結構
3336	成員「size」
3338	成員「MediaDataInfo」
3340、3400	MediaInfoHeader結構
3343	SampleTables結構
3350	MediaDataInfo結構
3353	成員「size」

3356 成員「version & Flags」

3358 成員DataRefTable

3360 DataRefTable資料結構

3363 成員「size」

3366 成員「version & Flags」

3368 數資料參考

3370 成員「dataRef array」

3380 dataRef資料結構

3383 成員「size」

3386 成員「version & Flags」

3388 成員「dataRefType」

3390 成員「dataRefData」

3400 SampleTables結構

3403 成員「size」

3406 SampleToChunkTable結構

3408 ChunkOffsetTable結構

3410 TimeToSampleTable結構

3413 SampleSizeTable結構

3416 SyncSampleTable結構

3418 SampleDescriptionTable結構

3420 樣本表資料結構

3423 成員「size」

3426 成員「version & Flags」

3428 成員「numEntries」

3430 成員「table array」

3440 SampleSizeTable資料結構

3443 成員「size」

3446 成員「version & Flags」

3450 成員「sampleSize」

3453 成員「numEntries」

3456 成員「table array」

3460 SampleToChunk資料結構

3463 成員「firstChunk」

3466 成員「numSamples」

3468 描述

3470 TimeToSample資料結構

3473 成員「count」

3476 成員「duration」

3480 SampleDescription資料結構

3483 成員「size」

3486 成員「dataFormat」

3488 成員「dataReference」

3490 成員「esd版本」

3493 成員「esdSize」

3496 成員「esdData」

3500 操作

3510 操作

3520 操作

3525	操作
3530	操作
3535	操作
3540	操作
3545	操作
3560	操作
3570	操作
3580	操作
3600	操作
3610	操作
3620	操作
3630	操作
3640	操作
3650	操作
3660	操作
3700	操作
3710	操作
3720	操作
3725	操作
3730	操作
3735	操作
3740	操作
3745	操作
3760	操作

3770	操作
3780	操作
3800	操作
3810	操作
3820	操作
3825	操作
3830	操作
3835	操作
3840	操作
3845	操作
3850	操作
3855	操作
3860	操作
3870	操作
3880	操作
3900	操作
3910	操作
3920	操作
3925	操作
3930	操作
3935	操作
3940	操作
3945	操作
3960	操作

3970	操作
3980	操作
4000	操作
4010	操作
4020	操作
4025	操作
4030	操作
4035	操作
4040	操作
4045	操作
4050	操作
4055	操作
4060	操作
4065	操作
4070	操作
4080	操作
4090	操作
4100	操作
4110	操作
4120	操作
4125	操作
4130	操作
4135	操作
4140	操作

4145	操作
4150	操作
4155	操作
4160	操作
4165	操作
4170	操作
4175	操作
4180	操作
4190	操作
4195	操作
4200	操作
4210	操作
4220	操作
4230	操作
4240	操作
4250	操作
4260	操作
4270	操作
4280	操作
4300	操作
4310	操作
4315	操作
4320	操作
4330	操作

4340 操作

4350 操作

4355 操作

4360 操作

4365 操作

4370 操作

4375 程序「VisualDataToXml」

4380 操作

4385 操作

4390 操作

4400 操作

4410 操作

4420 操作

4430 操作

4440 操作

4450 操作

4460 操作

4470 操作

4480 操作

4500 操作

4510 操作

4520 操作

4525 程序

4530 操作

4535 操作

4540 操作

4550 操作

4560 操作

4565 操作

4570 程序「得到標籤大小」

4575 數量「esdPos」

4580 操作

4585 操作

4600 操作

4610 操作

4620 操作

4630 操作

4640 操作

4650 操作

4660 操作

4670 操作

4680 操作

4690 操作

4695 操作

4700 操作

4705 操作

4710 操作

4715 操作

4720	操作
4725	操作
4730	操作
4735	操作
4740	操作
4745	操作
4750	操作
4755	操作
4760	操作
4770	操作
4780	操作
4800	操作
4805	操作
4810	操作
4820	操作
4825	操作
4830	操作
4835	操作
4840	操作
4845	操作
4850	操作
4860	操作
4865	操作
4870	操作

4875	操作
4880	操作
4885	操作
4890	操作
4895	操作
4900	操作
4910	操作
4920	操作
4925	操作
4930	操作
4940	操作
4950	操作
4960	操作
4965	操作
4970	操作
4980	操作
4990	操作
5000	操作
5010	操作
5020	操作
5030	操作
5040	操作
5050	操作
5060	操作

5070	操作
5080	操作
5100	操作
5105	操作
5110	操作
5115	操作
5120	操作
5125	操作
5130	操作
5140	操作
5145	操作
5150	操作
5160	操作
5165	操作
5170	操作
5180	操作
5185	操作
5190	操作
5200	操作
5205	操作
5210	操作
5215	操作
5220	操作
5225	操作

5230	操作
5240	操作
5245	操作
5250	操作
5260	操作
5265	操作
5270	操作
5275	操作
5280	操作
5285	操作
5290	操作
5300	操作
5305	操作
5310	操作
5315	操作
5320	操作
5325	操作
5330	操作
5335	操作
5340	操作
5345	操作
5350	操作
5355	操作
5360	操作

5365	操作
5370	操作
5375	操作
5380	操作
5400	操作
5410	操作
5420	操作
5430	操作
5440	操作
5450	操作
5460	操作
5470	操作
5480	操作
5490	操作
5500	操作
5510	操作
5520	操作
5530	操作
5540	操作
5550	操作
5560	操作
5570	操作
5580	操作
5590	操作

5600	操作
5610	操作
5620	操作
5630	操作
5640	操作
5650	操作
5660	操作
5670	操作
5680	操作
5690	操作
5700	操作
5710	操作
5720	操作
5730	操作
5740	操作
5750	操作
5760	操作
5770	操作
5780	操作
5790	操作
5800	操作
5805	操作
5810	操作
5820	操作

5825	操作
5830	操作
5835	操作
5840	操作
5850	操作
5860	操作
5870	操作
5880	操作
5890	操作
5900	操作
5920	操作
5940	操作
5960	操作
5980	操作
6000	操作
6010	操作
6020	操作
6030	操作
6040	操作
6050	操作
6100	操作
6110	操作
6120	操作
6130	操作

6140	操作
6150	操作
6160	操作
6170	操作
6180	操作
6190	操作
6200	第一行
6220	第二行
6240	第一行
6260	第二行
6300	操作
6310	操作
6320	操作
6330	操作
6340	操作
6350	操作
6360	操作
6370	操作
6380	操作
6390	操作
6400	操作
6410	操作
6420	操作
6430	操作

6440	操作
6450	操作
6500	操作
6510	操作
6520	操作
6530	操作
6540	操作
6550	操作
6560	操作
6570	操作
6580	操作
6590	操作

伍、中文發明摘要：

本發明揭示一種用於將一二進制動畫專家群第四階(mp4)二進制檔案，轉換為一可擴充動畫專家群第四階本文(XMT)檔案的方法、系統及電腦程式產品。本發明包括產生至少一代表該mp4二進制檔案的中間結構文件。一翻譯器係配置成用以輸入該中間文件，並產生一XMT結構文件。然後，用一XMT串聯器根據該XMT結構文件，建立該XMT檔案。

陸、英文發明摘要：

A method, system, and computer program product for converting a binary MPEG-4 (mp4) binary file into an Extensible MPEG-4 Textual (XMT) file. The invention includes generating at least one intermediary structured document representing the mp4 binary file. A translator is configured to input the intermediate document and generate an XMT structured document. An XMT serializer is then used to create the XMT file based on the XMT structured document.

柒、指定代表圖：

(一)本案指定代表圖為：第（ 1 ）圖。

(二)本代表圖之元件代表符號簡單說明：

100	XMT-A檔案/XMT-A文件
110	Header元件
120	Body元件
130	InitialObjectDescriptor元件
140	「par」元件
150	Profiles元件
160	Descr元件
170	「esDescr」元件
180、190	「ES_Descriptor」子元件

捌、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

拾壹、圖式：

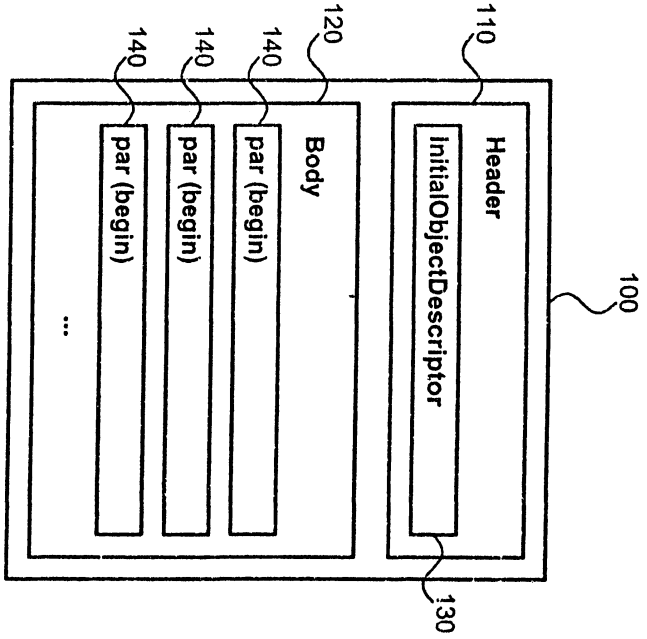

Header
InitialObjectDescriptor
Body
par (begin)
par (begin)
par (begin)
...
100
110
120
130
140
140
140

圖1A

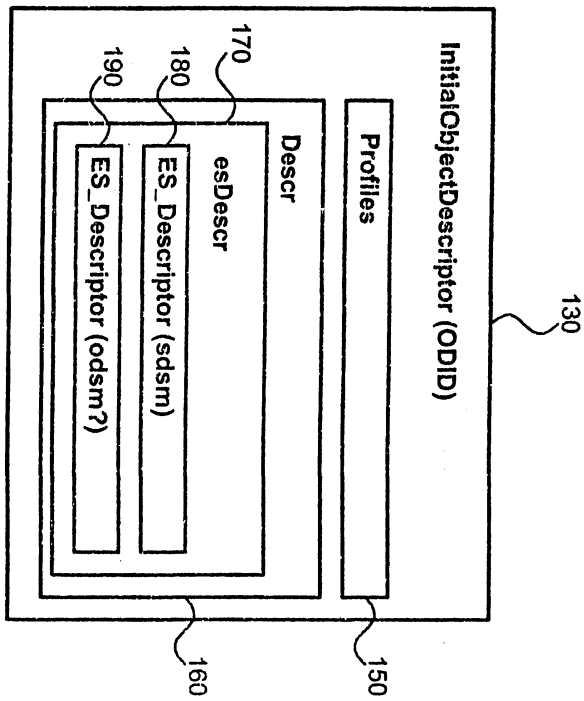

InitialObjectDescriptor (ODID)
Profiles
Descr
esDescr
ES_Descriptor (sdsm)
ES_Descriptor (odsm?)
130
150
160
170
180
190

圖1B

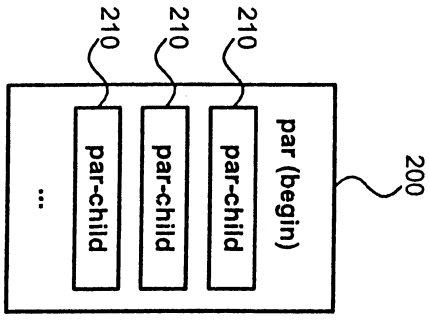

par (begin)
par-child
par-child
par-child
...
200
210
210
210

圖 2A

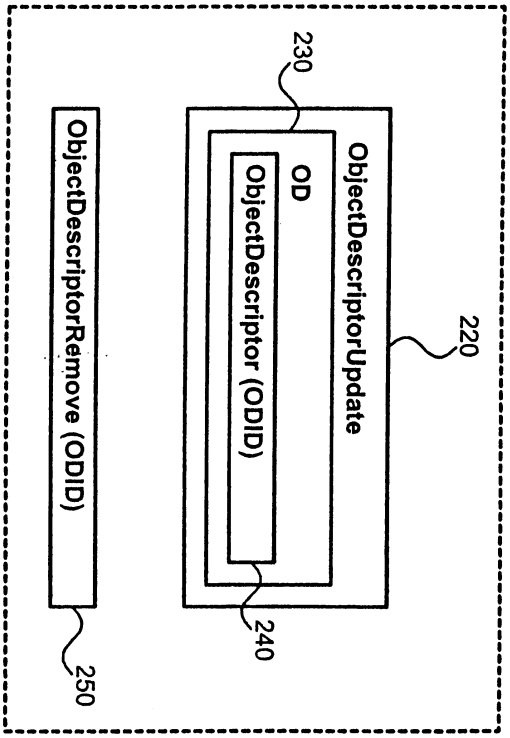

ObjectDescriptorUpdate
OD
ObjectDescriptor (ODID)
ObjectDescriptorRemove (ODID)
220
230
240
250

圖 2B

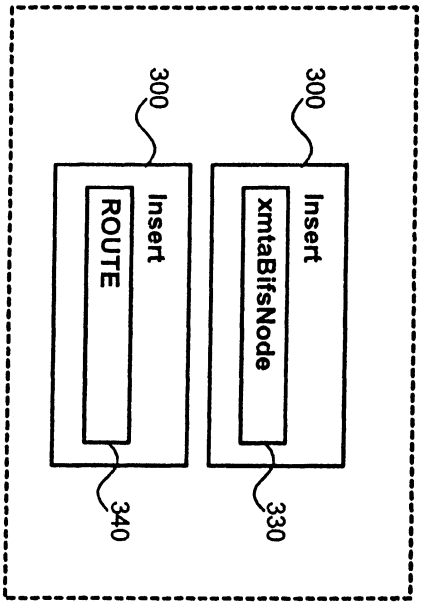

300
Insert
xmtaBifsNode
330
300
Insert
ROUTE
340

圖3A

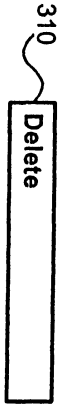

310
Delete

圖3B

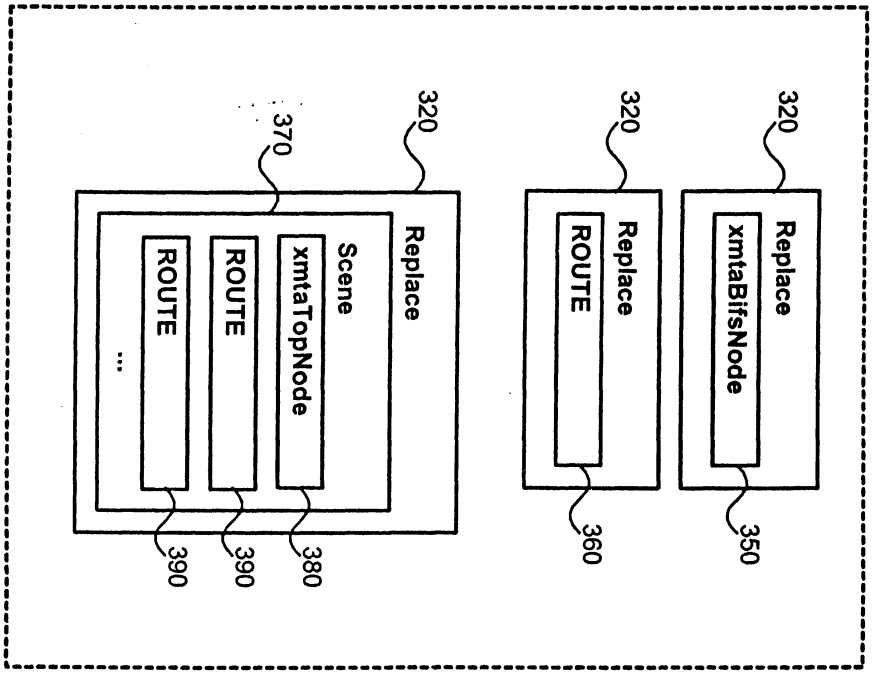

320
Replace
xmtaBifsNode
350
320
Replace
ROUTE
360
320
370
Replace
Scene
xmtaTopNode
380
ROUTE
390
ROUTE
...
390

圖3C

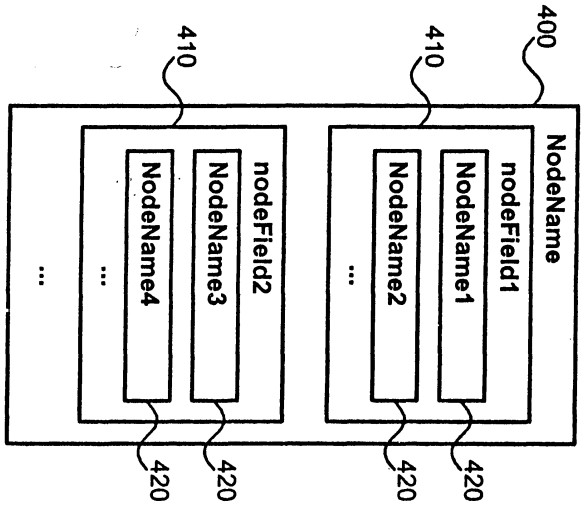

400
410
410
NodeName
nodeField1
NodeName1
NodeName2
...
nodeField2
NodeName3
NodeName4
...
...
420
420
420
420

圖4

```
500 <NodeName
510     DEF = "nodeIDstring"
520     field0 = "value0"
520     field2 = "value2"
520     field3 = "value3"
520     field5 = "value5" . . . >

530     <field1>
550         <NodeName1 . . . / >
550         <NodeName2 . . . / >
                . . .
540     </field1>

530     <field4>
550         <NodeName3 . . . / >
                . . .
540     </field4>

560 </NodeName>
```

圖5A

```
570
<NodeName                    590
    USE = "nodeIDstring" / >
580
```

圖5B

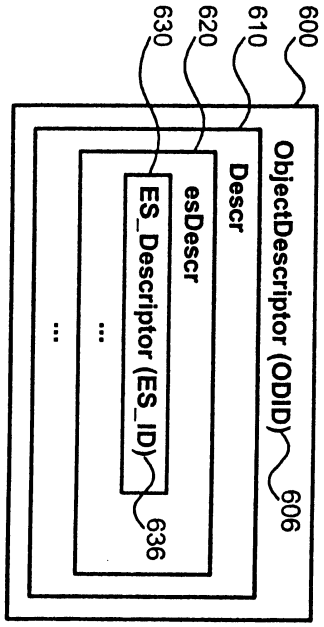

600
610
620
630
ObjectDescriptor (ODID)
606
Descr
esDescr
ES_Descriptor (ES_ID)
636
...
...

圖 6A

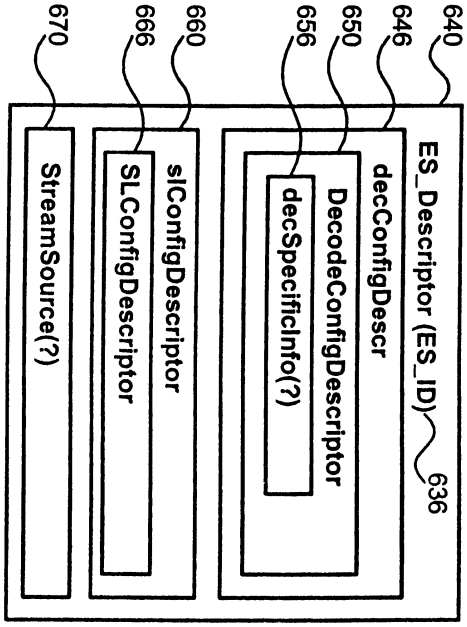

640
646
650
656
660
666
670
ES_Descriptor (ES_ID)
636
decConfigDescr
DecodeConfigDescriptor
decSpecificInfo(?)
slConfigDescriptor
SLConfigDescriptor
StreamSource(?)

圖 6B

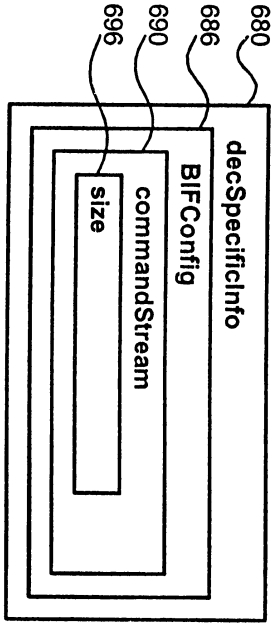

680
686
690
696
decSpecificInfo
BIFConfig
commandStream
size

圖 6C

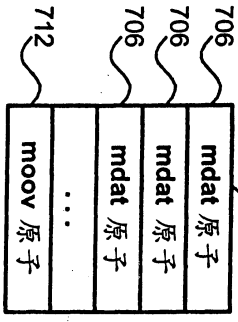


圖7A

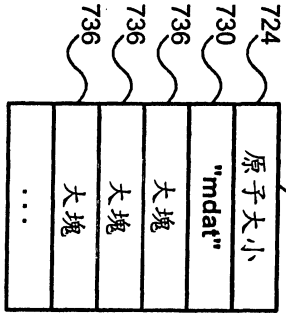


圖7B

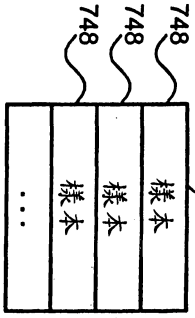


圖7C

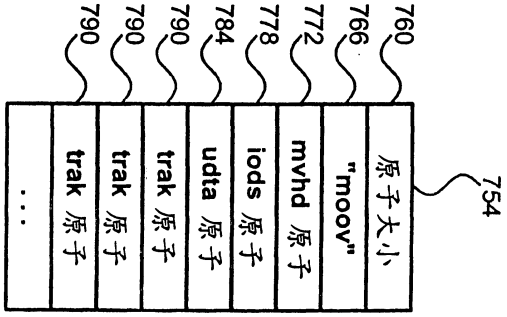


圖7D

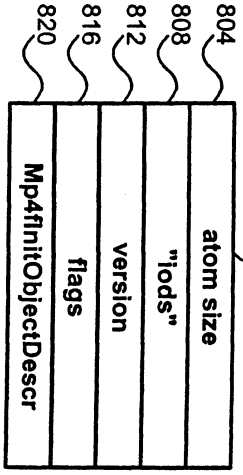

800
804
808
812
816
820
atom size
"iods"
version
flags
Mp4fInitObjectDescr

圖 8A

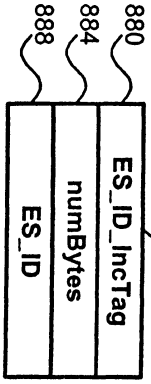

876
880
884
888
ES_ID_IncTag
numBytes
ES_ID

圖 8C

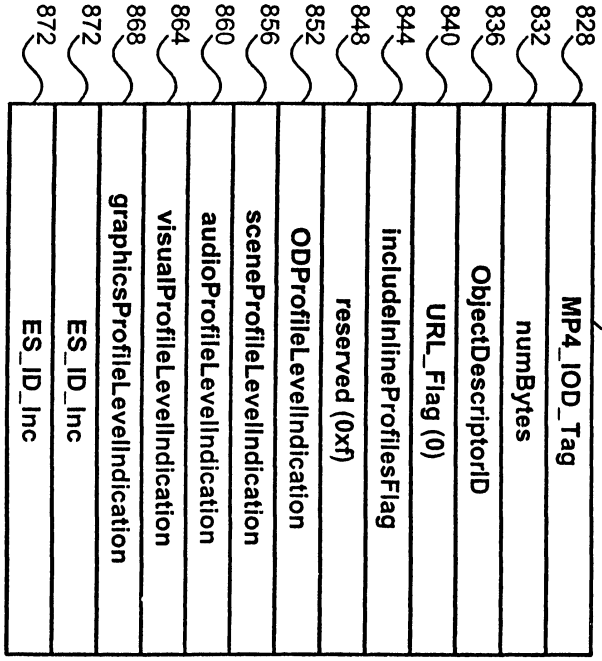

824
828
832
836
840
844
848
852
856
860
864
868
872
872
MP4_IOD_Tag
numBytes
ObjectDescriptorID
URL_Flag (0)
includeInlineProfilesFlag
reserved (0xf)
ODProfileLevelIndication
sceneProfileLevelIndication
audioProfileLevelIndication
visualProfileLevelIndication
graphicsProfileLevelIndication
ES_ID_Inc
ES_ID_Inc

圖 8B

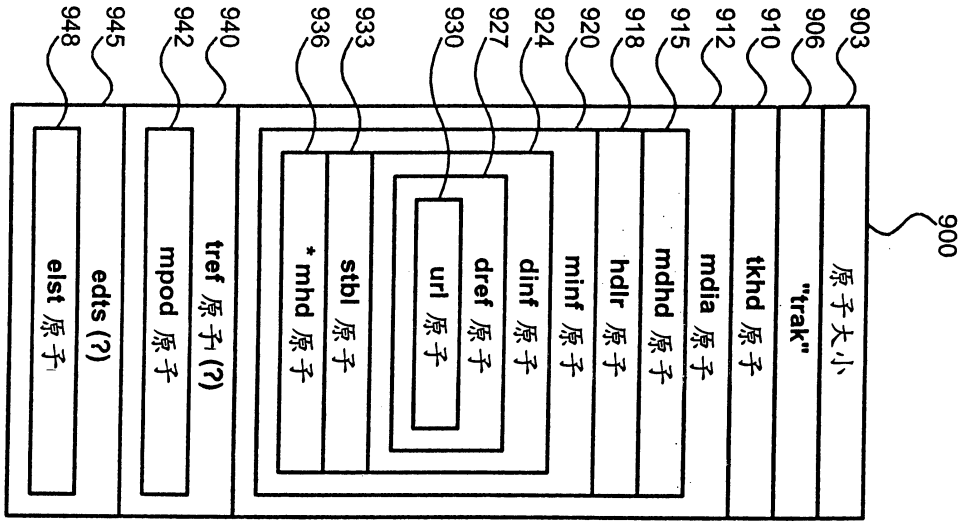

900
903 原子大小
906 "trak"
910 tkhd 原子
912 mdia 原子
915 mdhd 原子
918 hdlr 原子
920 minf 原子
924 dinf 原子
927 dref 原子
930 url 原子
933 stbl 原子
936 * mhd 原子
940 tref 原子 (?)
942 mpod 原子
945 edts (?)
948 elst 原子

