Appendix E: RealMedia File Format (RMFF) Reference

The Helix architecture supports RealMedia File Format (RMFF), which enables Helix to deliver high- quality multimedia content over a variety of network bandwidths. Third-party developers can convert their media formats into RMFF, enabling Helix Universal Server to deliver the files to RealPlayer or other applications built with the Helix Client and Server Software Development Kit. Third-party developers can thereby use Helix to transport content over the Internet to their own applications.