圖9A

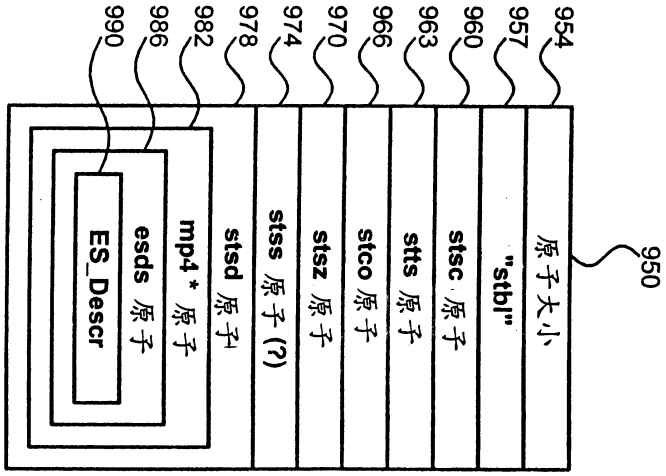

950
954 原子大小
957 "stbl"
960 stsc 原子
963 stts 原子
966 stco 原子
970 stsz 原子
974 stss 原子 (?)
978 stsd 原子
982 mp4 * 原子
986 esds 原子
990 ES_Descr

圖9B

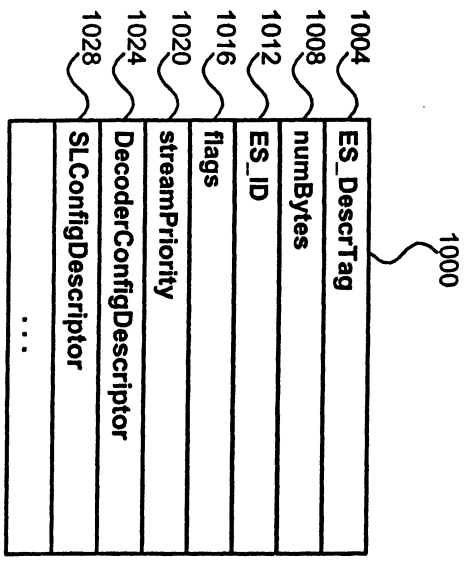


圖10A

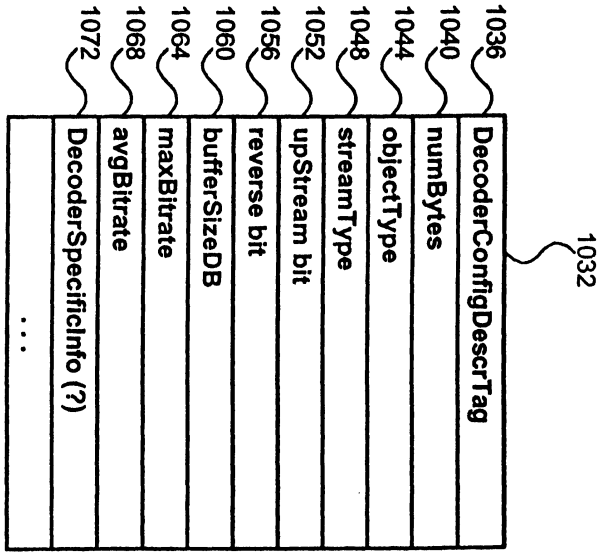


圖10B

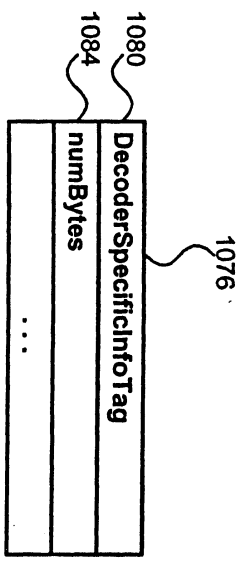


圖10C

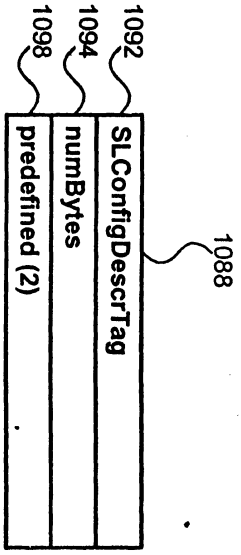


圖10D

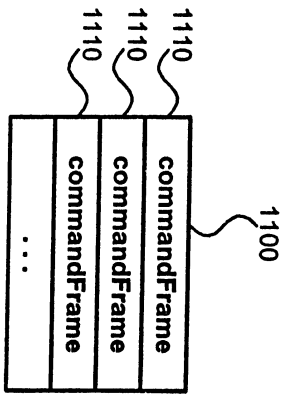

1110
1110
1110
commandFrame
commandFrame
commandFrame
...
1100

圖11A

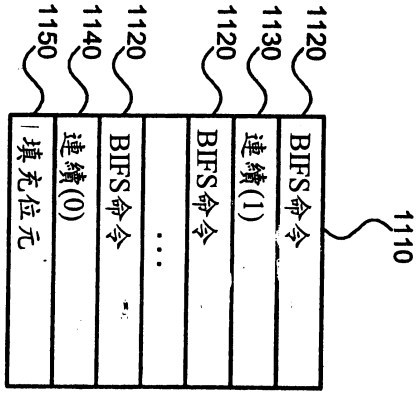

1120
1130
1120
1120
1140
1150
BIFS命令
連續(1)
BIFS命令
...
BIFS命令
連續(0)
填充位元
1110

圖11B

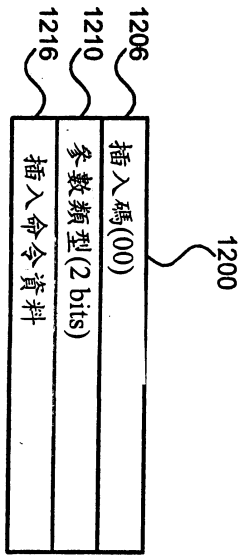


圖 12A

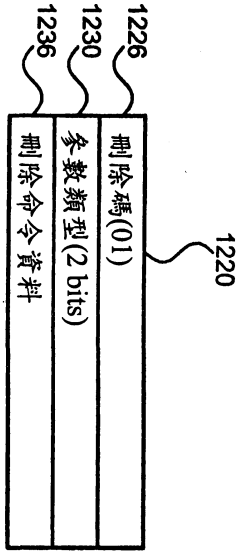


圖 12B

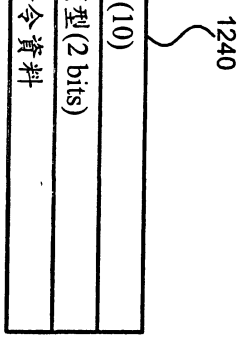


圖 12C

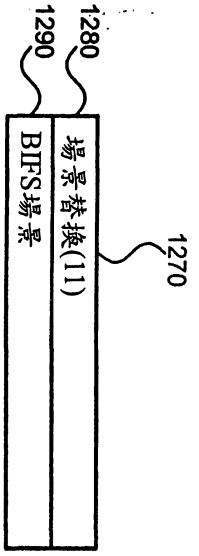


圖 12D

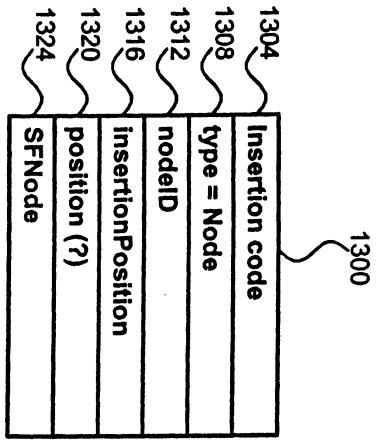


圖13A

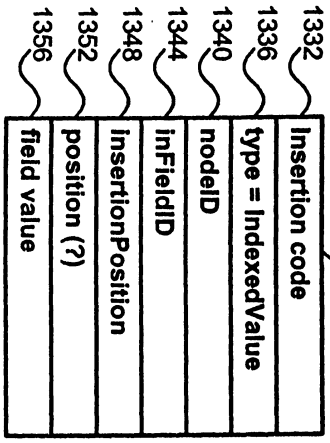


圖13B

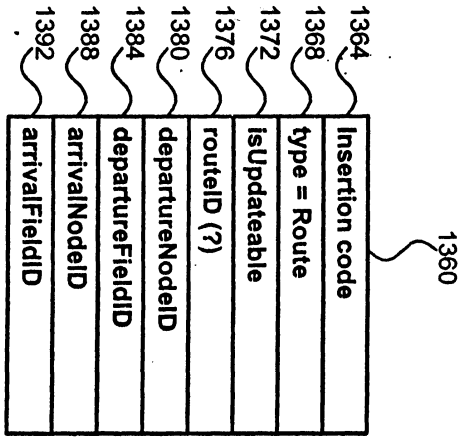


圖13C

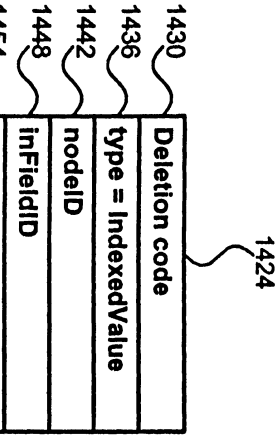


圖14B

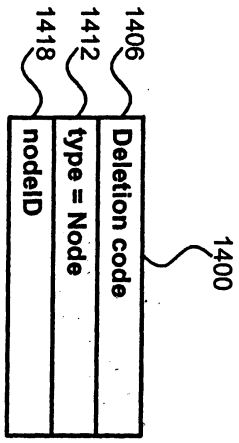


圖14A

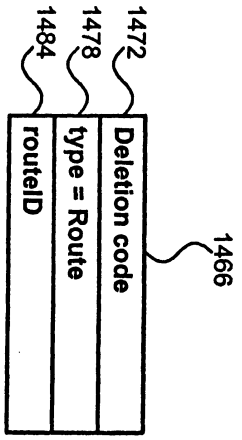


圖14C

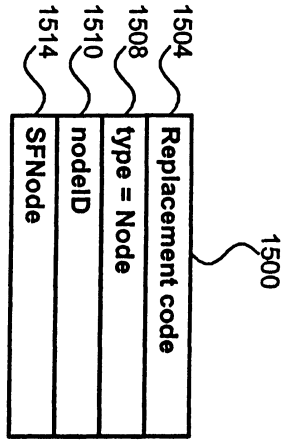


圖15A

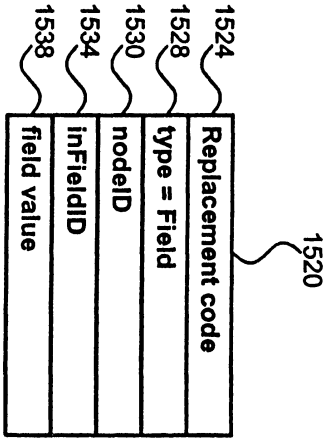


圖15B

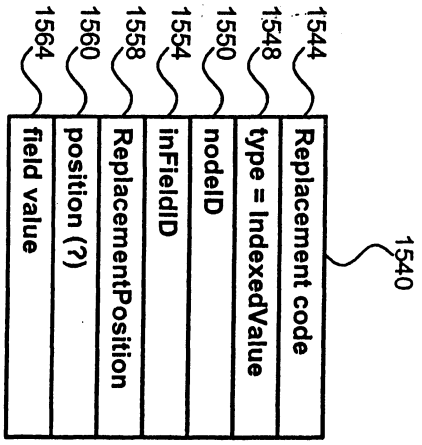


圖15C

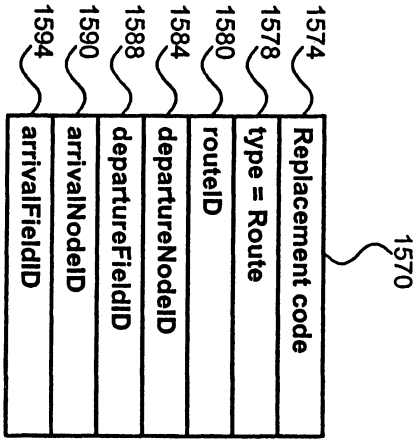


圖15D

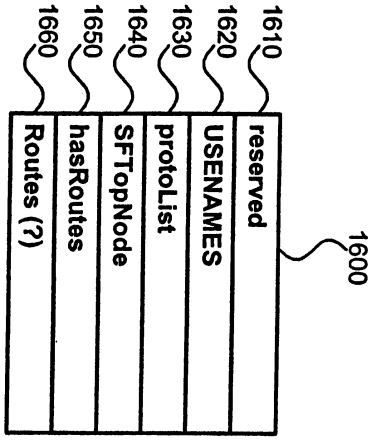

1610
1620
1630
1640
1650
1660
reserved
USENAMES
protoList
SFTopNode
hasRoutes
Routes (?)
1600

圖 16

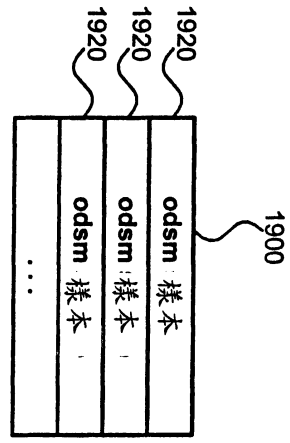

1900
1920
1920
1920
odsm 樣本
odsm 樣本
odsm 樣本
...

圖19A

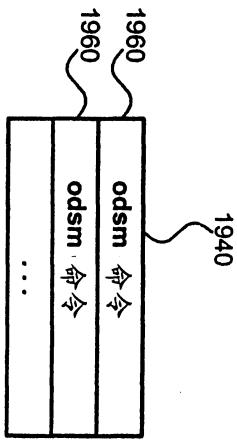

1940
1960
1960
odsm 命令
odsm 命令
...

圖19B

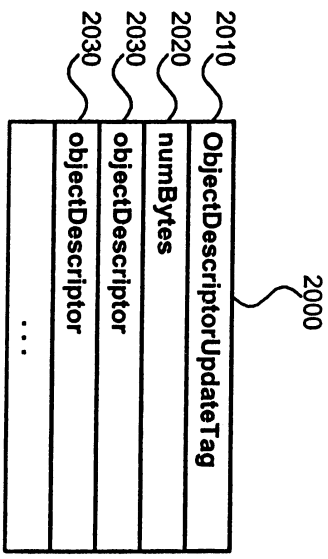

2000
2010
ObjectDescriptorUpdateTag
2020
numBytes
2030
objectDescriptor
2030
objectDescriptor
...

圖20A

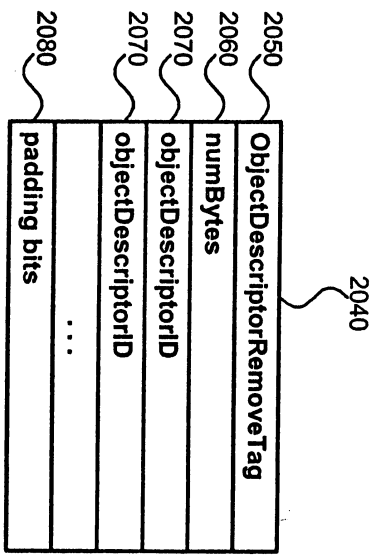

2040
2050
ObjectDescriptorRemoveTag
2060
numBytes
2070
objectDescriptorID
2070
objectDescriptorID
...
2080
padding bits

圖20B

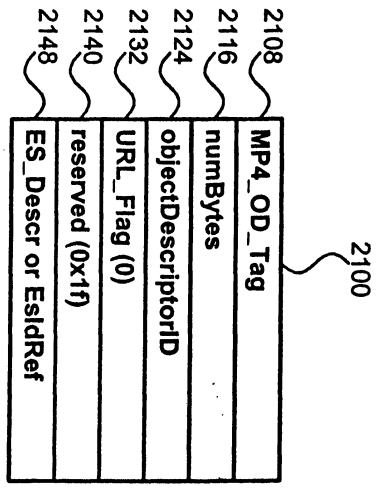

2100
2108
MP4_OD_Tag
2116
numBytes
2124
objectDescriptorID
2132
URL_Flag (0)
2140
reserved (0x1f)
2148
ES_Descr or EsIdRef

圖21A

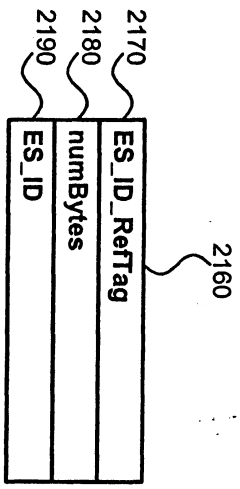

2160
2170
ES_ID_RefTag
2180
numBytes
2190
ES_ID

圖21B

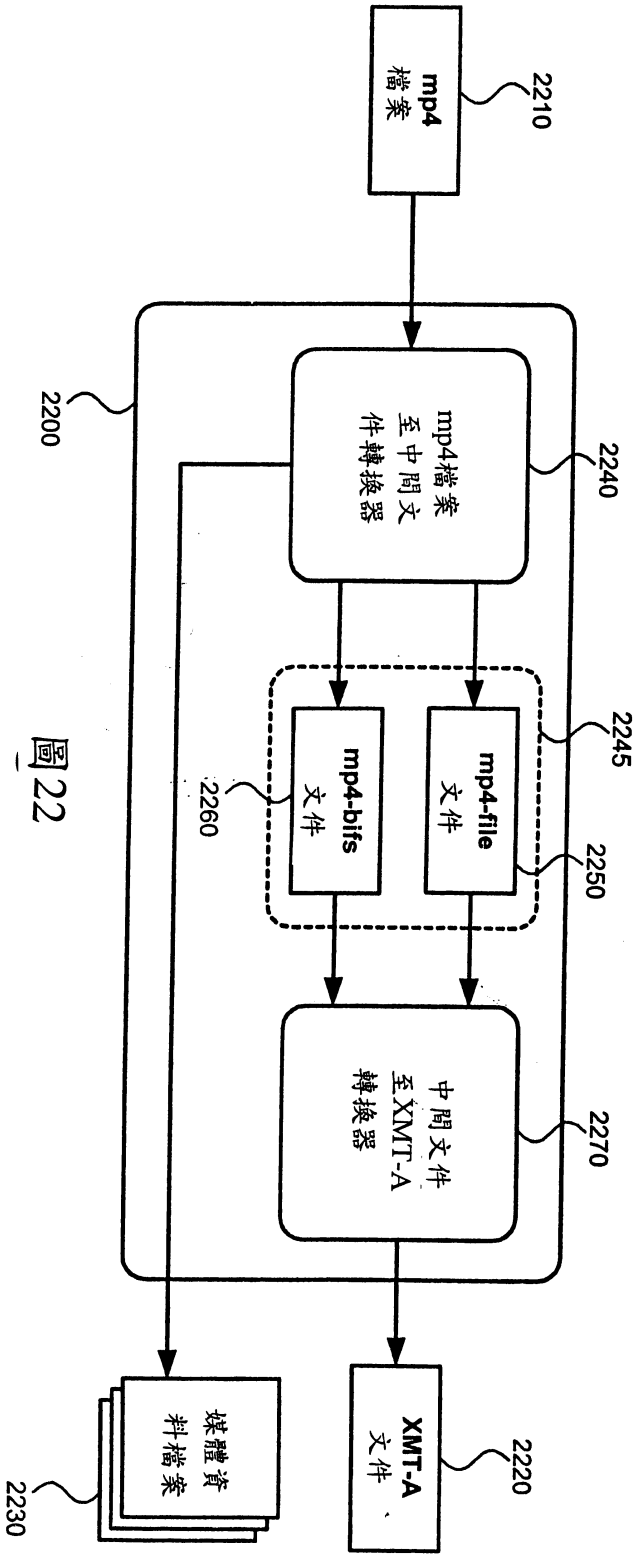

mp4
檔案
2210
2200
mp4檔案
至中間文
件轉換器
2240
mp4-bifs
文件
2260
mp4-file
文件
2245
2250
中間文件
至XMT-A
轉換器
2270
媒體資
料檔案
2230
XMT-A
文件
2220

圖22

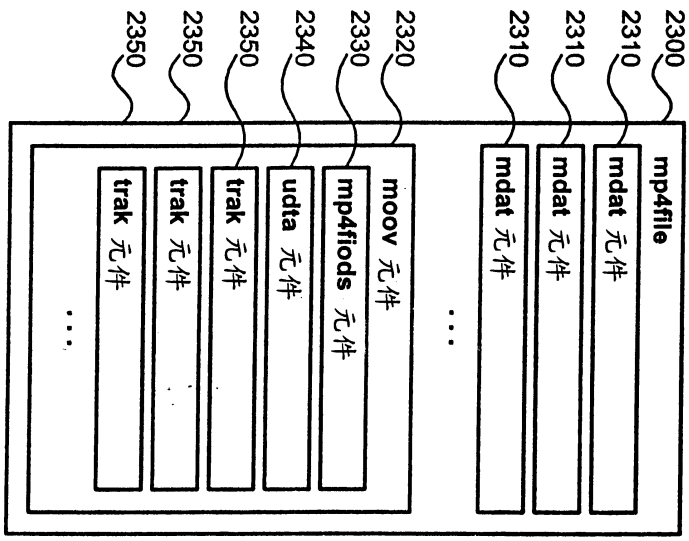

2300
mp4file
2310
mdat 元件
2310
mdat 元件
2310
mdat 元件
...
2320
moov 元件
2330
mp4fiods 元件
2340
udta 元件
2350
trak 元件
2350
trak 元件
2350
trak 元件
...

圖23A

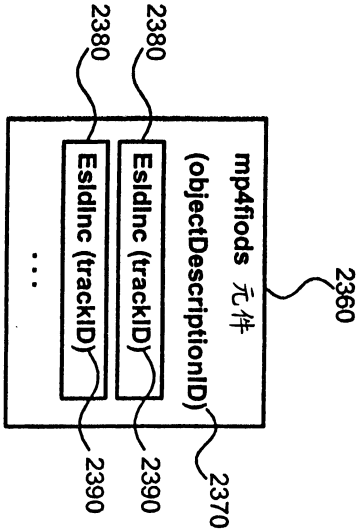

2360
mp4fiods 元件
(objectDescriptionID)
2370
2380
EsIdInc (trackID)
2390
2380
EsIdInc (trackID)
2390
...

圖23B

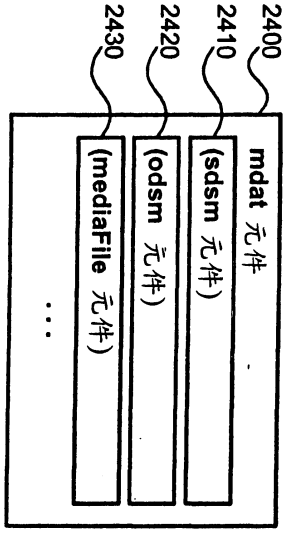


圖24A

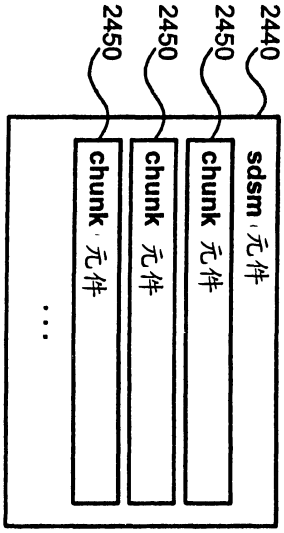


圖24B

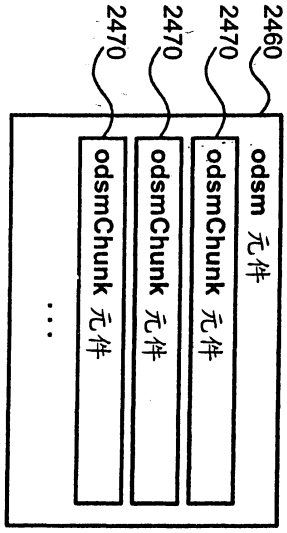


圖24C

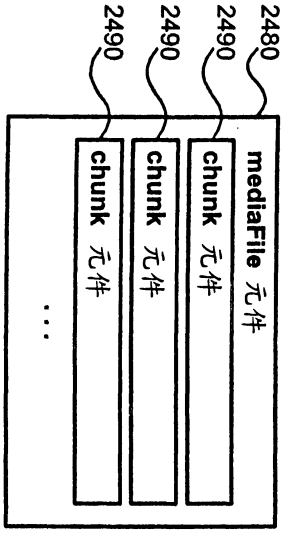


圖24D

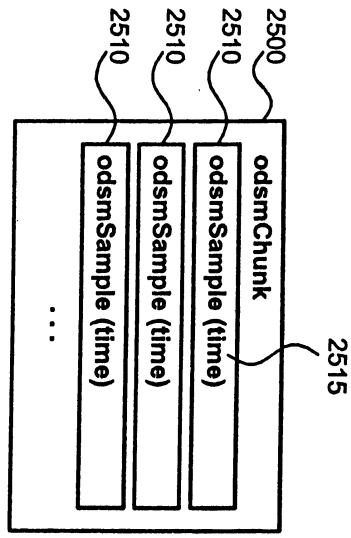


圖25A

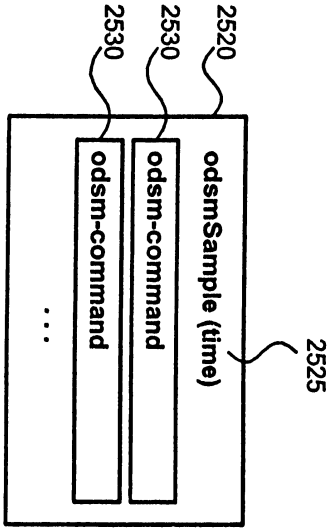


圖25B

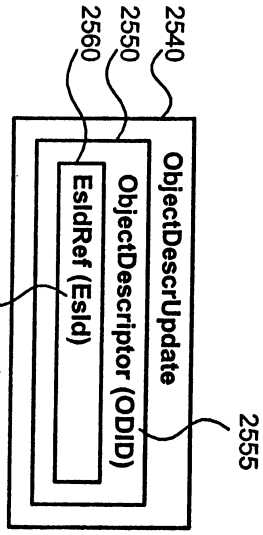


圖25C



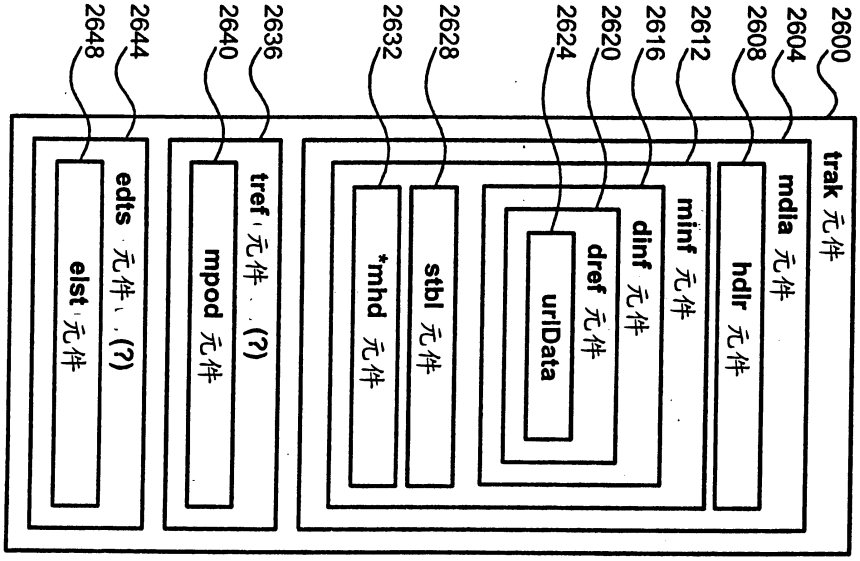

2600
trak 元件
2604
mdia 元件
2608
hdlr 元件
2612
minf 元件
2616
dinf 元件
2620
dref 元件
2624
urlData
2628
stbl 元件
2632
*mhd 元件
2636
tref 元件 (?)
2640
mpod 元件
2644
edts 元件 (?)
2648
elst 元件

圖26A

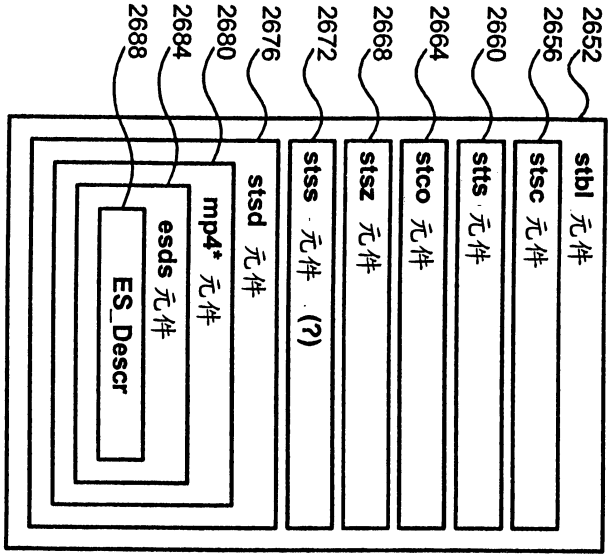

2652
stbl 元件
2656
stsc 元件
2660
stts 元件
2664
stco 元件
2668
stsz 元件
2672
stss 元件 (?)
2676
stsd 元件
2680
mp4* 元件
2684
esds 元件
2688
ES_Descr

圖26B

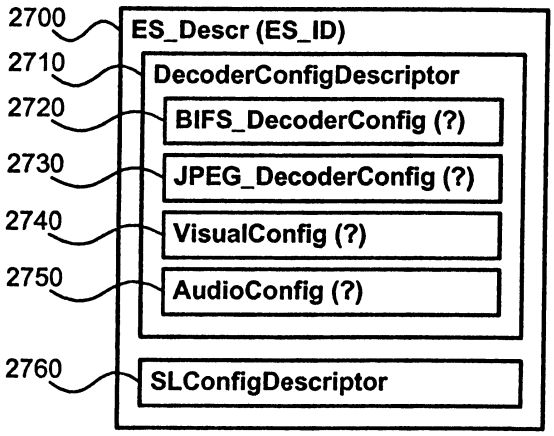

2700
ES_Descr (ES_ID)
2710
DecoderConfigDescriptor
2720
BIFS_DecoderConfig (?)
2730
JPEG_DecoderConfig (?)
2740
VisualConfig (?)
2750
AudioConfig (?)
2760
SLConfigDescriptor

圖27

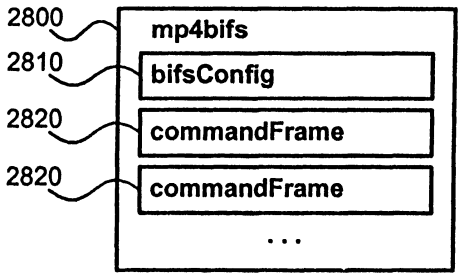

2800
mp4bifs
2810
bifsConfig
2820
commandFrame
2820
commandFrame
...

圖28A

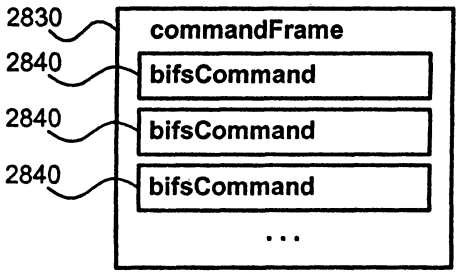

2830
commandFrame
2840
bifsCommand
2840
bifsCommand
2840
bifsCommand
...

圖28B

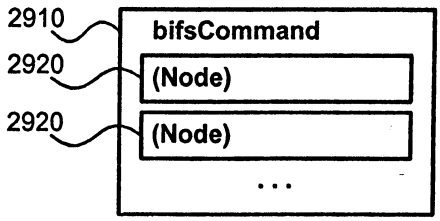

2910
bifsCommand
2920
(Node)
2920
(Node)
...

圖29A

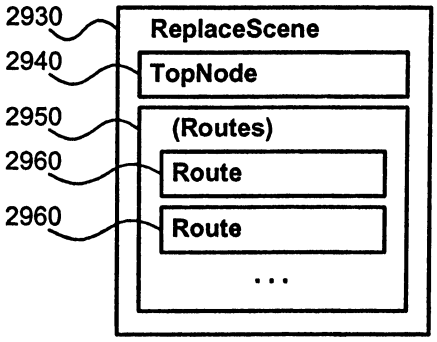

2930
ReplaceScene
2940
TopNode
2950
(Routes)
2960
Route
2960
Route
...

圖29B

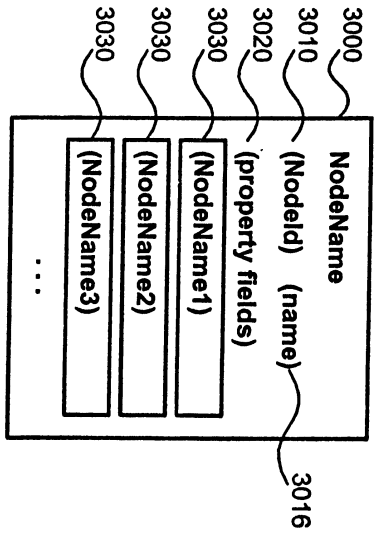


圖 30A

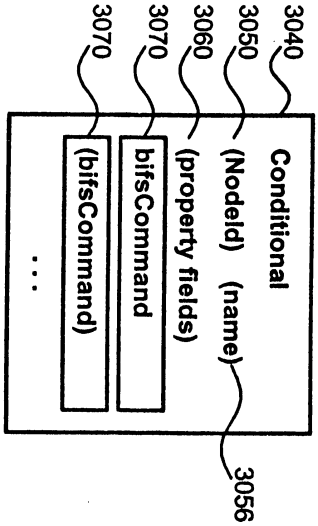


圖 30B



圖 30C

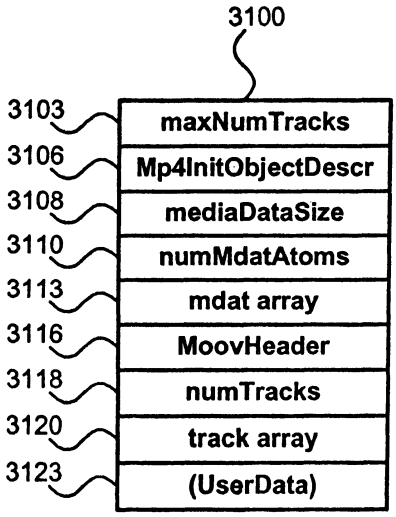


圖31A

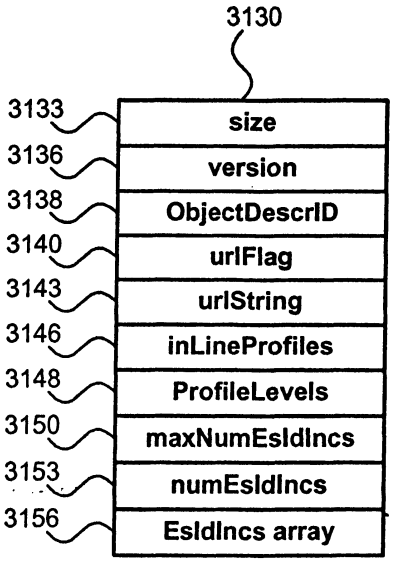


圖31B

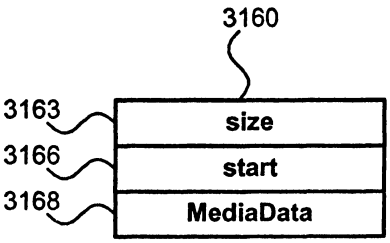


圖31C

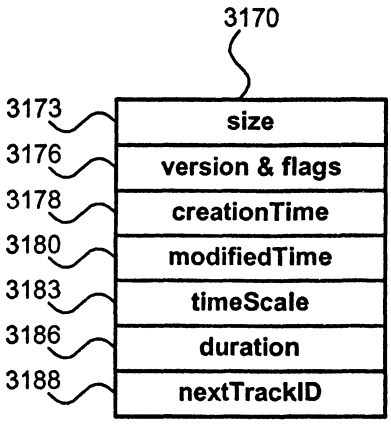


圖31D

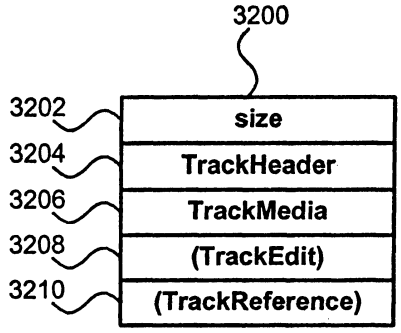


圖32A

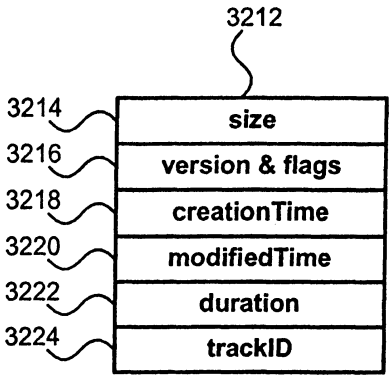


圖32B

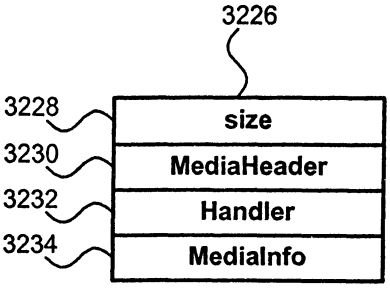


圖32C

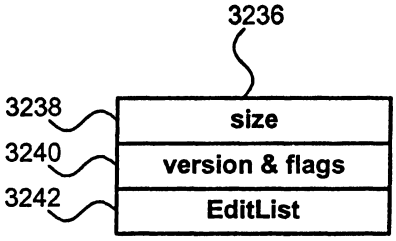


圖32D

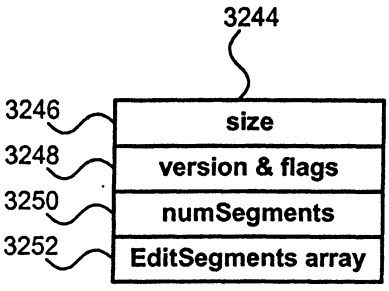


圖32E

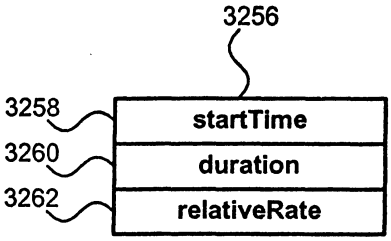


圖32F

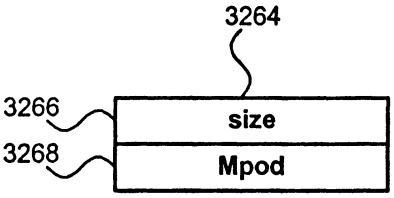


圖32G

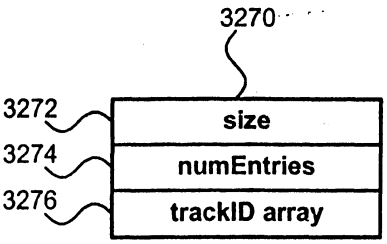


圖32H

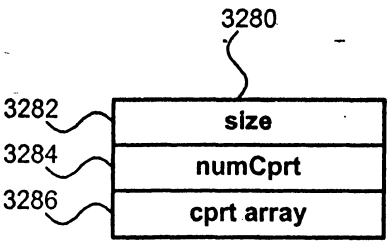


圖32I

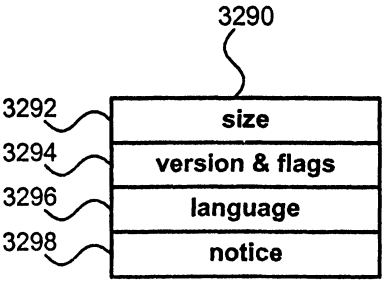


圖32J

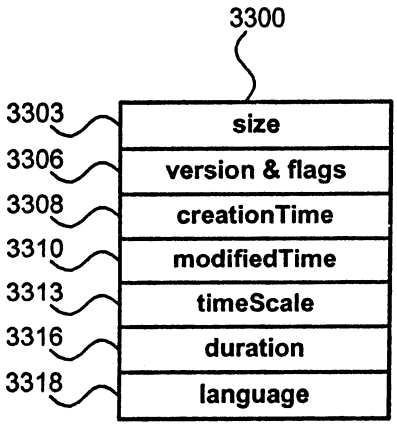


圖33A

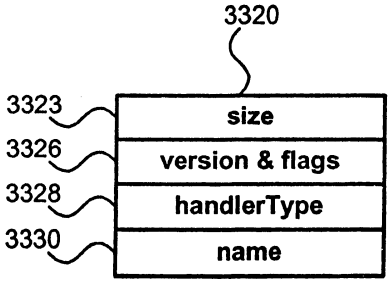


圖33B

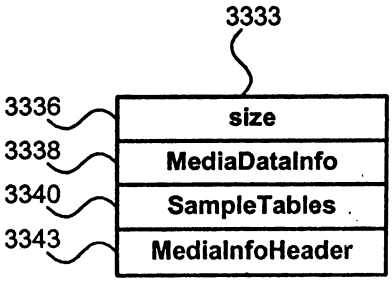


圖33C

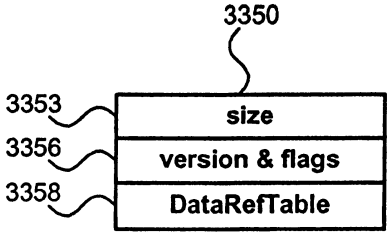


圖33D

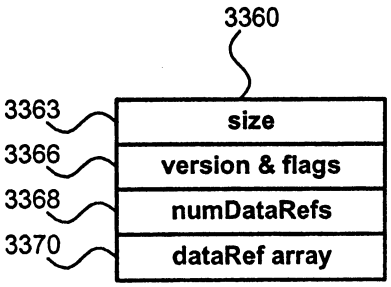


圖33E

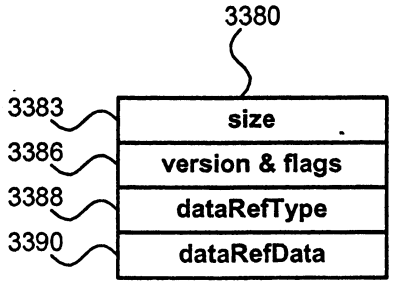


圖33F

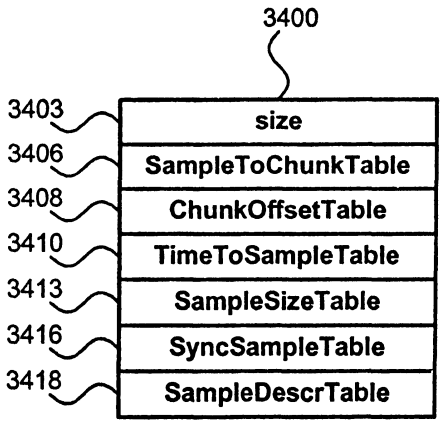


圖34A

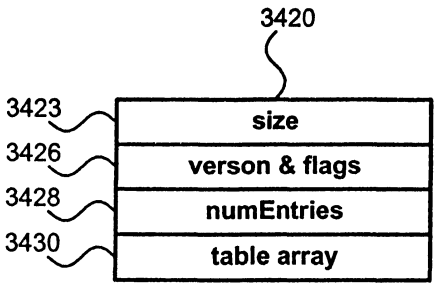


圖34B

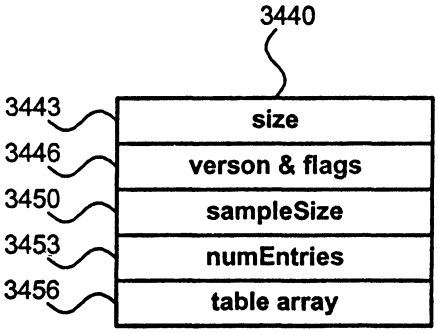


圖34C

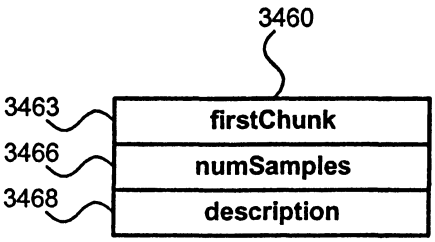


圖34D

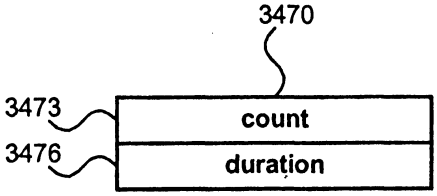


圖34E

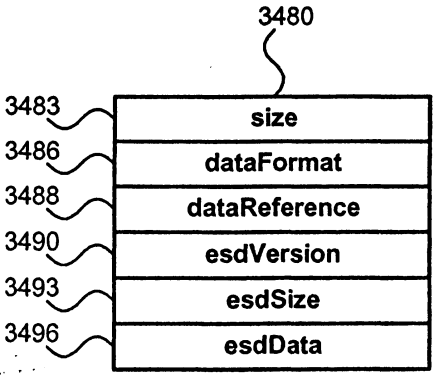


圖34F

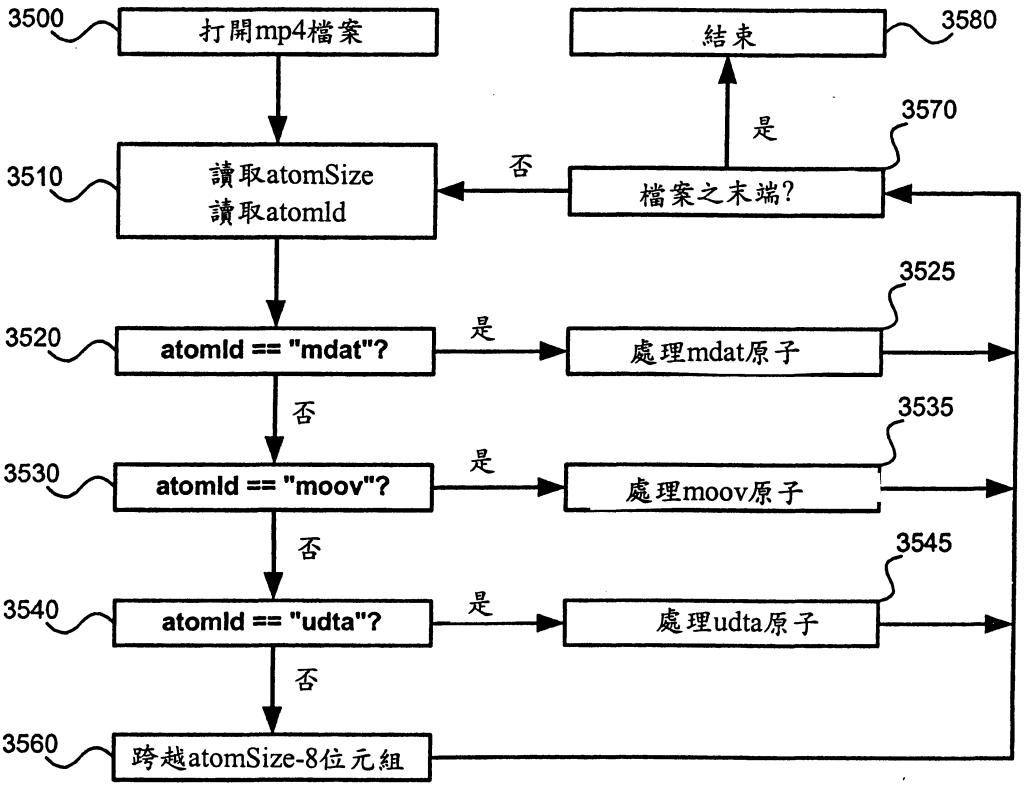

3500
打開mp4檔案
3510
讀取atomSize
讀取atomId
3520
atomId == "mdat"?
是
3525
處理mdat原子
否
3530
atomId == "moov"?
是
3535
處理moov原子
否
3540
atomId == "udta"?
是
3545
處理udta原子
否
3560
跨越atomSize-8位元組
檔案之末端?
否
是
3570
結束
3580

圖35

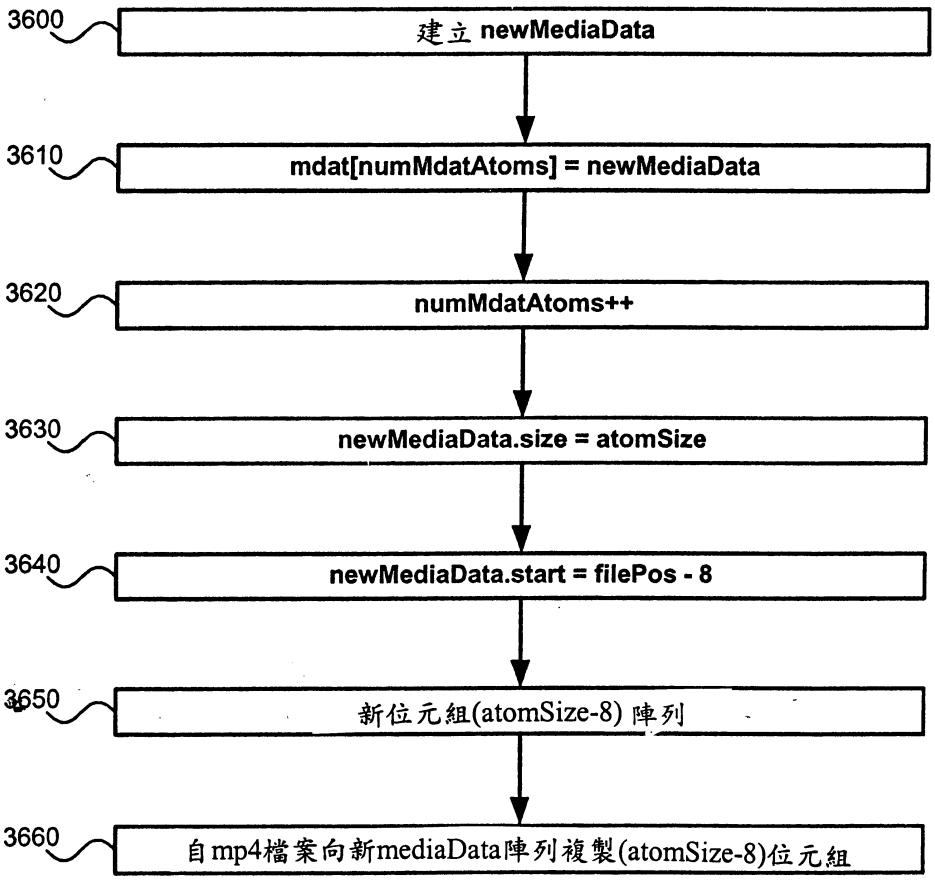

3600
建立 newMediaData
3610
mdat[numMdatAtoms] = newMediaData
3620
numMdatAtoms++
3630
newMediaData.size = atomSize
3640
newMediaData.start = filePos - 8
3650
新位元組(atomSize-8) 陣列
3660
自mp4檔案向新mediaData陣列複製(atomSize-8)位元組

圖36

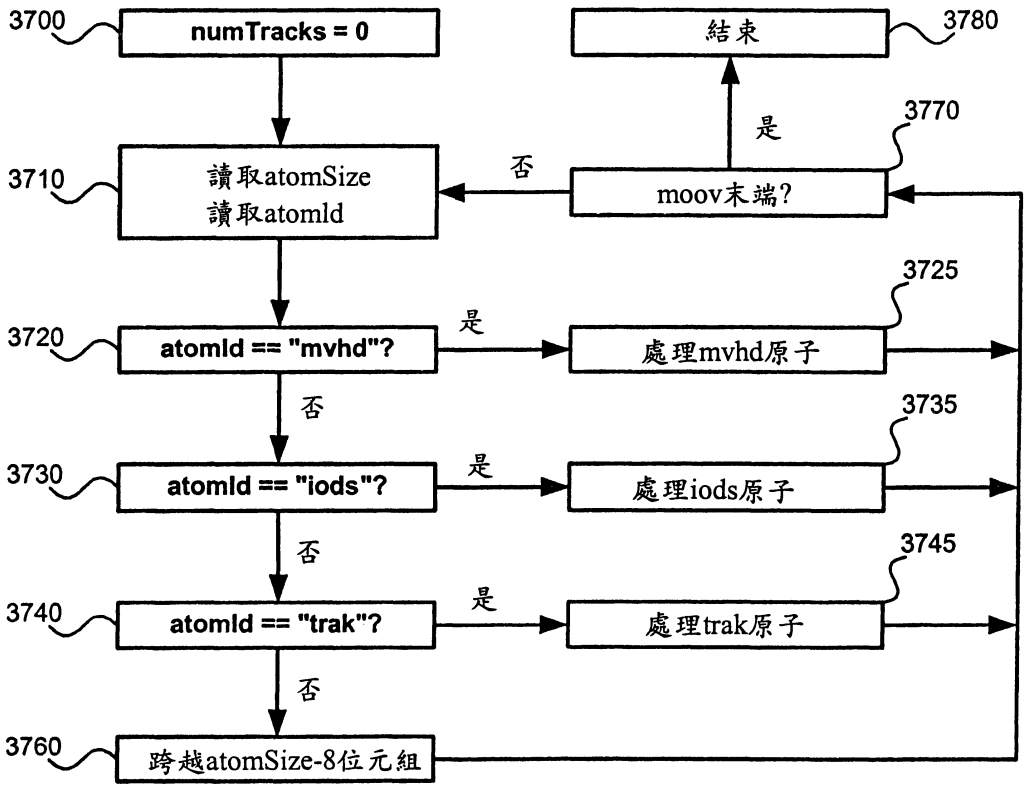

3700
numTracks = 0
3710
讀取atomSize
讀取atomId
3720
atomId == "mvhd"?
是
處理mvhd原子
3725
否
3730
atomId == "iods"?
是
處理iods原子
3735
否
3740
atomId == "trak"?
是
處理trak原子
3745
否
3760
跨越atomSize-8位元組
否
moov末端?
是
3770
結束
3780

圖37

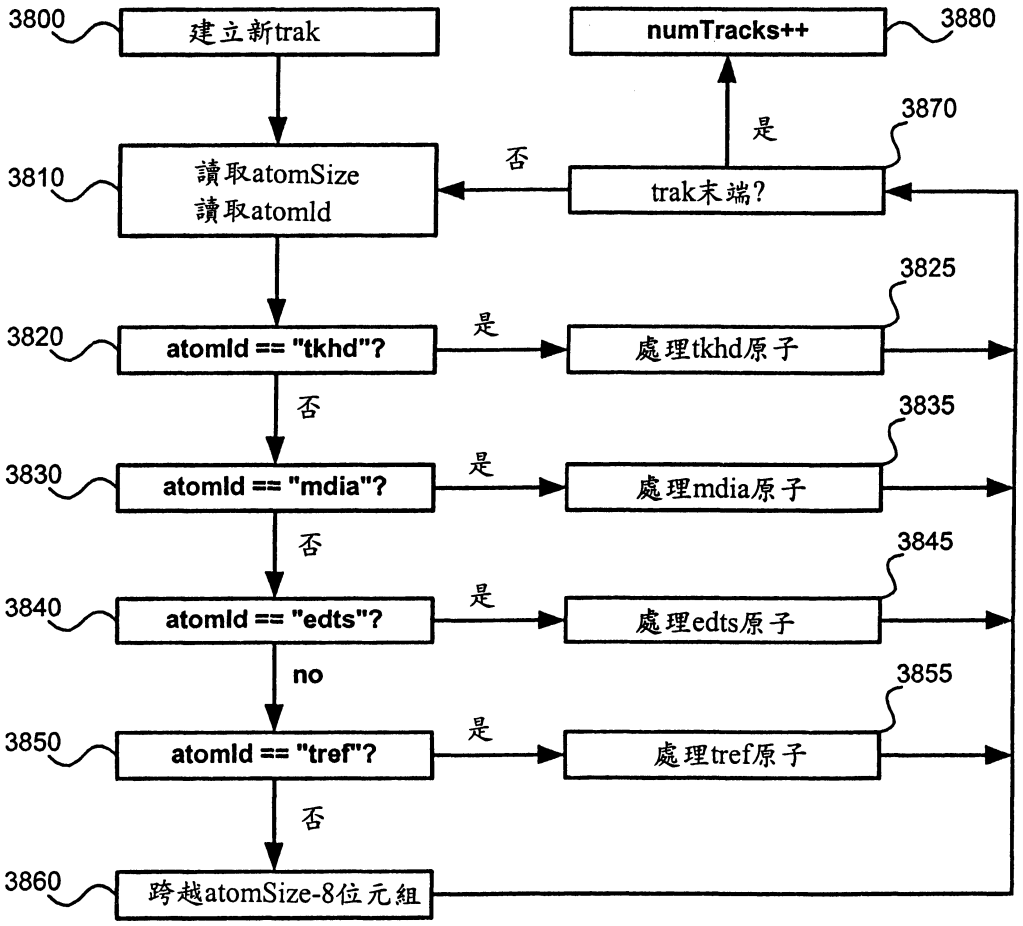

3800
建立新trak
3810
讀取atomSize
讀取atomId
3820
atomId == "tkhd"?
是
處理tkhd原子
3825
否
3830
atomId == "mdia"?
是
處理mdia原子
3835
否
3840
atomId == "edts"?
是
處理edts原子
3845
no
3850
atomId == "tref"?
是
處理tref原子
3855
否
3860
跨越atomSize-8位元組
否
trak末端?
是
3870
numTracks++
3880

圖38

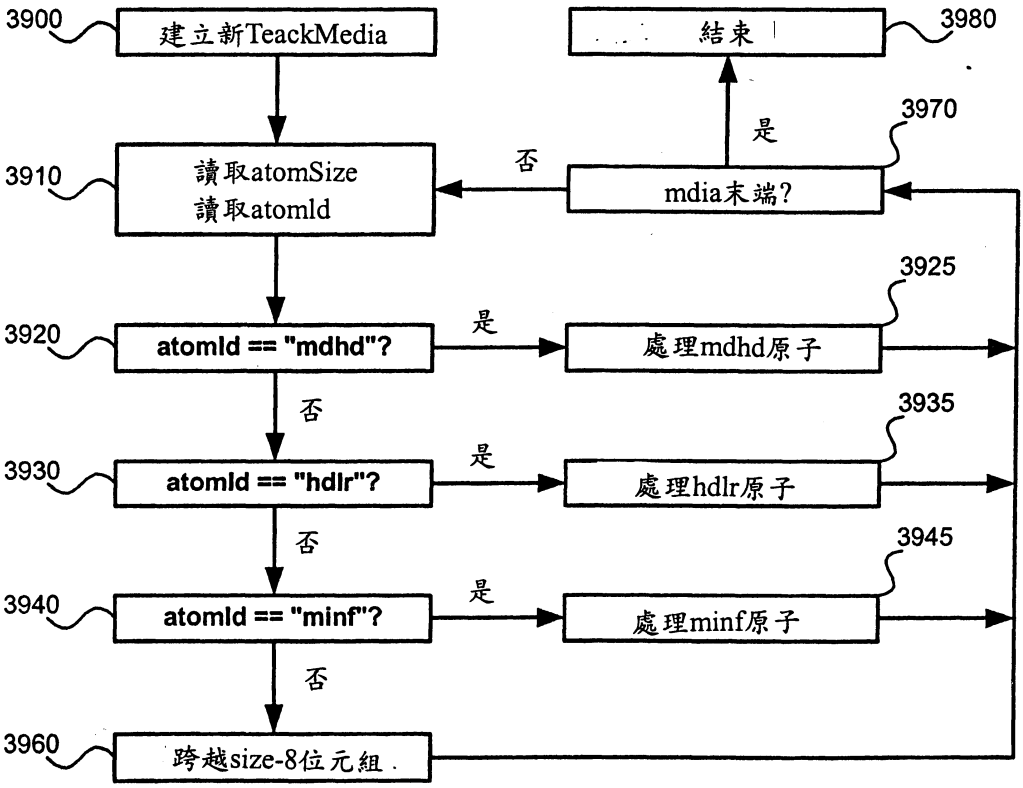

3900
建立新TeackMedia
3910
讀取atomSize
讀取atomId
3920
atomId == "mdhd"?
3930
atomId == "hdlr"?
3940
atomId == "minf"?
3960
跨越size-8位元組
否
是
3970
mdia末端?
3980
結束
3925
處理mdhd原子
3935
處理hdlr原子
3945
處理minf原子

圖39

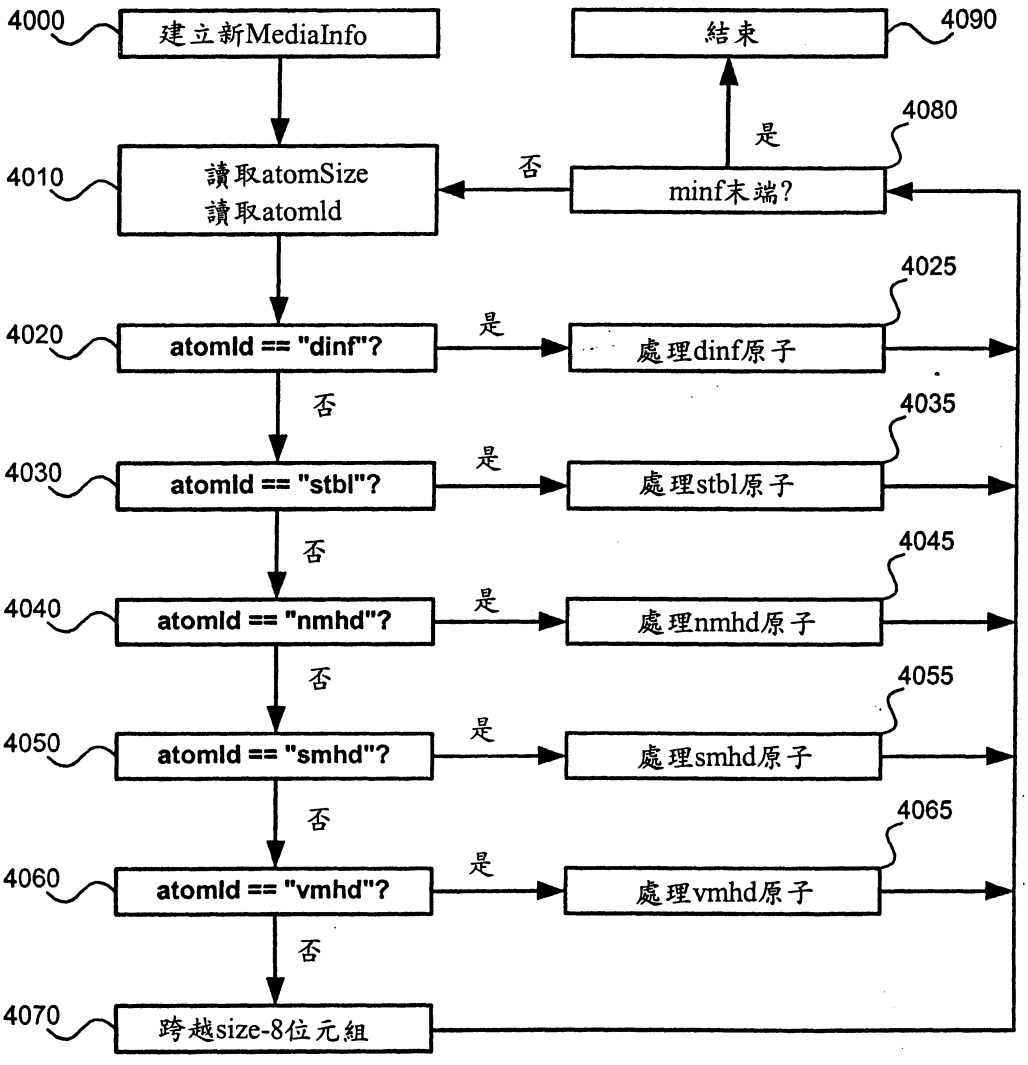

4000
建立新MediaInfo
4010
讀取atomSize
讀取atomId
4020
atomId == "dinf"?
是
處理dinf原子
4025
否
4030
atomId == "stbl"?
是
處理stbl原子
4035
否
4040
atomId == "nmhd"?
是
處理nmhd原子
4045
否
4050
atomId == "smhd"?
是
處理smhd原子
4055
否
4060
atomId == "vmhd"?
是
處理vmhd原子
4065
否
4070
跨越size-8位元組
4080
minf末端?
否
是
結束
4090

圖40

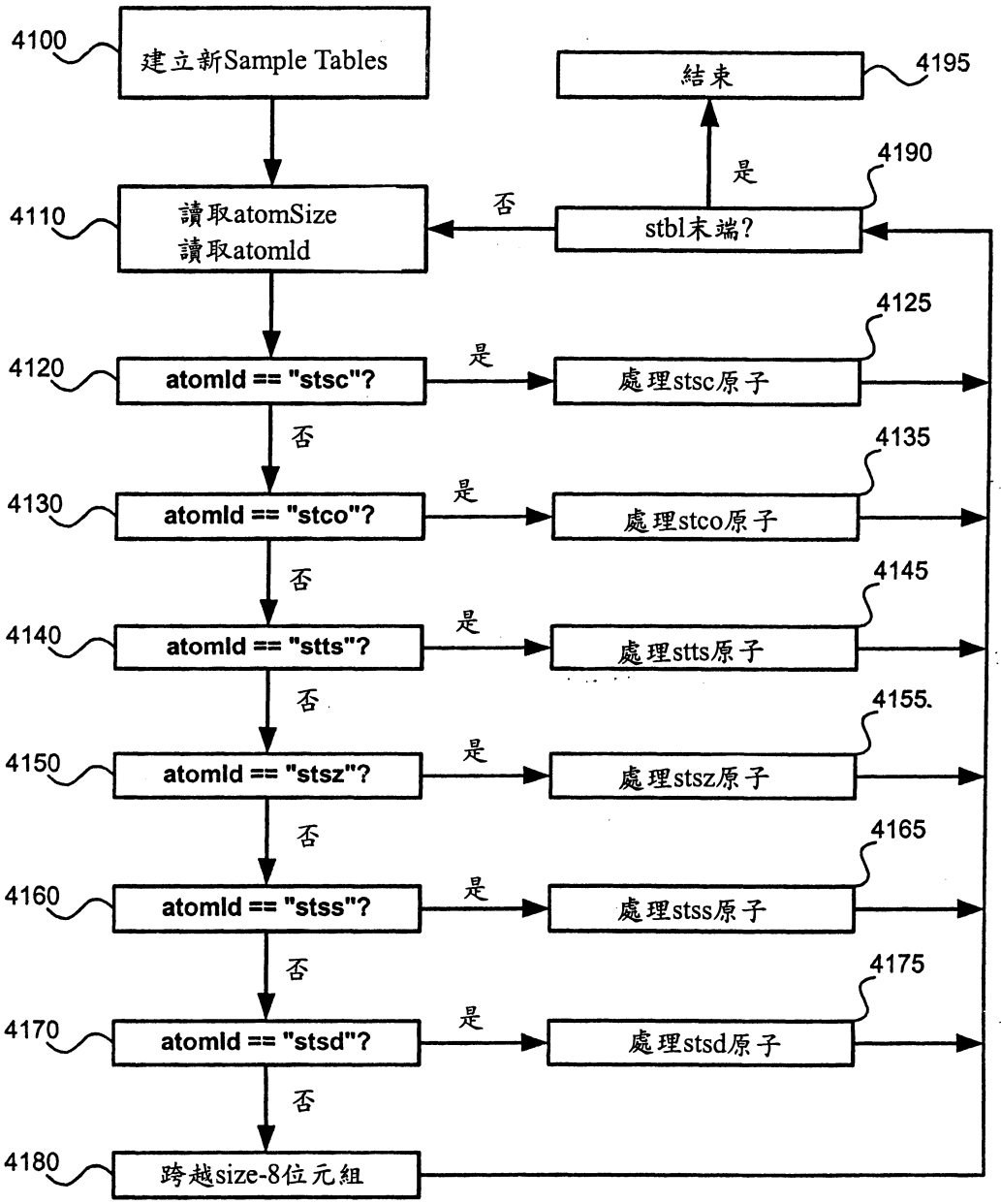

4100
建立新Sample Tables
4110
讀取atomSize
讀取atomId
4120
atomId == "stsc"?
4125
處理stsc原子
4130
atomId == "stco"?
4135
處理stco原子
4140
atomId == "stts"?
4145
處理stts原子
4150
atomId == "stsz"?
4155
處理stsz原子
4160
atomId == "stss"?
4165
處理stss原子
4170
atomId == "stsd"?
4175
處理stsd原子
4180
跨越size-8位元組
4190
stbl末端?
4195
結束
是
否

圖41

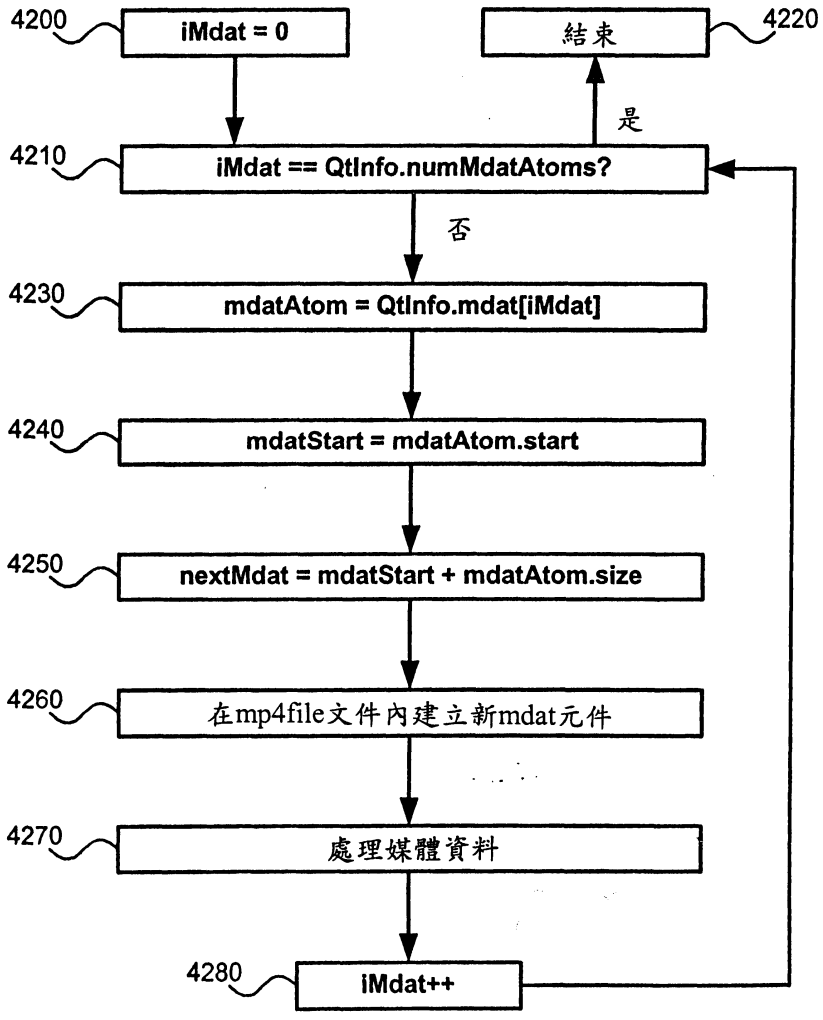

4200
iMdat = 0
結束
4220
是
4210
iMdat == QtInfo.numMdatAtoms?
否
4230
mdatAtom = QtInfo.mdat[iMdat]
4240
mdatStart = mdatAtom.start
4250
nextMdat = mdatStart + mdatAtom.size
4260
在mp4file文件內建立新mdat元件
4270
處理媒體資料
4280
iMdat++

圖42

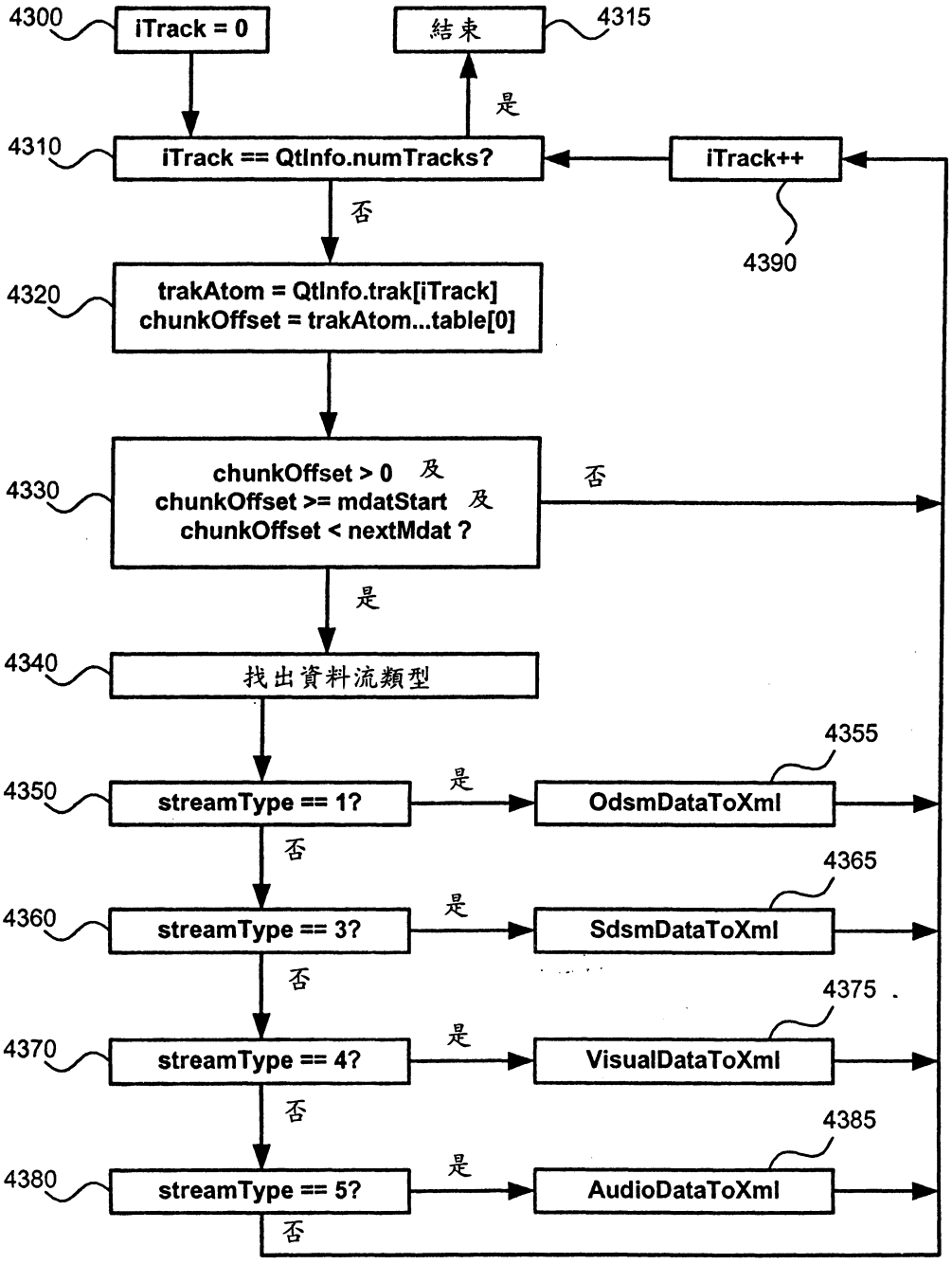

4300
iTrack = 0
結束
4315
是
4310
iTrack == QtInfo.numTracks?
iTrack++
否
4390
4320
trakAtom = QtInfo.trak[iTrack]
chunkOffset = trakAtom...table[0]
4330
chunkOffset > 0 及
chunkOffset >= mdatStart 及
chunkOffset < nextMdat ?
否
是
4340
找出資料流類型
4355
4350
streamType == 1?
是
OdsmDataToXml
否
4365
4360
streamType == 3?
是
SdsmDataToXml
否
4375
4370
streamType == 4?
是
VisualDataToXml
否
4385
4380
streamType == 5?
是
AudioDataToXml
否

圖43

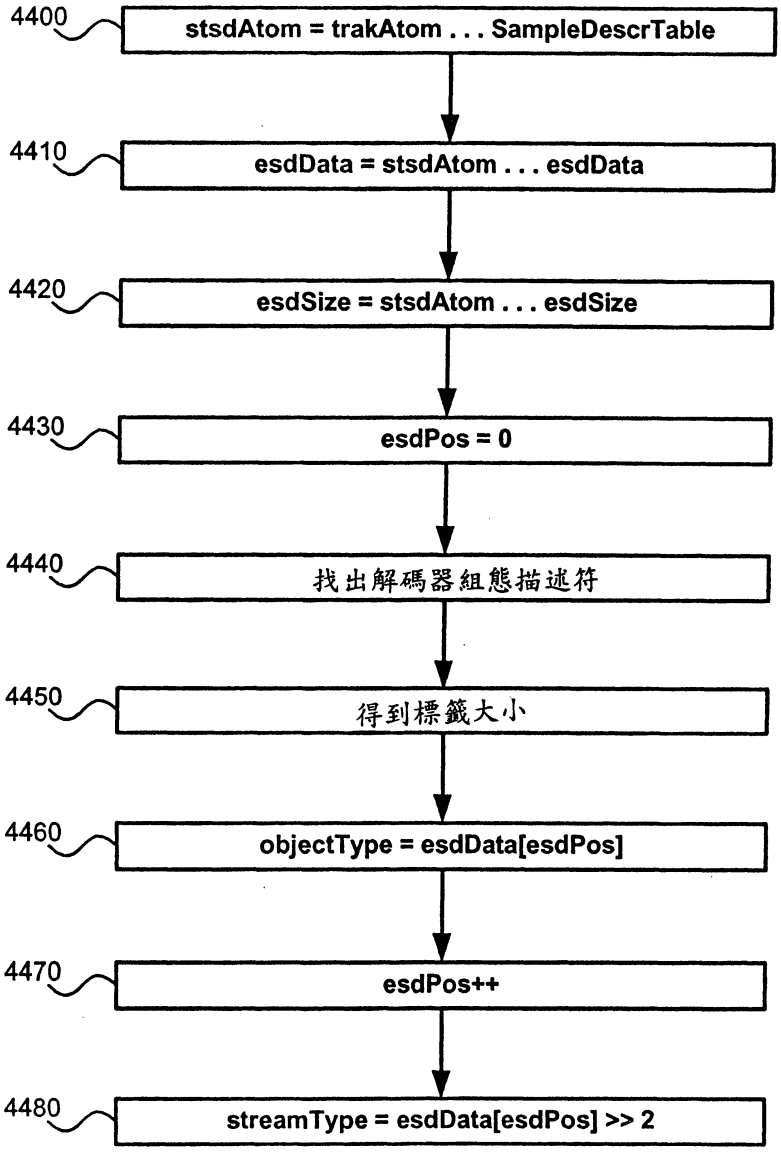

4400
stsdAtom = trakAtom . . . SampleDescrTable
4410
esdData = stsdAtom . . . esdData
4420
esdSize = stsdAtom . . . esdSize
4430
esdPos = 0
4440
找出解碼器組態描述符
4450
得到標籤大小
4460
objectType = esdData[esdPos]
4470
esdPos++
4480
streamType = esdData[esdPos] >> 2

圖44

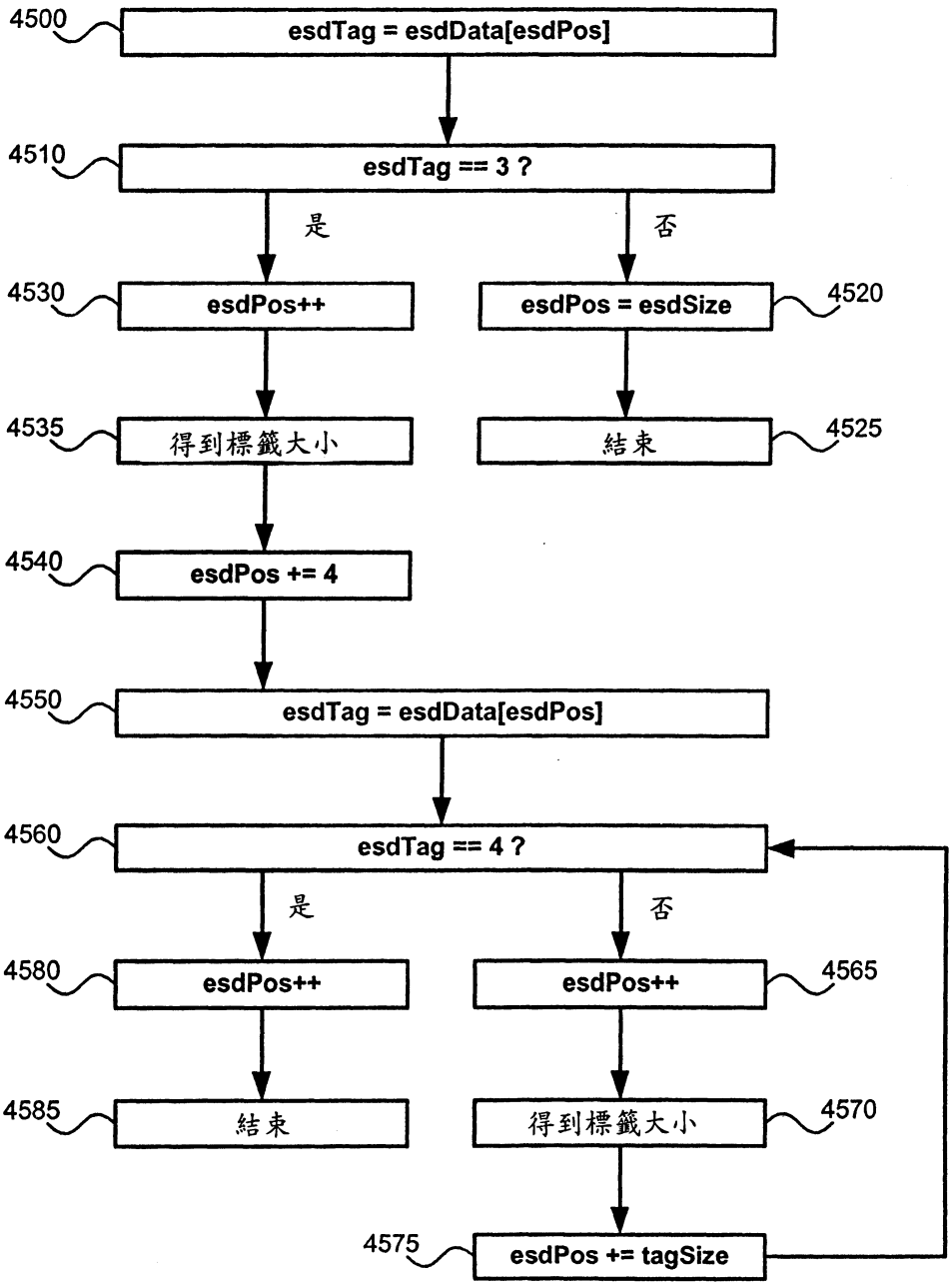

4500
esdTag = esdData[esdPos]
4510
esdTag == 3 ?
是
否
4530
esdPos++
esdPos = esdSize
4520
4535
得到標籤大小
結束
4525
4540
esdPos += 4
4550
esdTag = esdData[esdPos]
4560
esdTag == 4 ?
是
否
4580
esdPos++
esdPos++
4565
4585
結束
得到標籤大小
4570
4575
esdPos += tagSize

圖45

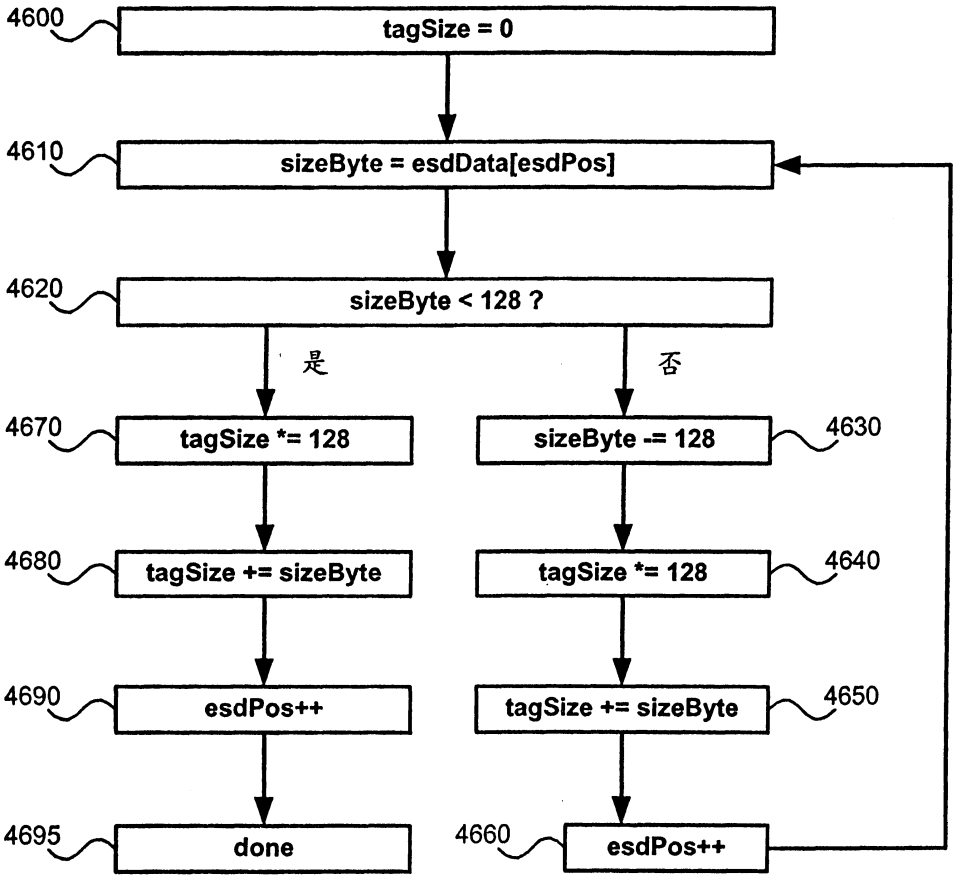

4600
tagSize = 0
4610
sizeByte = esdData[esdPos]
4620
sizeByte < 128 ?
是
否
4670
tagSize *= 128
4680
tagSize += sizeByte
4690
esdPos++
4695
done
sizeByte -= 128
4630
tagSize *= 128
4640
tagSize += sizeByte
4650
4660
esdPos++

圖46

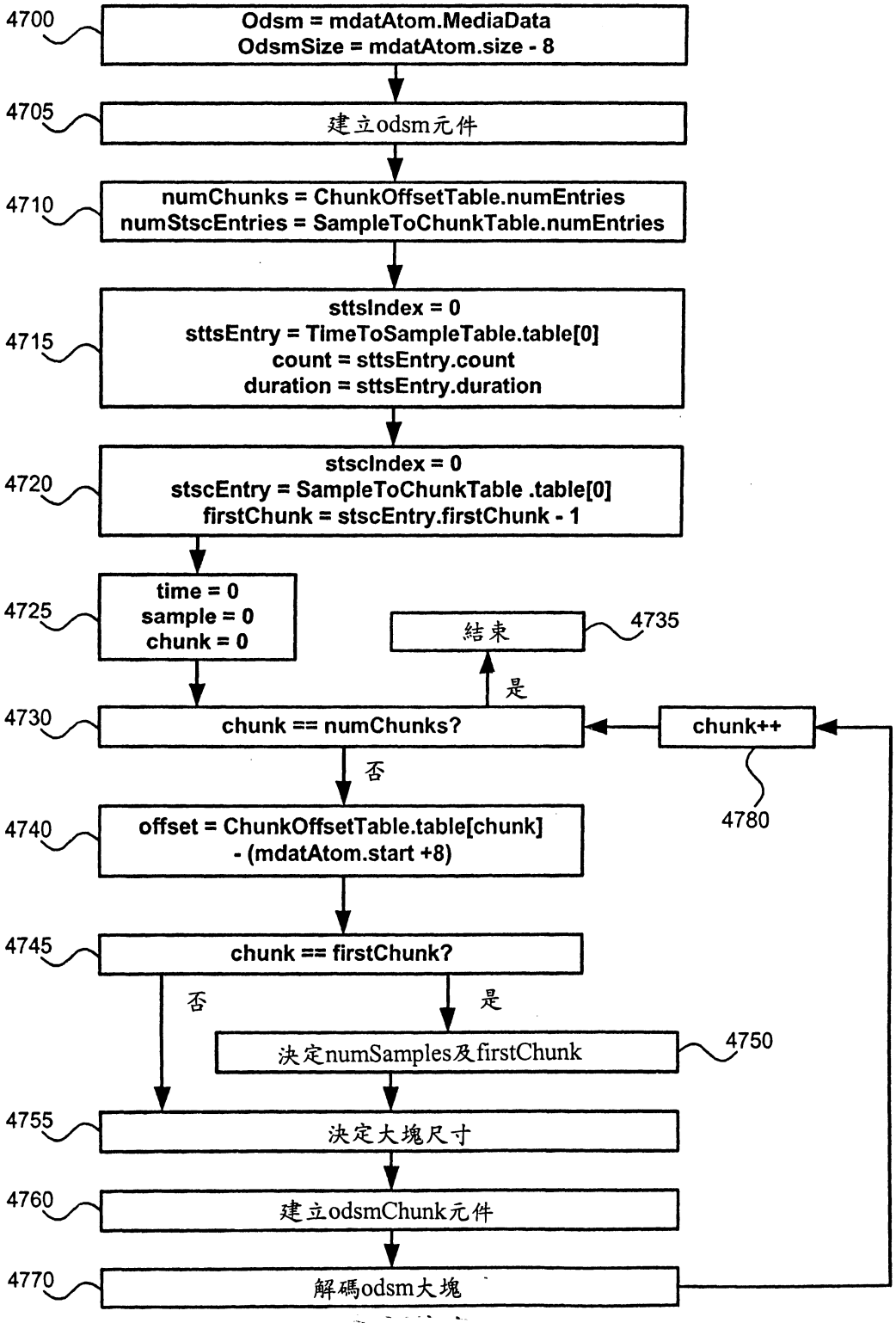

4700
Odsm = mdatAtom.MediaData
OdsmSize = mdatAtom.size - 8
4705
建立odsm元件
4710
numChunks = ChunkOffsetTable.numEntries
numStscEntries = SampleToChunkTable.numEntries
4715
sttsIndex = 0
sttsEntry = TimeToSampleTable.table[0]
count = sttsEntry.count
duration = sttsEntry.duration
4720
stscIndex = 0
stscEntry = SampleToChunkTable .table[0]
firstChunk = stscEntry.firstChunk - 1
4725
time = 0
sample = 0
chunk = 0
結束
4735
是
4730
chunk == numChunks?
chunk++
否
4780
4740
offset = ChunkOffsetTable.table[chunk]
- (mdatAtom.start +8)
4745
chunk == firstChunk?
否
是
決定numSamples及firstChunk
4750
4755
決定大塊尺寸
4760
建立odsmChunk元件
4770
解碼odsm大塊

圖47

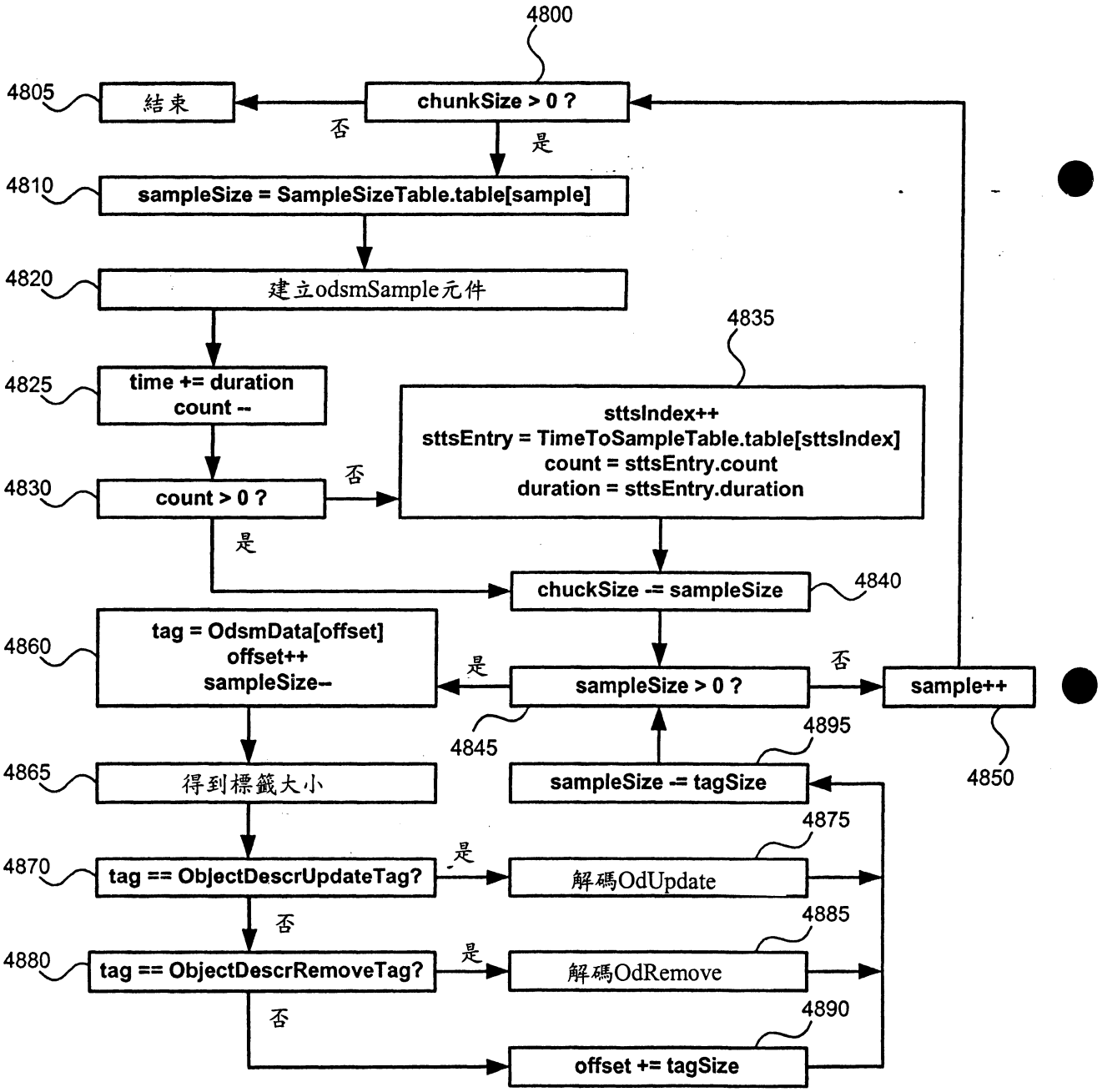

4800
4805
結束
chunkSize > 0 ?
否
是
4810
sampleSize = SampleSizeTable.table[sample]
4820
建立odsmSample元件
4835
4825
time += duration
count --
sttsIndex++
sttsEntry = TimeToSampleTable.table[sttsIndex]
count = sttsEntry.count
duration = sttsEntry.duration
4830
count > 0 ?
否
是
chuckSize -= sampleSize
4840
4860
tag = OdsmData[offset]
offset++
sampleSize--
是
sampleSize > 0 ?
否
sample++
4845
4895
4850
4865
得到標籤大小
sampleSize -= tagSize
4875
4870
tag == ObjectDescrUpdateTag?
是
解碼OdUpdate
否
4885
4880
tag == ObjectDescrRemoveTag?
是
解碼OdRemove
否
4890
offset += tagSize

圖48

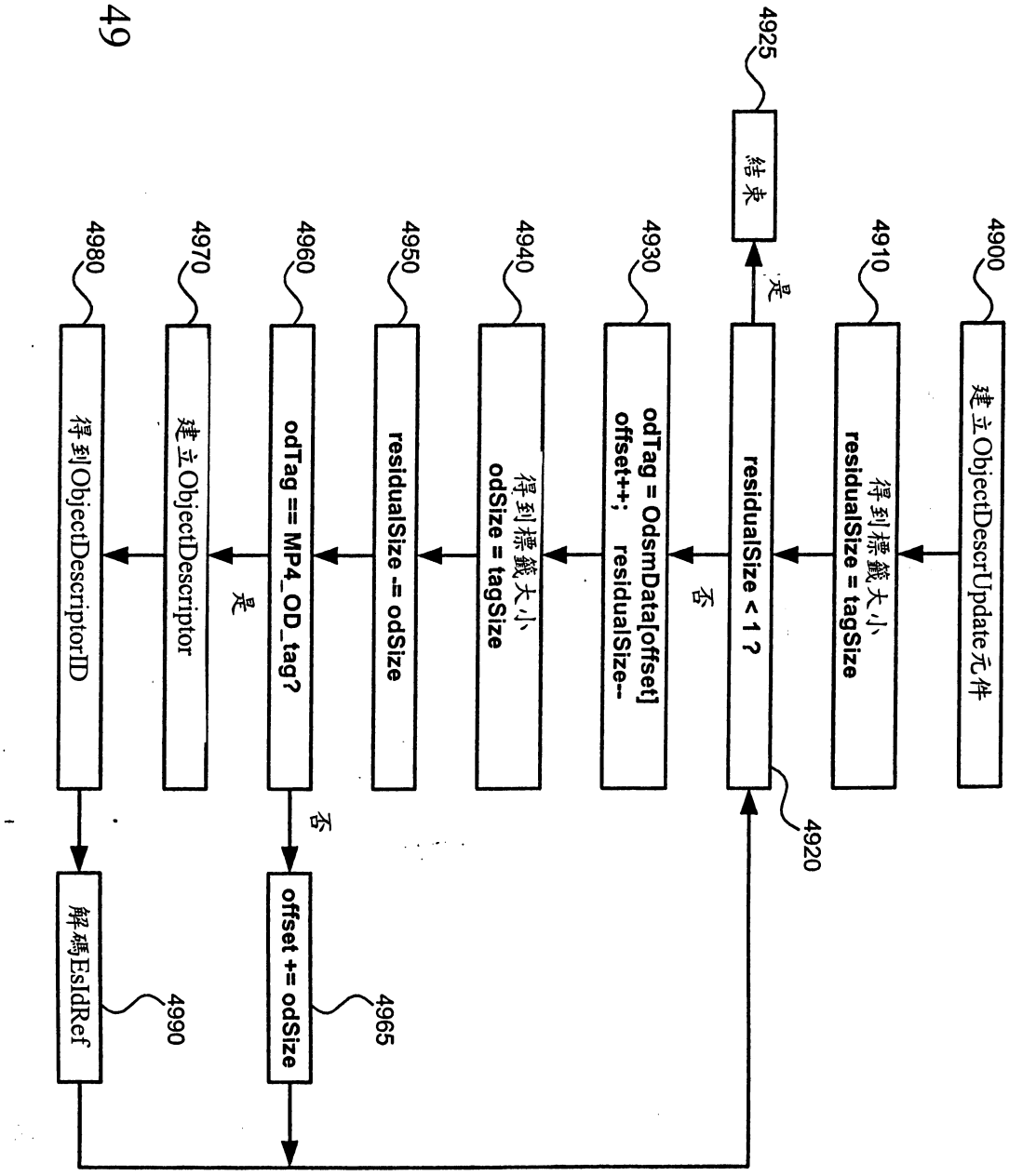

4900
建立ObjectDescrUpdate元件
4910
得到標籤大小
residualSize = tagSize
4920
residualSize < 1 ?
是
4925
結束
否
4930
odTag = OdsmData[offset]
offset++; residualSize--
4940
得到標籤大小
odSize = tagSize
4950
residualSize -= odSize
4960
odTag == MP4_OD_tag?
是
4970
建立ObjectDescriptor
否
4980
得到ObjectDescriptorID
4965
offset += odSize
4990
解碼EsIdRef

圖49

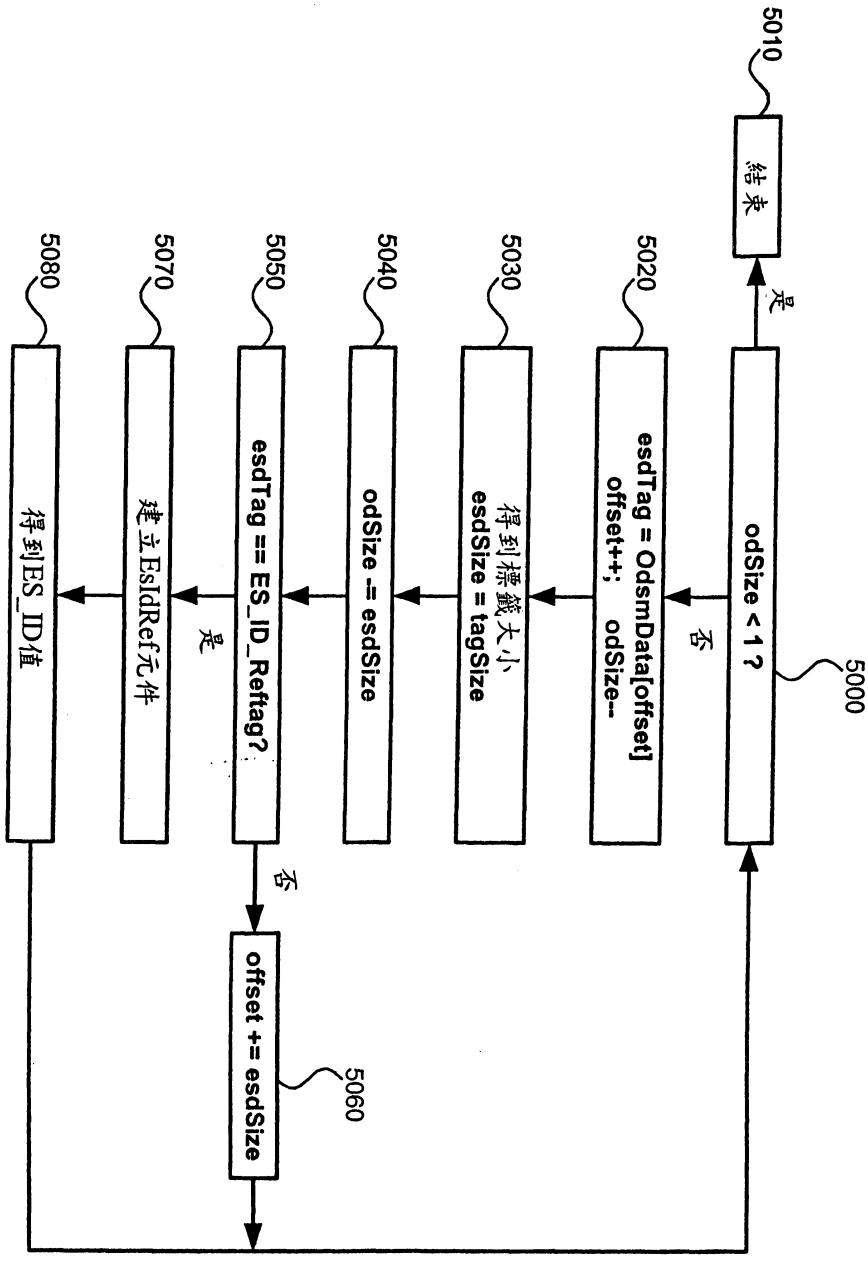

5010
結束
是
5000
odSize < 1 ?
否
5020
esdTag = OdsmData[offset]
offset++; odSize--
5030
得到標籤大小
esdSize = tagSize
5040
odSize -= esdSize
5050
esdTag == ES_ID_Reftag?
是
5070
建立EsIdRef元件
5080
得到ES_ID值
否
5060
offset += esdSize

圖 50

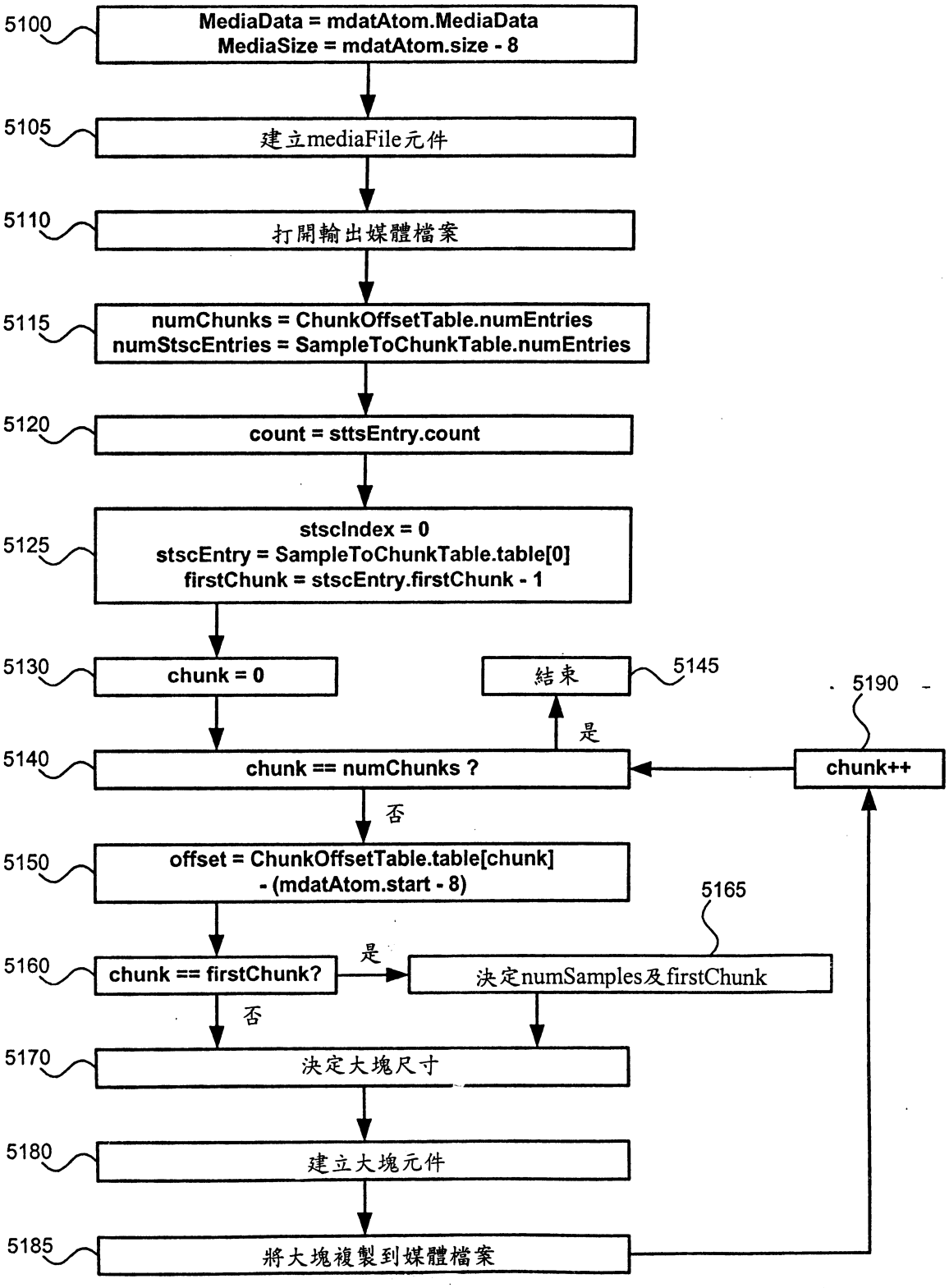

5100
MediaData = mdatAtom.MediaData
MediaSize = mdatAtom.size - 8
5105
建立mediaFile元件
5110
打開輸出媒體檔案
5115
numChunks = ChunkOffsetTable.numEntries
numStscEntries = SampleToChunkTable.numEntries
5120
count = sttsEntry.count
5125
stscIndex = 0
stscEntry = SampleToChunkTable.table[0]
firstChunk = stscEntry.firstChunk - 1
5130
chunk = 0
5140
chunk == numChunks ?
是
結束
5145
否
5150
offset = ChunkOffsetTable.table[chunk]
- (mdatAtom.start - 8)
5160
chunk == firstChunk?
是
5165
決定numSamples及firstChunk
否
5170
決定大塊尺寸
5180
建立大塊元件
5185
將大塊複製到媒體檔案
5190
chunk++

圖51

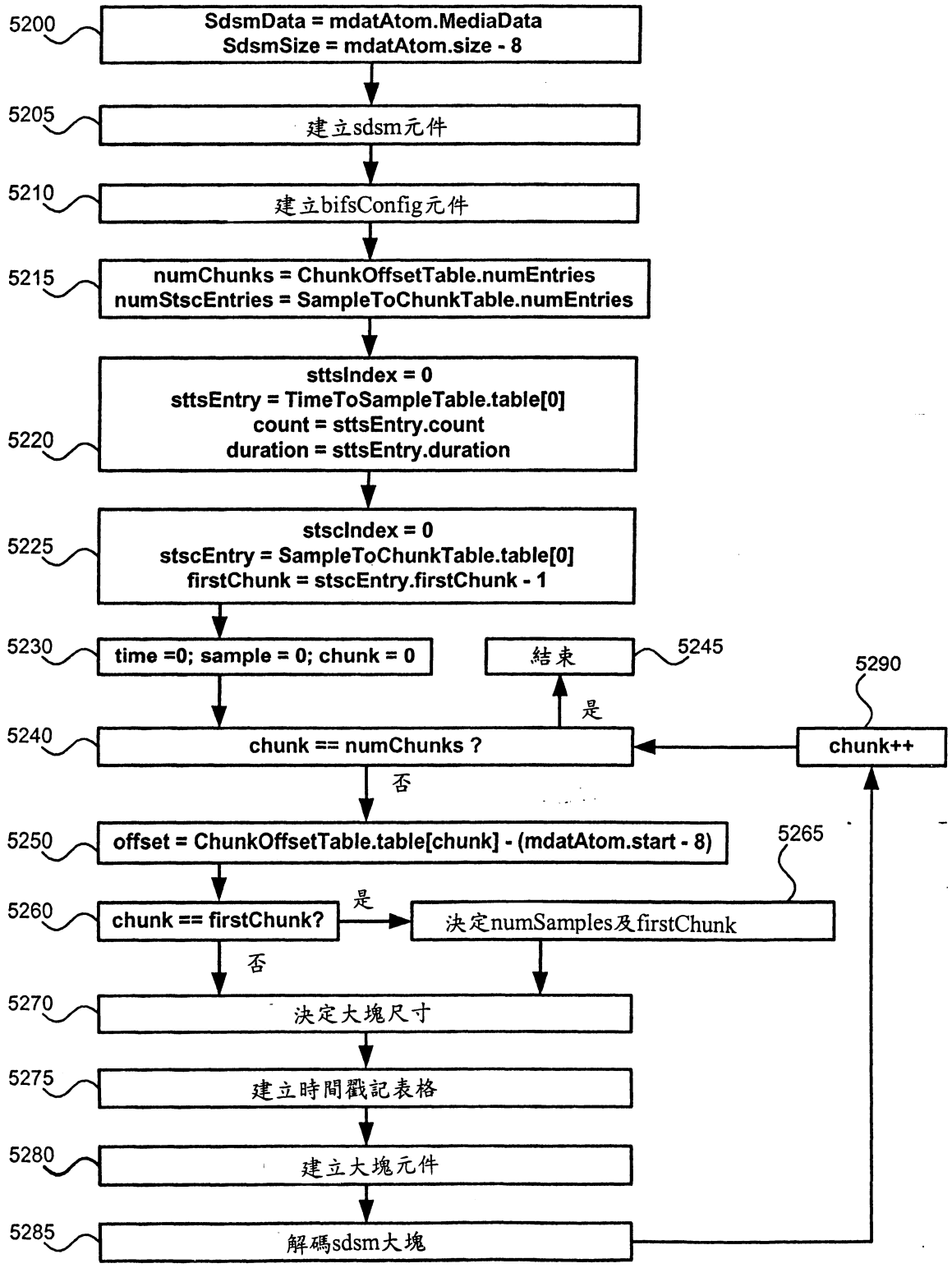

5200
SdsmData = mdatAtom.MediaData
SdsmSize = mdatAtom.size - 8
5205
建立sdsm元件
5210
建立bifsConfig元件
5215
numChunks = ChunkOffsetTable.numEntries
numStscEntries = SampleToChunkTable.numEntries
5220
sttsIndex = 0
sttsEntry = TimeToSampleTable.table[0]
count = sttsEntry.count
duration = sttsEntry.duration
5225
stscIndex = 0
stscEntry = SampleToChunkTable.table[0]
firstChunk = stscEntry.firstChunk - 1
5230
time =0; sample = 0; chunk = 0
結束
5245
5290
是
5240
chunk == numChunks ?
chunk++
否
5250
offset = ChunkOffsetTable.table[chunk] - (mdatAtom.start - 8)
5265
5260
chunk == firstChunk?
是
決定numSamples及firstChunk
否
5270
決定大塊尺寸
5275
建立時間戳記表格
5280
建立大塊元件
5285
解碼sdsm大塊

圖52

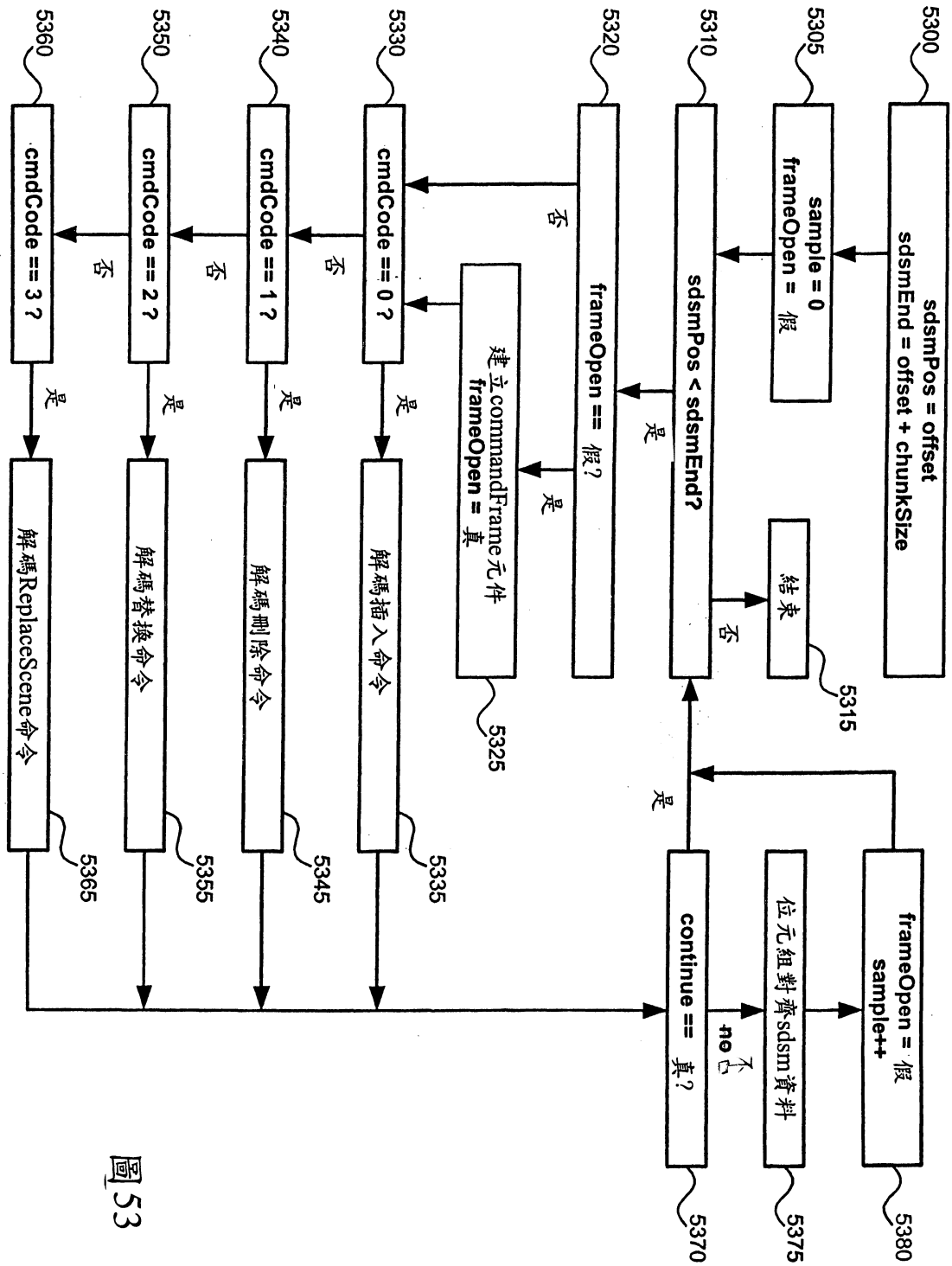

5300
sdsmPos = offset
sdsmEnd = offset + chunkSize
5305
sample = 0
frameOpen = 假
5310
sdsmPos < sdsmEnd?
是
否
5315
結束
5320
frameOpen == 假?
是
否
5325
建立commandFrame元件
frameOpen = 真
5330
cmdCode == 0 ?
是
否
5335
解碼插入命令
5340
cmdCode == 1 ?
是
否
5345
解碼刪除命令
5350
cmdCode == 2 ?
是
否
5355
解碼替換命令
5360
cmdCode == 3 ?
是
否
5365
解碼ReplaceScene命令
5370
continue == 真?
是
否
5375
位元組對齊sdsm資料
5380
frameOpen = 假
sample++

圖53

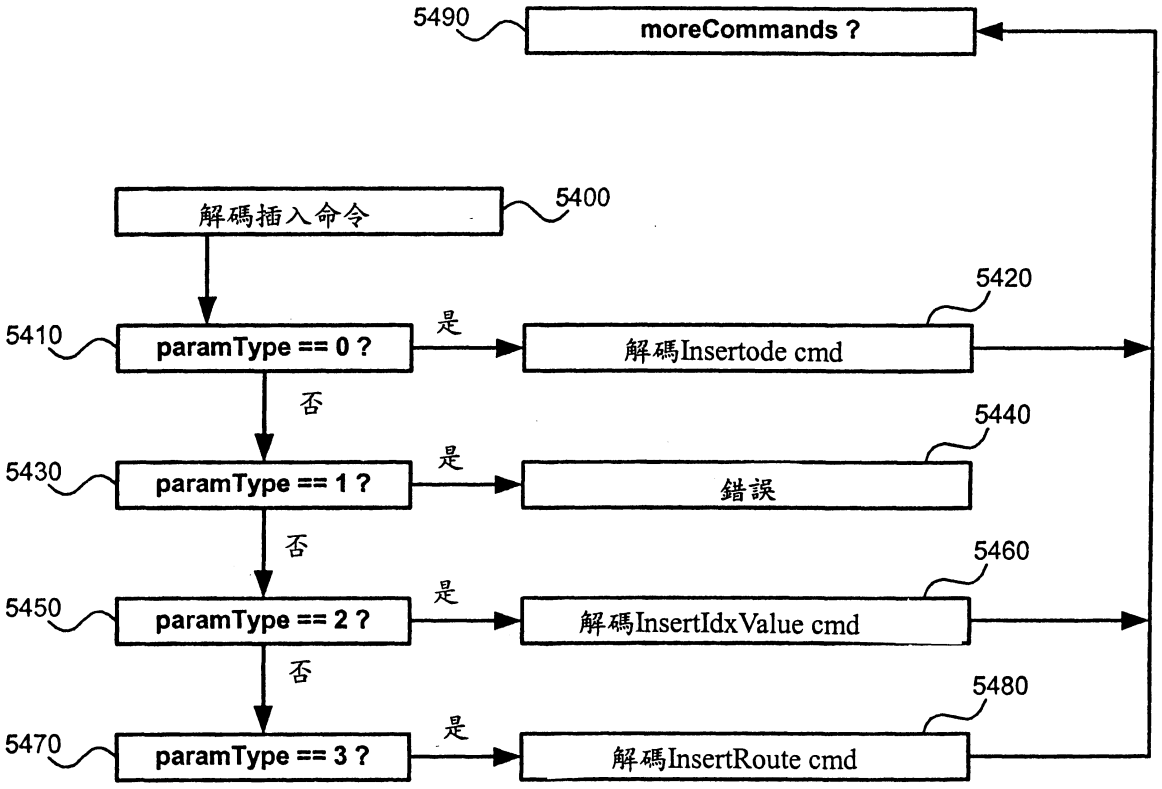

5490
moreCommands ?
解碼插入命令
5400
5410
paramType == 0 ?
是
5420
解碼Insertode cmd
否
5430
paramType == 1 ?
是
5440
錯誤
否
5450
paramType == 2 ?
是
5460
解碼InsertIdxValue cmd
否
5470
paramType == 3 ?
是
5480
解碼InsertRoute cmd

圖54

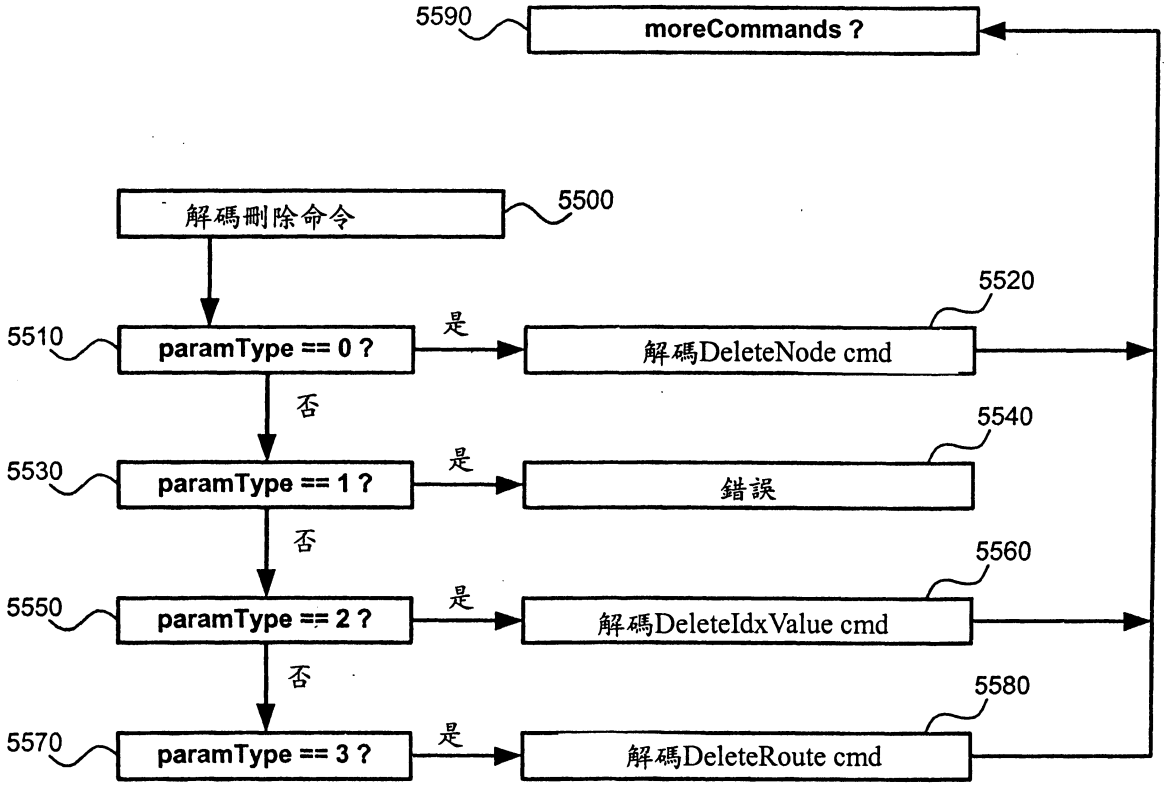

5590
moreCommands ?
解碼刪除命令
5500
5520
5510
paramType == 0 ?
是
解碼DeleteNode cmd
否
5540
5530
paramType == 1 ?
是
錯誤
否
5560
5550
paramType == 2 ?
是
解碼DeleteIdxValue cmd
否
5580
5570
paramType == 3 ?
是
解碼DeleteRoute cmd

圖55

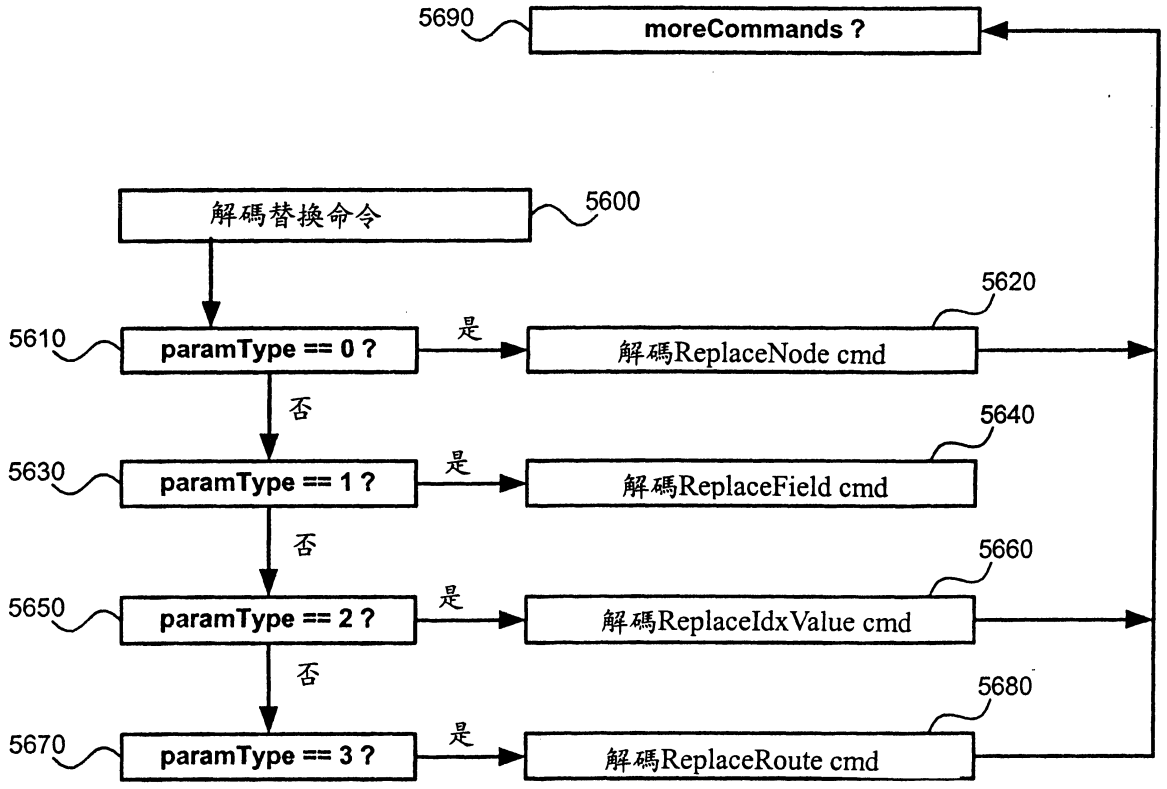

5690
moreCommands ?
解碼替換命令
5600
5610
paramType == 0 ?
是
5620
解碼ReplaceNode cmd
否
5630
paramType == 1 ?
是
5640
解碼ReplaceField cmd
否
5650
paramType == 2 ?
是
5660
解碼ReplaceIdxValue cmd
否
5670
paramType == 3 ?
是
5680
解碼ReplaceRoute cmd

圖56

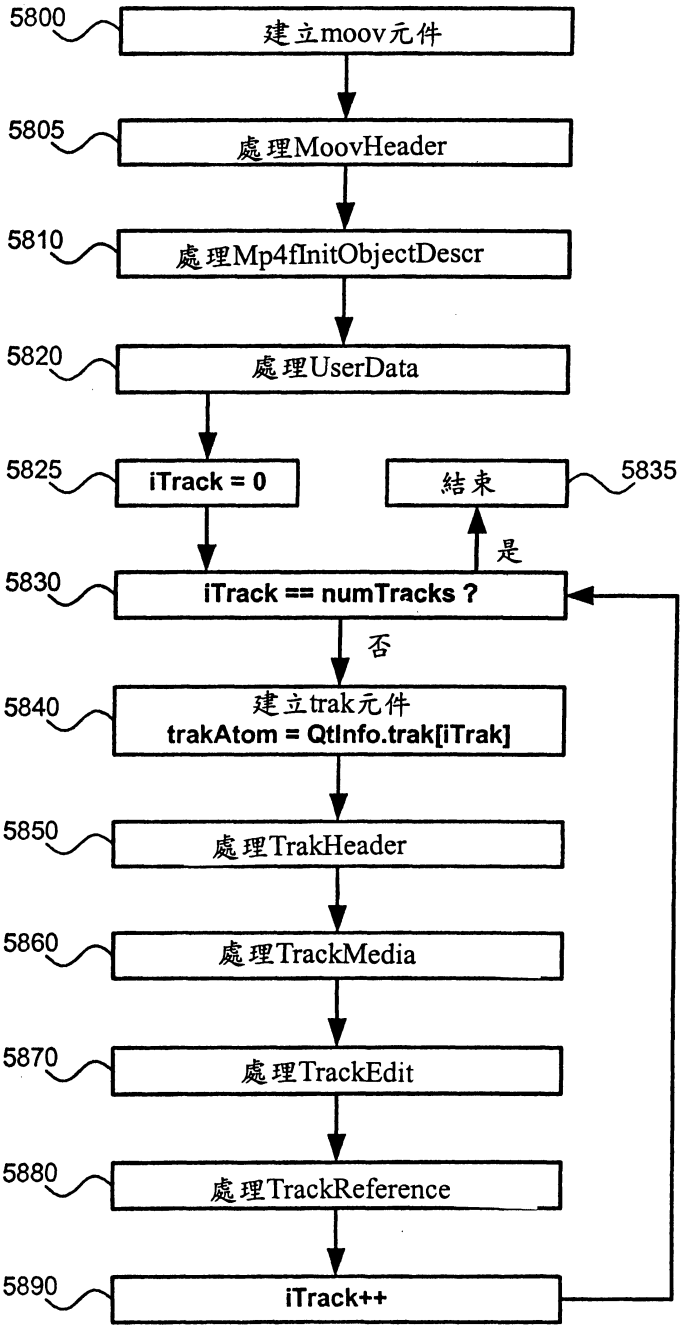

5800
建立moov元件
5805
處理MoovHeader
5810
處理Mp4fInitObjectDescr
5820
處理UserData
5825
iTrack = 0
結束
5835
是
5830
iTrack == numTracks ?
否
5840
建立trak元件
trakAtom = QtInfo.trak[iTrak]
5850
處理TrakHeader
5860
處理TrackMedia
5870
處理TrackEdit
5880
處理TrackReference
5890
iTrack++

圖58

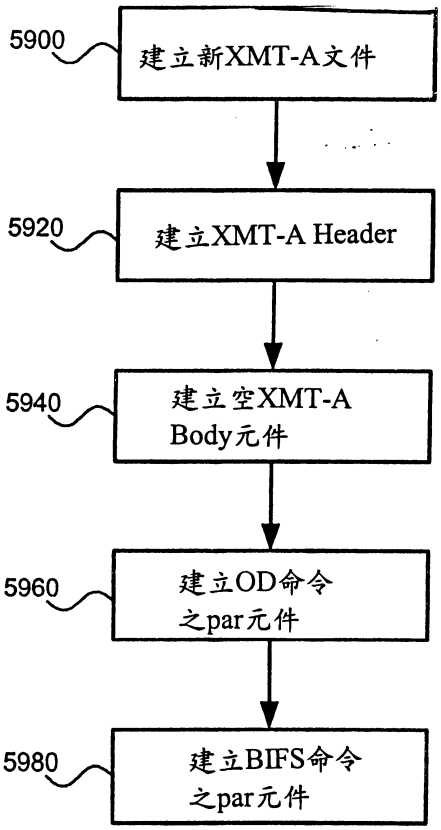

5900
建立新XMT-A文件
5920
建立XMT-A Header
5940
建立空XMT-A Body元件
5960
建立OD命令之par元件
5980
建立BIFS命令之par元件

圖59

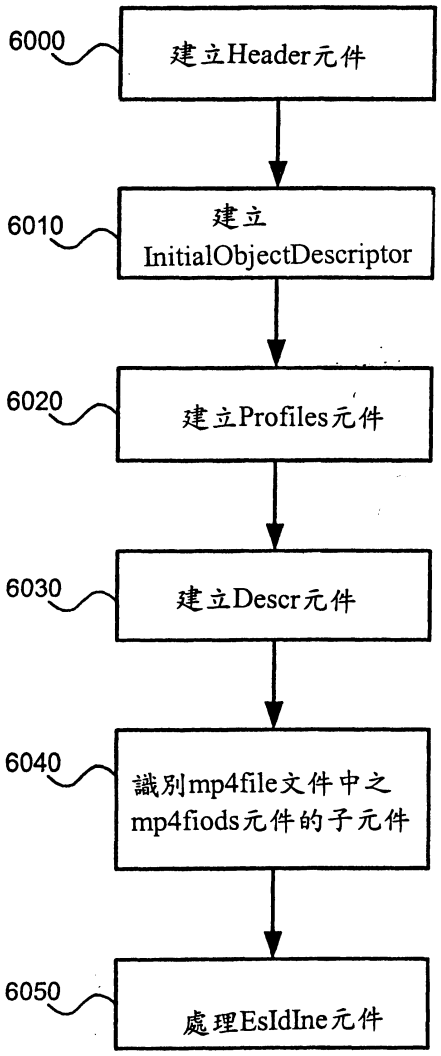

6000
建立Header元件
6010
建立
InitialObjectDescriptor
6020
建立Profiles元件
6030
建立Descr元件
6040
識別mp4file文件中之
mp4fiods元件的子元件
6050
處理EsIdIne元件

圖60

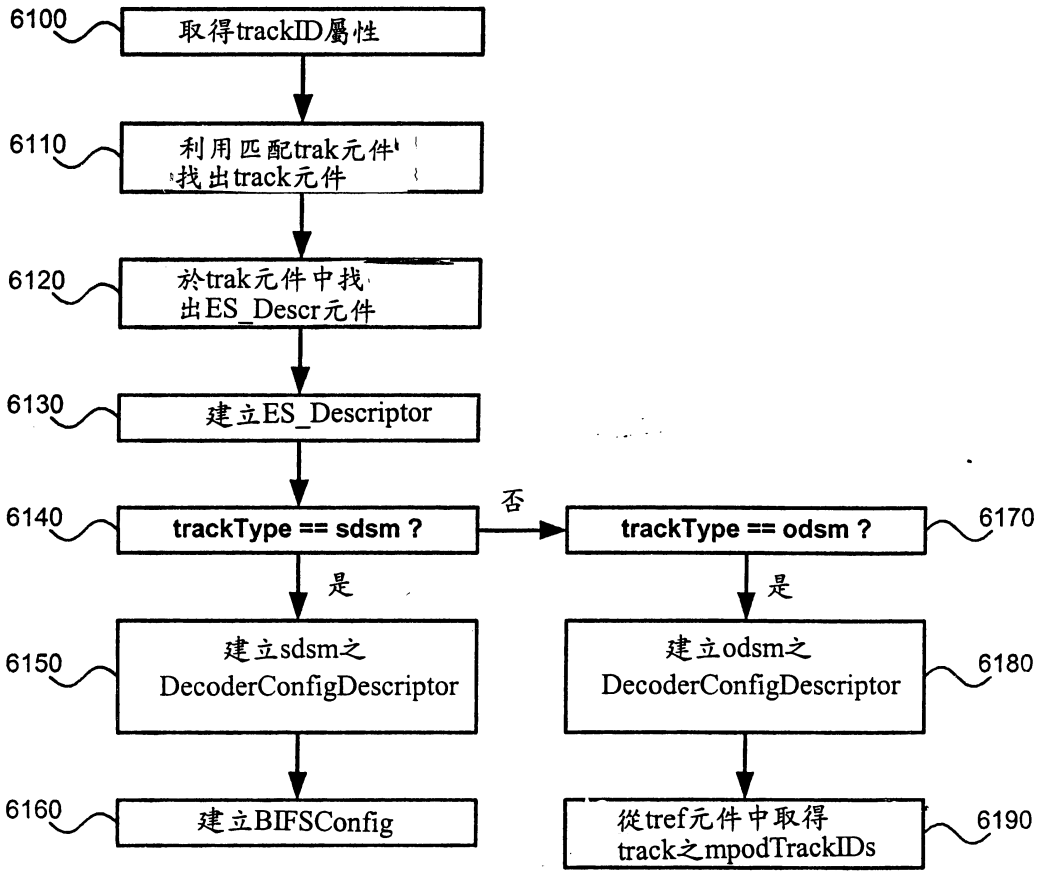
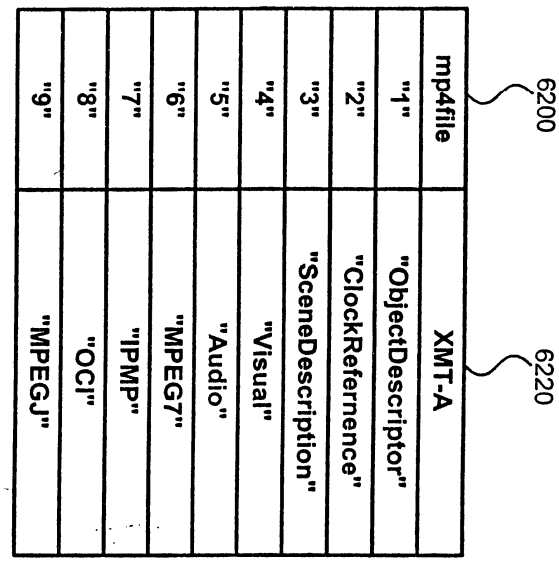

6100
取得trackID屬性
6110
利用匹配trak元件
找出track元件
6120
於trak元件中找
出ES_Descr元件
6130
建立ES_Descriptor
6140
trackType == sdsm ?
否
trackType == odsm ?
6170
是
是
6150
建立sdsm之
DecoderConfigDescriptor
建立odsm之
DecoderConfigDescriptor
6180
6160
建立BIFSConfig
從tref元件中取得
track之mpodTrackIDs
6190

圖61

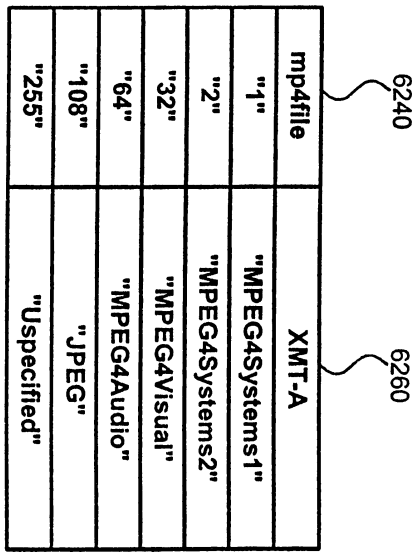


mp4file	XMT-A
"1"	"ObjectDescriptor"
"2"	"ClockReference"
"3"	"SceneDescription"
"4"	"Visual"
"5"	"Audio"
"6"	"MPEG7"
"7"	"IPMP"
"8"	"OCI"
"9"	"MPEGJ"

6200

6220

圖62A



mp4file	XMT-A
"1"	"MPEG4Systems1"
"2"	"MPEG4Systems2"
"32"	"MPEG4Visual"
"64"	"MPEG4Audio"
"108"	"JPEG"
"255"	"Uspecified"

6240

6260

圖62B

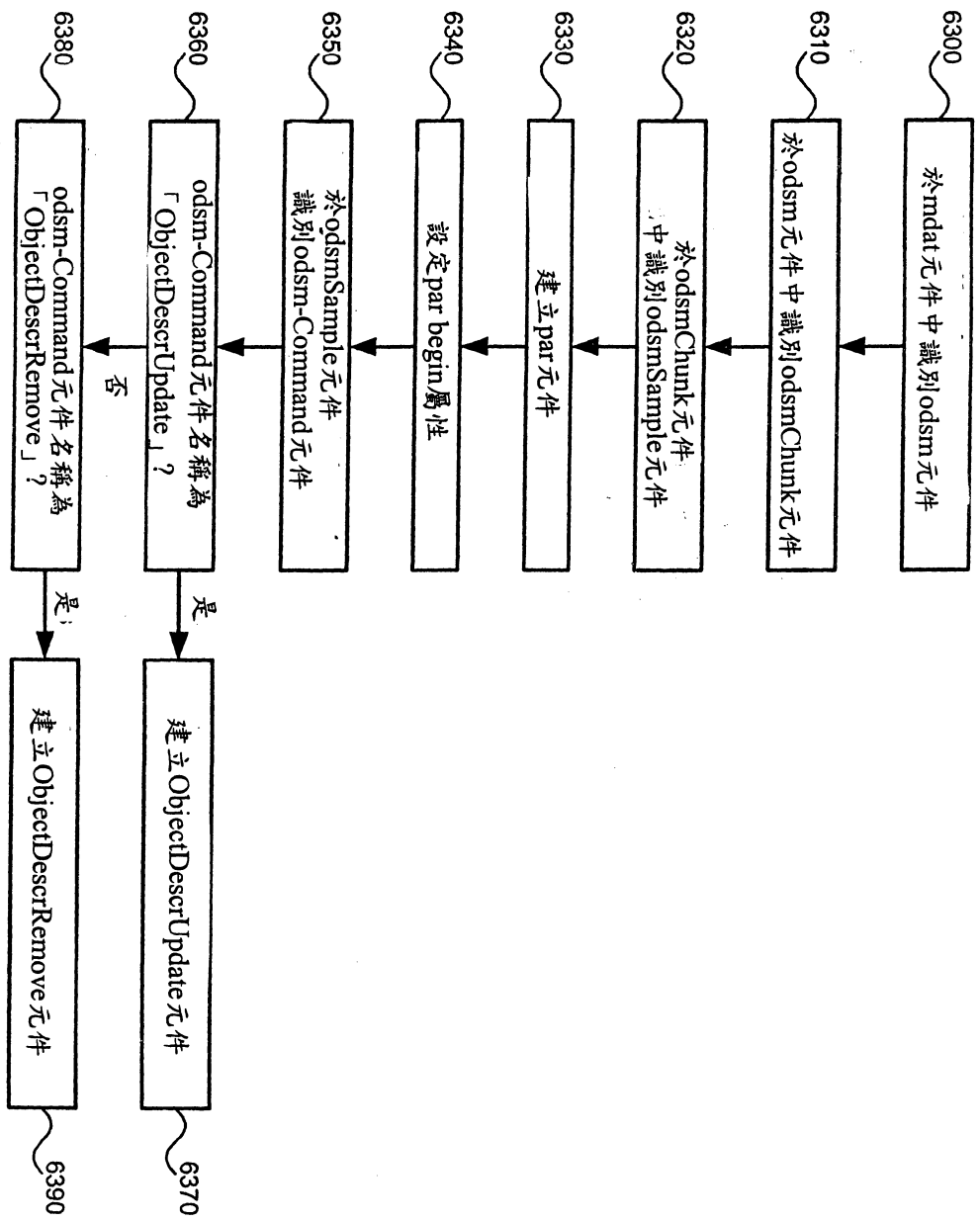

6300
於mdat元件中識別odsm元件
6310
於odsm元件中識別odsmChunk元件
6320
於odsmChunk元件中識別odsmSample元件
6330
建立par元件
6340
設定par begin屬性
6350
於odsmSample元件識別odsm-Command元件
6360
odsm-Command元件名稱為「ObjectDescrUpdate」?
是
否
6370
建立ObjectDescrUpdate元件
6380
odsm-Command元件名稱為「ObjectDescrRemove」?
是
6390
建立ObjectDescrRemove元件

圖63

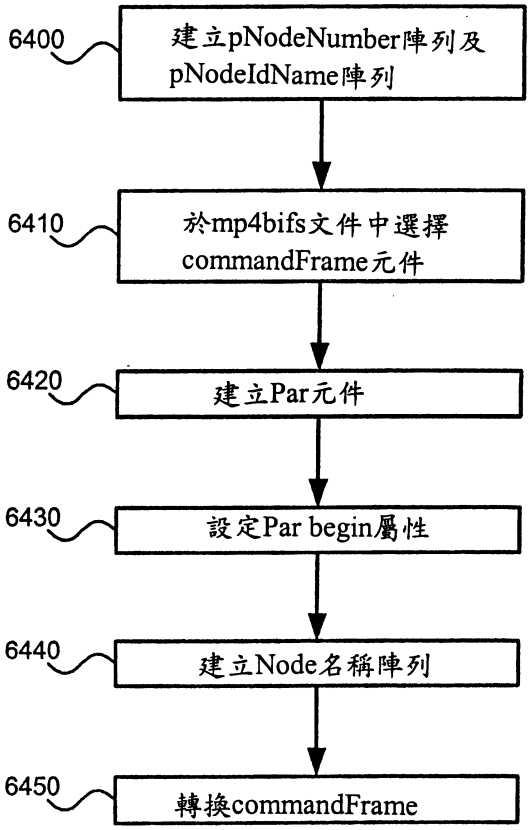

6400
建立pNodeNumber陣列及pNodeIdName陣列
6410
於mp4bifs文件中選擇commandFrame元件
6420
建立Par元件
6430
設定Par begin屬性
6440
建立Node名稱陣列
6450
轉換commandFrame

圖64

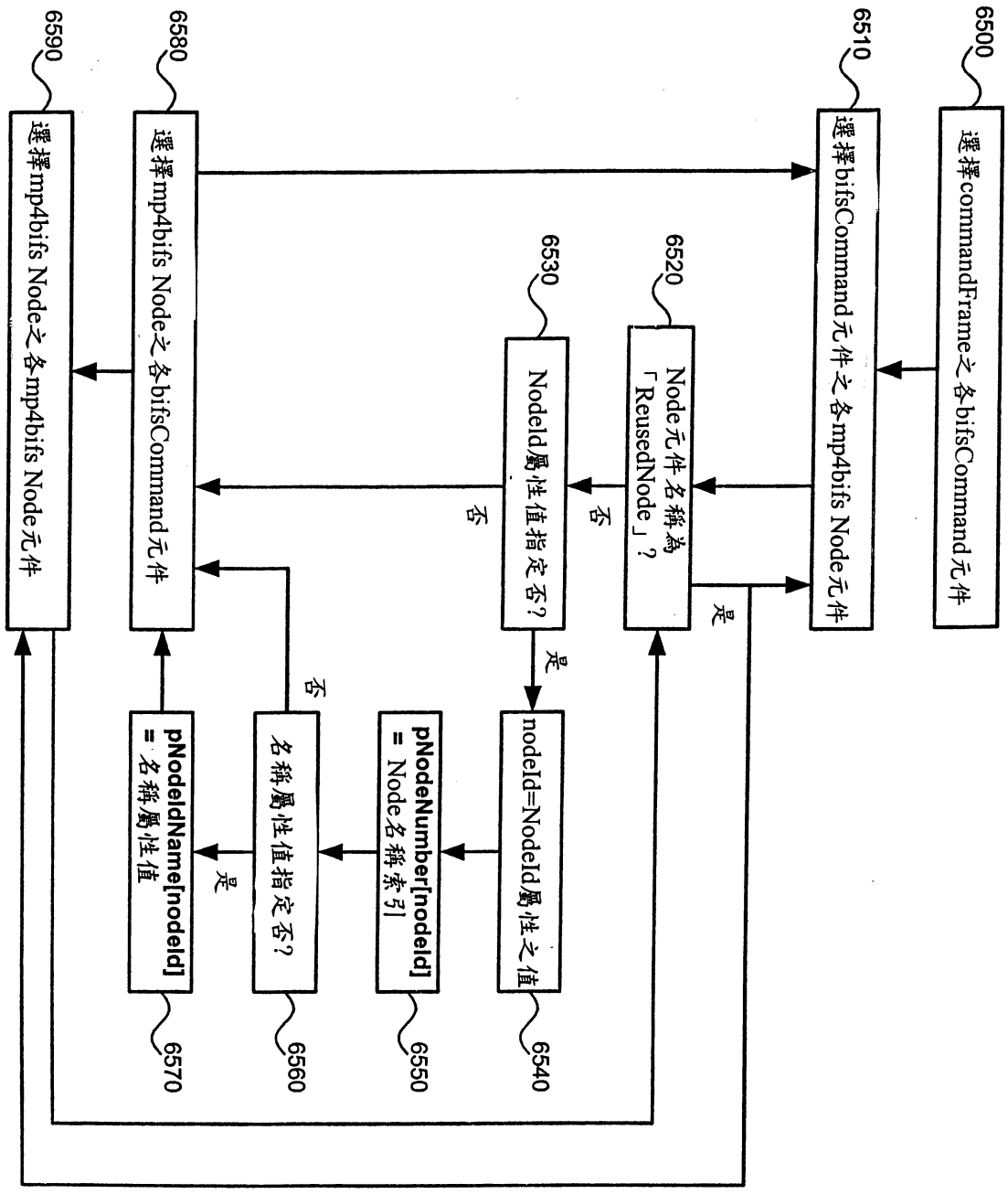

6500
選擇commandFrame之各bifsCommand元件
6510
選擇bifsCommand元件之各mp4bifs Node元件
6520
Node元件名稱為「ReusedNode」?
是
否
6530
NodeId屬性值指定否?
是
否
6540
nodeId=NodeId屬性之值
6550
pNodeNumber[nodeId] = Node名稱索引
6560
名稱屬性值指定否?
是
否
6570
pNodeIdName[nodeId] = 名稱屬性值
6580
選擇mp4bifs Node之各bifsCommand元件
6590
選擇mp4bifs Node之各mp4bifs Node元件

圖65

6. 操作3735，若數量「atomId」之值對應於「iods」，則執行程序「處理iods原子」。然後，以下述操作3770繼續。

7. 操作3740，比較數量「atomId」之值與「trak」值。

8. 操作3745，若數量「atomId」之值對應於「trak」，則執行程序「處理trak原子」。然後，以下述操作3770繼續。

9. 否則操作3760，跨越mp4檔案中的atomSize-8位元組。即，自該mp4檔案中讀取該位元組數，但不用所得值進行任何操作。

10. 操作3770，檢查到達moov原子754末端之情況。自mp4檔案從moov原子754之起端，讀取的資料位元組數等於操作3710中所決定的數量atomSize的值時，即滿足到達moov原子754之末端的條件。

11. 若已到達moov原子(754)之末端，則操作3780完成該過程。否則，重複操作3710至操作3770。

程序「處理使用者資料原子」可用於代表UserData結構3280中可選udta原子784的內容。該資訊並非最終XMT-A檔案所必需，因此，其足以僅跨越任何udta原子784的剩餘部分(atomsize-8位元組)。

或者，可用下列步驟解譯udat原子784的內容：

1. 建立一新UserData資料結構3280。

2. 將該新UserData資料結構3280的位址指派給QtInfo結構3100中的成員UserData 3123。

3. 將數量「atomSize」之值指派給該新UserData資料結構3280中的成員「size」3282。

array」3276。

7. 將零值指派給數量「entry」。

8. 重複以下步驟(8至10)numEntries次(直至數量「entry」之值等於數量「numEntries」之值)：

9. 自mp4檔案中讀取一32位元整數，並將結果指派給目前Mpod資料結構3270中陣列「trackID array」3276的項目「entry」。

10. 將數量「entry」之值遞增1。

程序「處理媒體標頭原子」由以下步驟組成：

1. 建立一新MediaHeader結構3300。

2. 將該新的MediaHeader結構3300的位址，指派給目前TrackMedia結構3226中的成員「MediaHeader」3230。

3. 將數量「atomSize」之值指派給TrackHeader結構3300中的成員「size」3303。

4. 自mp4檔案中讀取一32位元整數，並將結果指派給MediaHeader結構3300中的成員「version & Flags」3306。

5. 自mp4檔案中讀取一32位元整數，並將結果指派給MediaHeader結構3300的成員「creationTime」3308。

6. 自mp4檔案中讀取一32位元整數，並將結果指派給MediaHeader結構3300的成員「modifiedTime」3310。

7. 自mp4檔案中讀取一32位元整數，並將結果指派給MediaHeader結構3300的成員「timeScale」3313。

8. 自mp4檔案中讀取一32位元整數，並將結果指派給MediaHeader結構3300的成員「duration」3316。

9. 自mp4檔案中讀取一16位元整數，並將結果指派給MediaHeader結構3300的成員「language」3318。

10. 讀取mp4檔案中的2位元組，並忽略所得值。

程序「處理處置器原子」由以下步驟組成：

1. 建立一新Handler結構3320。

2. 將該新Handler結構3320之位址指派給目前TrackMedia結構3226內成員「Handler」3232。

3. 將數量「atomSize」之值指派給Handler結構3320內成員「size」3323。

4. 自mp4檔案讀取一32位元整數，並將結果指派給Handler結構3300內成員「version & Flags」3326。

5. 自mp4檔案讀取一32位元整數，並將結果指派給Handler結構3300內成員「handlerType」3328。

6. 讀取mp4檔案內12位元組，並忽略所得值。

7. 自mp4檔案讀取空終止字元串，將所得字元序列儲存於字元陣列內，並將該字元陣列之位址指派給Handler結構3320內成員「name」3330。

如圖40所示，程序「處理媒體資訊原子」由以下步驟組成：

1. 操作4000，建立新MediaInfo資料結構3333。將該新MediaInfo資料結構3333之位址指派給目前TrackMedia資料結構3226內成員MediaInfo 3234。將數量「atomSize」值指派給MediaInfo資料結構3333內成員「size」3336。

2. 操作4010，自mp4檔案讀取一32位元整數，並將該整

程序「處理資料參考表原子」由以下步驟組成：

1. 建立新DataRefTable資料結構3360。

2. 將該新DataRefTable資料結構3360之位址指派給目前MediaDataInfo資料結構3350內成員DataRefTable 3358。

3. 將數量「atomSize」值指派給DataRefTable資料結構3360之成員「size」3363。

4. 自mp4檔案讀取一32位元整數並將結果指派給DataRefTable結構3360內成員「version & Flags」3366。

5. 自mp4檔案讀取一32位元整數並將該值指派給數量「numDataRefs」。

6. 將數量「numDataRefs」值指派給DataRefTable結構3360內成員「numDataRefs」3368.

7. 建立具有「numDataRefs」項目之位址陣列，並將該陣列位址指派給新DataRefTable資料結構3360之成員「dataRef array」3370。

8. 將零值指派給數量「entry」。

9. 重複以下步驟numDataRefs次(直至數量「entry」值等於數量「numDataRefs」之值)：

10. 建立新dataRef資料結構3380。

11. 將該新dataRef資料結構3380之位址指派給DataRefTable資料結構3360之dataRef陣列成員3370內的項目「entry」。

12. 自mp4檔案讀取一32位元整數並將該值指派給數量「drefSize」。

13. 將數量「drefSize」值指派給該新dataRef資料結構3380之成員「size」3383。

14. 自mp4檔案讀取一32位元整數並將結果指派給目前dataRef資料結構3380內成員「version & Flags」3386。

15. 自mp4檔案讀取一32位元整數並將結果指派給該新dataRef資料結構3380內成員「dataRefType」3388。數量「type」值可解譯為四個本文字元之序列。

16. 建立(drefSize-12)位元組陣列並將該陣列位址指派給目前DataRef資料結構3380內成員「dataRefData」3390。

17. 自mp4檔案讀取(drefSize-12)位元組序列並將其複製到陣列「dataRefData」3390內連續項目。

18. 將數量「entry」值遞增1。

程序「處理動畫專家群第四階媒體標頭原子」由以下步驟組成：

1. 建立新Mpeg4MediaHeader資料結構。Mpeg4 MediaHeader資料結構包含成員「size」及「version & Flags」。

2. 將該新Mpeg4MediaHeader資料結構之位址指派給目前MediaInfo資料結構3333內成員MediaInfoHeader 3343。

3. 將數量「atomSize」值指派給該新Mpeg4Media Header資料結構之成員「size」。

4. 自mp4檔案讀取一32位元整數並將結果指派給該新Mpeg4MediaHeader資料結構內成員「verFlags」。

5. 跨越動畫專家群第四階媒體標頭(nmhd)原子936內包含的任何剩餘資料(若atomSize大於12)。

5. 自mp4檔案讀取一16位元整數並將結果指派給新VideoMediaHeader結構內成員「mode」。

6. 自mp4檔案讀取一16位元整數並將結果指派給目前VideoMediaHeader結構內成員「opRed」。

7. 自mp4檔案讀取一16位元整數並將結果指派給目前VideoMediaHeader結構內成員「opGreen」。

8. 自mp4檔案讀取一16位元整數並將結果指派給目前VideoMediaHeader結構內成員「opBlue」。

9. 跨越視頻媒體標頭(vmhd)原子936內包含的任何剩餘資料(若atomSize大於20)。

如圖41所示，程序「處理樣本表原子」由以下步驟組成：

1. 操作4100，建立新SampleTables資料結構3400。將該新SampleTables資料結構3400之位址指派給目前MediaInfo資料結構3333內成員SampleTables 3340。將「atomSize」值指派給新SampleTables資料結構3400之成員「size」3403。

2. 操作4110，自mp4檔案讀取一32位元整數並將該整數值指派給數量「atomSize」。接著自mp4檔案讀取第二32位元整數並將該整數值指派給數量「atomId」。

3. 操作4120，比較數量「atomId」值與「stsc」值。

4. 操作4125，若數量「atomId」值對應「stsc」則執行程序「處理sample to chunk table (stsc)原子」。然後以操作4190繼續。

5. 操作4130，比較數量「atomId」值與「stco」值。

6. 操作4135，若數量「atomId」值對應「stco」則執行

組成：

1. 建立新樣本表資料結構3420。

2. 將該新樣本表資料結構3420之位址指派給目前SampleTables資料結構3400內成員SampleToChunkTable 3406。

3. 將數量「atomSize」值指派給新樣本表資料結構3420之成員「size」3423。

4. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「version & Flags」3426。

5. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「numEntries」3428。

6. 建立具有numEntries項目之位址陣列，並將該陣列位址指派給新樣本表資料結構3420之成員「table array」3430。

7. 將零值指派給數量「entry」。

8. 重複以下步驟numEntries次(直至數量「entry」值等於成員「numEntries」3428值)。

9. 建立新SampleToChunk資料結構3460。

10. 將該新SampleToChunk資料結構3460之位址指派給新樣本表資料結構3420之成員「table array」3430內的項目「entry」。

11. 自mp4檔案讀取一32位元整數並將結果指派給新SampleToChunk資料結構3460內成員「firstChunk」3463。

12. 自mp4檔案讀取一32位元整數並將結果指派給新SampleToChunk資料結構3460內成員「numSamples」3466。

13. 自mp4檔案讀取一32位元整數並將結果指派給SampleToChunk資料結構3460內成員「description」3468。

14. 將數量「entry」值遞增1。

程序「處理大塊偏移表(stco)原子」由以下步驟組成：

1. 建立新樣本表資料結構3420。

2. 將該新樣本表資料結構3420之位址指派給目前SampleTables資料結構3400內成員ChunkOffsetTable 3408。

3. 將數量「atomSize」值指派給新樣本表資料結構3420之成員「size」3423。

4. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「version & Flags」3426。

5. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內成員「numEntries」3428。

6. 建立具有numEntries項目之整數陣列，並將該陣列位址指派給新樣本表資料結構3420之成員「table array」3430。

7. 將零值指派給數量「entry」。

8. 重複以下步驟numEntries次(直至數量「entry」值等於成員「numEntries」3428值)。

9. 自mp4檔案讀取一32位元整數並將結果指派給新樣本表資料結構3420內陣列「table array」3430的項目「entry」。

10. 將數量「entry」值遞增1。

程序「處理time to sample table (stts)原子」由以下步驟組成：

1. 建立新樣本表資料結構3420。

成員「dataFormat」值可代表以下4字元序列之一：「mp4a」、「mp4v」、「mp4s」。

13. 自mp4檔案讀取6位元組並忽略其值。

14. 自mp4檔案讀取一16位元整數並將結果指派給SampleDescription資料結構3480內成員「dataReference」3488。

15. 若數量「dataFormat」值對應「mp4a」串，自mp4檔案讀取20位元組並忽略其值。

16. 若數量「dataFormat」值對應「mp4v」串，自mp4檔案讀取70位元組並忽略其值。

17. 自mp4檔案讀取一32位元整數並將結果指派給數量「atomSize」。

18. 從mp4檔案讀取一32位元整數並將該值指派給數量「atomId」。

19. 若數量「atomId」值對應「esds」則執行程序「處理esds原子」。否則自mp4檔案讀取(atomSize-8)位元組並忽略其值。

程序「處理esds原子」由以下步驟組成：

1. 自mp4檔案讀取一32位元整數並將結果指派給SampleDescription資料結構3480內成員「esdVersion」3490。

2. 自數量「atomSize」值減去12並將結果指派給目前SampleDescription資料結構3480內成員「esdSize」3493。

3. 建立具有esdSize 3493項目之位元組的新陣列，並將該陣列之位址指派給目前SampleDescription資料結構3480內

成員「esdData」3496。

4. 自mp4檔案讀取esdSize 3496位元組序列並將其值複製到陣列「esdData」3496內的連續項目。

根據QtInfo資料結構建立中間xml文件及媒體資料檔案

根據mp4二進制檔案700、2210建立QtInfo資料結構3100後，用QtInfo資料結構3100建立一組媒體資料檔案2230及一對中間xml文件2250、2260。此等xml文件包括一「mp4file」文件2250、2300及一「mp4bifs」文件2260、2800。mp4file文件2250、2300之結構在圖23至27內表示。mp4bifs文件2260、2800之結構在圖28至30內表示。

該處理過程分成兩個步驟：

1. 形成在QtInfo資料結構3100內3113、3160代表之媒體資料(mdat)原子706的xml代表2310，以及在xml文件或二進制媒體資料檔案2230內儲存相關之媒體資料3168。

2. 形成QtInfo資料結構3100內代表之moov原子712的xml代表2320。

以下說明每個該等步驟。

步驟1：形成QtInfo資料結構內代表之媒體資料原子的xml代表以及在xml文件或二進制媒體資料檔案內儲存相關之媒體資料。

該步驟中，QtInfo資料結構3100內代表之每個mdat原子706由mp4file文件2300內mdat元件2310代表。與聲頻或視覺資料流(若有)相關之媒體資料3168傳送至新二進制媒體資料檔案2230。與場景描述資料流(sdsm或BIFS)相關之媒體資

4. 操作4430，將零值指派給數量「esdPos」。

5. 操作4440，執行程序「找出解碼器組態描述符」。

6. 操作4450，執行程序「得到標籤大小」。

7. 操作4460，將陣列「esdData」內項目「esdPos」值指派給數量「objectType」。

8. 操作4470，將數量「esdPos」值遞增一。

9. 操作4480，獲得陣列「esdData」內項目「esdPos」值，將結果向右移動兩個位元(除以4)並將結果指派給數量「streamType」。

若步驟5至9中數量「esdPos」值等於或超過數量「esdSize」值，將零值指派給數量「streamType」及「objectType」並終止該處理過程。

程序「找出解碼器組態描述符」

程序「找出解碼器組態描述符」如圖45所示由以下步驟組成：

1. 操作4500，將陣列「esdData」內項目「esdPos」值指派給數量「esdTag」。

2. 操作4510，比較數量「esdTag」值與「3」值。

3. 操作4520，若數量「esdTag」值非「3」，將數量「esdSize」值指派給數量「esdPos」並結束該程序4525。否則執行以下步驟。

4. 操作4530，將數量「esdPos」值遞增一。

5. 操作4535，執行程序「得到標籤大小」。

6. 操作4540，將數量「esdPos」值遞增「4」。

7. 操作4550，將陣列「esdData」內項目「esdPos」值指派給數量「esdTag」。

8. 操作4560，比較數量「esdTag」值與「4」值。

9. 操作4565，若數量「esdTag」值非「4」，將數量「esdPos」值遞增一，執行程序「得到標籤大小」4570，將數量「tagSize」值加入數量「esdPos」4575值，然後重複操作4550。

10. 否則操作4580，將數量「esdPos」值遞增一。

11. 操作4585，完成該程序並繼續程序「找出資料流類型」。

程序「得到標籤大小」

程序「得到標籤大小」由以下步驟組成：

1. 操作4600，將零值指派給數量「tagSize」。

2. 操作4610，將陣列「esdData」內項目「esdPos」值指派給數量「sizeByte」。

3. 操作4620，比較數量「sizeByte」值與值128。若數量「sizeByte」值大於或等於128，執行以下操作：

4. 操作4630，從數量「sizeByte」值減去128(或者以該值之較低7位元替換數量「sizeByte」值)。

5. 操作4640，將數量「tagSize」值乘以128(或者將該值向左移動7位元)。

6. 操作4650，將數量「sizeByte」值加入數量「tagSize」值。

7. 操作4660，將數量「esdPos」值遞增「1」值。

8. 重複操作4610。若數量「sizeByte」值小於128，執行

以下操作：

9. 操作4670，將數量「tagSize」值乘以128(或者向左移動7位元)。

10. 操作4680，將數量「sizeByte」值加入數量「tagSize」值。

11. 操作4690，將數量「esdPos」值遞增「1」值。

12. 操作4695，完成該程序並繼續調用該程序之程序。

程序「OdsmDataToXml」

程序「OdsmDataToXml」4355建立包含於mdat結構「mdatAtom」4230內的odsm資料流(物件描述符資料流)之xml代表。如圖47所示，以下操作用於建立該odsm資料流之xml代表：

1. 操作4700將mdat結構「mdatAtom」4230之成員「MediaData」3168值指派給代表位元組陣列位址的數量「OdsmData」。mdat結構「mdatAtom」4230之成員「size」3163值指派給整數數量「OdsmSize」。

2. 操作4705建立新「Odsm」元件2460並附加至目前mdat元件2400、4260。包含於trak結構3200「trakAtom」4320內之TrackHeader結構3212、3204的成員trackID3224值指派給新「odsm」元件之屬性「trackID」值。

3. 操作4710將樣本表結構3420ChunkOffsetTable 3408之成員numEntries 3428值指派給數量「numChunks」。樣本表結構3420 SampleToChunkTable 3406之成員numEntries 3428值指派給數量「numStscEntries」。樣本表結構

4890。

18. 操作4895，其中數量「sampleSize」值遞減數量「tagSize」值(如操作4865內所決定)。

19. 以操作4845繼續處理。

程序「decode OdUpdate」

若在操作4860用於odsm命令之命令標籤值等於「ObjectDescrUpdateTag」值(操作4870、2000)，執行以下步驟4875，如圖49所示：

1. 操作4900建立新ObjectDescrUpdate元件2540並將其附加至目前odsmSample元件2520。

2. 操作4910使用程序「得到標籤大小」獲得數量「tagSize」值(numBytes 2020)。此與上述程序「得到標籤大小」等效，除了(a)位元組OdsmData陣列替代「esdData」陣列使用，以及(b)數量「offset」替代數量「esdPos」使用。將數量「tagSize」值指派給數量「residualSize」。

3. 操作4920比較數量「residualSize」值與「1」值。

4. 在操作4925，若數量「residualSize」值小於1，該程序完成。

5. 在操作4930，若數量「residualSize」值不小於1，將位元組「OdsmData」陣列內項目「offset」值指派給數量「odTag」。將數量「offset」值遞增「1」並將數量「residualSize」值遞減「1」。

6. 操作4940使用程序「得到標籤大小」獲得數量「tagSize」值(numBytes 2116)。此與上述程序「得到標籤

d. 若該數量「nodeId」之值為非空值且此數量所代表的該字串至少具有一字元，則將該數量「nodeId」的值附加至該前置「nodeId:」，且將該結果指派給新XMT-A Node元件400之「DEF」屬性。

4. 使用標準xml裝置以選擇目前mp4bifs Node元件3000所指定的各性質欄位屬性值3020。

5. 將目前屬性值的名稱之值指派給該字串數量「attributeName」。

6. 對mp4bifs Node元件3000之各性質欄位屬性3020執行下列步驟。

7. 決定欄位資料類型：

a. 決定該整數數量「field」之值。此值係等於一欄位名稱表中與「attributeName」之值匹配的該成員之項目數的值。該欄位名稱表係藉由動畫專家群第四階規格說明定義，其中該BIFS節點具有與目前mp4bifs Node元件相關的節點名稱及節點數。

b. 將一fieldDataType表中之項目「field」的值指派給該整數數量「fieldDataType」。該fieldDataType表值係藉由動畫專家群第四階規格說明定義，其中該BIFS節點具有與目前mp4bifs Node元件3000相關的節點名稱及節點數。

8. 檢查URL欄位資料類型。若該欄位資料類型等於動畫專家群第四階規格說明中定義的代表「URL」欄位資料類型之一值，則執行下列操作：

a. 將mp4bifs Node元件之「attributeName」屬性的值指派

給該字串數量「urlString」。

b. 將該數量「urlString」所代表的字元串數量之前五字元替換為前置「odid://ODID:」。

c. 將所得字串值指派給目前XMT-A Node元件400之「attributeName」屬性。

9. 檢查Node欄位資料類型。若該欄位資料類型等於動畫專家群第四階規格說明中定義的代表「Node」欄位資料類型之一值，則執行下列操作：

a. 建立元件名稱為「attributeName」之一新XMT-A元件且將其附加至目前XMT-A Node元件。此元件係定義為「目前XMT-A nodeField元件」。

b. 使用標準xml裝置以選擇目前mp4bifs Node元件所擁有的各附屬元件。各附屬元件預期係一另一mp4bifs Node元件。該等附屬元件係指定為「子」元件，而目前mp4bifs Node元件係指定為「母」元件。

c. 利用目前XMT-A nodeField元件為「母」元件，對各子元件執行程序「將一mp4bifs Node元件轉換成一XMT-A Node元件」。

10. 檢查Command Buffer欄位資料類型。若該欄位資料類型等於動畫專家群第四階規格說明中定義的代表「Command Buffer」欄位資料類型之一值，則執行下列操作：

a. 建立元件名稱為「attributeName」之一新XMT-A元件且將其附加至目前XMT-A Node元件。此元件係定義為「目

前XMT-A nodeField元件」400。

b. 利用目前XMT-A nodeField元件為「母」元件，執行程序「將mp4bifs commandFrame元件轉換成XMT-A元件」。

11. 處理其他屬性。若該欄位資料類型非前述步驟中識別的值之一，則執行下列操作：

a. 將mp4bifs Node元件之「attributeName」屬性的值指派給該字串數量「attributeValue」。

b. 將該數量「attributeValue」的值指派給目前XMT-A Node元件400之「attributeName」屬性。

為一mp4bifs ReusedNode元件建立XMT-ANode元件

在一mp4bifs ReusedNode元件3080情形中，一nodeRef值3090屬性係所需的屬性。對各mp4bifs ReusedNode元件3080執行下列步驟：

1. 將nodeRef屬性3090的值指派給該整數數量「nodeId」。

2. 將陣列pNodeNumber中之項目「nodeId」的值指派給該整數數量「nodeNumber」。

3. 將動畫專家群第四階規格說明中定義之一節點名稱表中的項目「nodeNumber」之值指派給該字串數量「nodeName」。

4. 建立「nodeName」類型之一新XMT-A節點元件400且將其附加至目前XMT-A母元件。

5. 將該數量nodeId的值指派給新XMT-A節點元件400之「USE」屬性。

基於圖解及說明的目的，前面已提出本發明之說明。該

拾、申請專利範圍：

1. 一種用於將一動畫專家群第四階(mp4)二進制檔案轉換成一可擴充動畫專家群第四階本文(XMT)檔案之方法，該方法包括：

   產生至少一代表該mp4二進制檔案的中間結構文件；

   將該至少一中間結構文件翻譯為一XMT結構文件；及

   根據該XMT結構文件，建立該XMT檔案。

2. 如申請專利範圍第1項之方法，其中該中間結構文件包括代表一moov原子與一或多個mdat原子的資料結構，該moov原子包括一iods原子與一或多個trak原子。

3. 如申請專利範圍第2項之方法，進一步包括根據代表該等mdat原子與trak原子的該等資料結構，產生外部媒體檔案。

4. 如申請專利範圍第2項之方法，其中翻譯該中間結構文件包括產生一mp4-檔案結構文件。

5. 如申請專利範圍第4項之方法，其中該mp4-檔案結構文件係基於可擴充標示語言。

6. 如申請專利範圍第4項之方法，其中該mp4-檔案結構文件包括一moov結構元件及一或多個mdat結構元件，該moov結構元件代表該moov原子，各mdat結構元件代表該中間結構文件所代表的該等mdat原子之一。

7. 如申請專利範圍第6項之方法，其中該moov結構元件包括一iods元件及一或多個trak元件，該等trak元件代表該中間結構文件的對應部分。

8. 如申請專利範圍第1項之方法，其中翻譯該中間結構文件包括產生一mp4-bifs結構文件，其代表一場景描述資料流。

9. 如申請專利範圍第4項之方法，其中該mp4-檔案結構文件包括代表一物件描述符資料流的元件。

10. 如申請專利範圍第4項之方法，其中該mp4-檔案結構文件包括代表一場景描述資料流的元件。

11. 如申請專利範圍第1項之方法，其中產生該中間結構文件包括：

產生一組代表該mp4二進制檔案的資料結構；及

將該組資料結構翻譯為該中間結構文件。

12. 如申請專利範圍第8項之方法，其中該mp4-bifs結構文件係一與該mp4-檔案結構文件分離之不同文件。

13. 一種用於將一動畫專家群第四階(mp4)二進制檔案轉換成一可擴充動畫專家群第四階本文(XMT)檔案之系統，該系統包括：

一二進制檔案解碼器，其配置成用以輸入該mp4二進制檔案，並產生至少一代表該mp4二進制檔案的中間結構文件；

一翻譯器，其配置成用以輸入該中間結構文件，並產生一XMT結構文件；及

一XMT串聯器，其配置成用以輸入該XMT結構文件，並產生該XMT檔案。

14. 如申請專利範圍第13項之系統，其中該至少一中間結構

文件包括代表一moov原子與一或多個mdat原子的資料結構。

15. 如申請專利範圍第14項之系統，其中代表一moov原子的該資料結構包括代表一iods原子與一或多個trak原子的資料結構。

16. 如申請專利範圍第15項之系統，進一步包括一或多個外部媒體檔案，其基於代表該等mdat原子與trak原子的該等資料結構。

17. 如申請專利範圍第14項之系統，其中該至少一中間結構文件包括一mp4-檔案結構文件，該mp4-檔案結構文件包括基於該組資料結構的一或多個mdat元件、一moov元件及一iods元件。

18. 如申請專利範圍第17項之系統，其中該mp4-檔案結構文件代表一物件描述符資料流。

19. 如申請專利範圍第17項之系統，進一步包括一代表一場景描述資料流的mp4-bifs結構文件。

20. 如申請專利範圍第19項之系統，其中該mp4-bifs結構文件係一與該mp4-檔案結構文件分離之不同的結構文件。

21. 一種具體化於一有形媒體中之電腦程式產品，其包括：

與該有形媒體耦合之電腦可讀取程式碼，用於將一動畫專家群第四階(mp4)二進制檔案轉換成一可擴充動畫專家群第四階本文(XMT)檔案，該電腦可讀取程式碼配置成使該程式可用於：

產生至少一代表該mp4二進制檔案的中間結構文件；

將該中間結構文件翻譯為一XMT結構文件；及

根據該XMT結構文件，建立該XMT檔案。

22. 如申請專利範圍第21項之電腦程式產品，其中該至少一中間結構文件包括代表一moov原子與一或多個mdat原子的資料結構，該moov原子包括一或多個trak原子。

23. 如申請專利範圍第22項之電腦程式產品，其中該電腦可讀取程式碼係進一步配置成用以根據代表該等mdat原子與該等trak原子的該等資料結構，產生外部媒體檔案。

24. 如申請專利範圍第21項之電腦程式產品，其中該至少一中間結構文件包括一mp4-檔案結構文件。

25. 如申請專利範圍第24項之電腦程式產品，其中該mp4-檔案結構文件包括基於該至少一中間結構文件的一或多個mdat元件、一moov元件及一iods元件。

26. 如申請專利範圍第25項之電腦程式產品，其中該mp4-檔案結構文件代表一物件描述符資料流。

27. 如申請專利範圍第25項之電腦程式產品，其中該mp4-檔案結構文件代表一場景描述資料流。

28. 如申請專利範圍第21項之電腦程式產品，其中該至少一中間結構文件包括一代表一場景描述資料流的mp4-bifs結構文件。

29. 如申請專利範圍第28項之電腦程式產品，其中該mp4-bifs結構文件係一與該mp4-檔案結構文件分離之不同文件。

30. 如申請專利範圍第21項之電腦程式產品，其中該電腦可讀取程式碼係進一步配置成用以：

產生一組代表該mp4-二進制檔案的資料結構；及

將該組資料結構翻譯為該至少一中間結構文件。

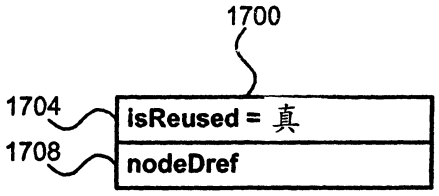


圖17A

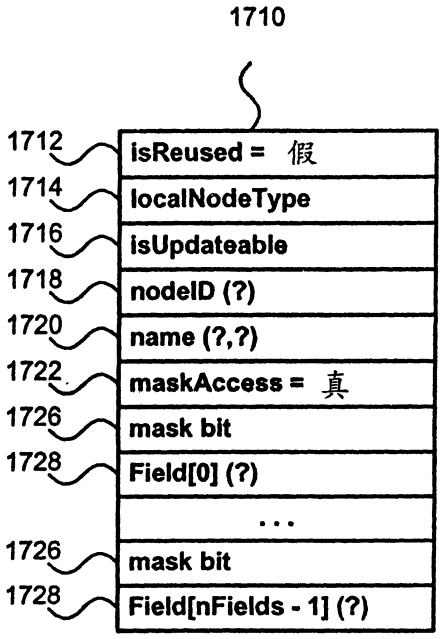


圖17B

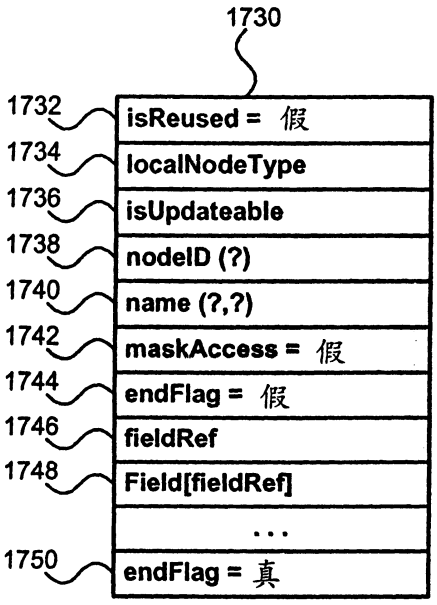


圖17C

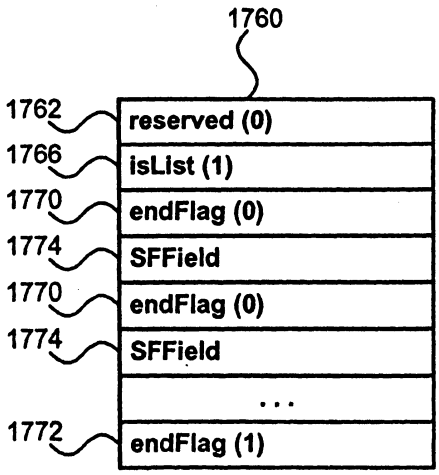


圖17D

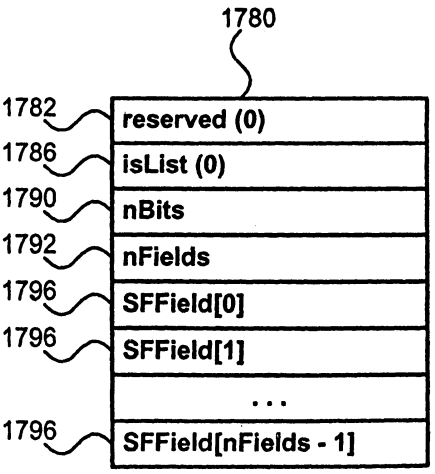


圖17E

第092130400號專利申請案
中文圖式替換頁(93年12月)

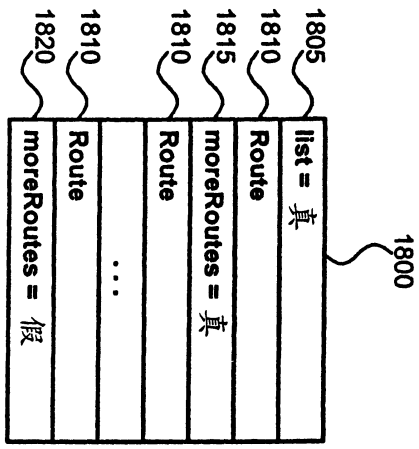


圖18A

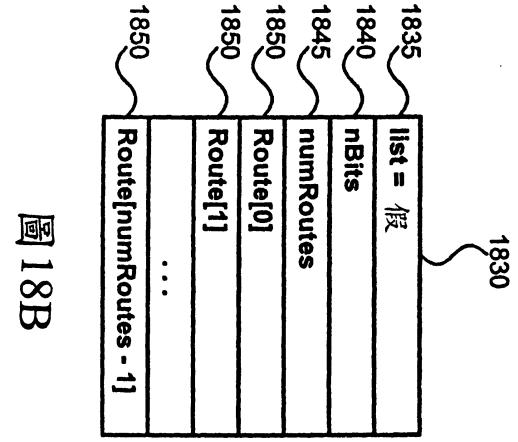


圖18B

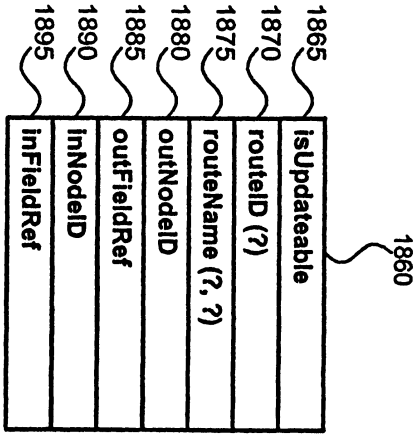


圖18C

第092130400號專利申請案
中文圖式替換頁(93年12月)

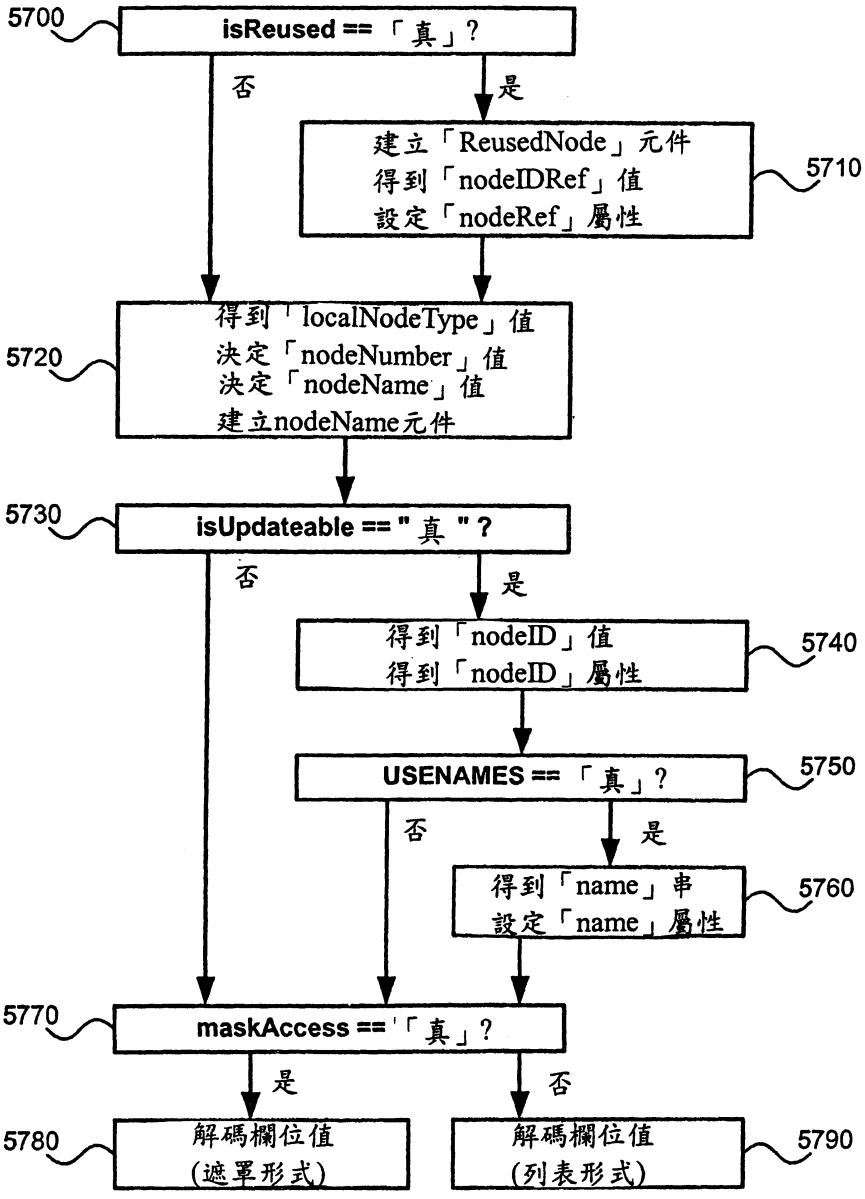


圖57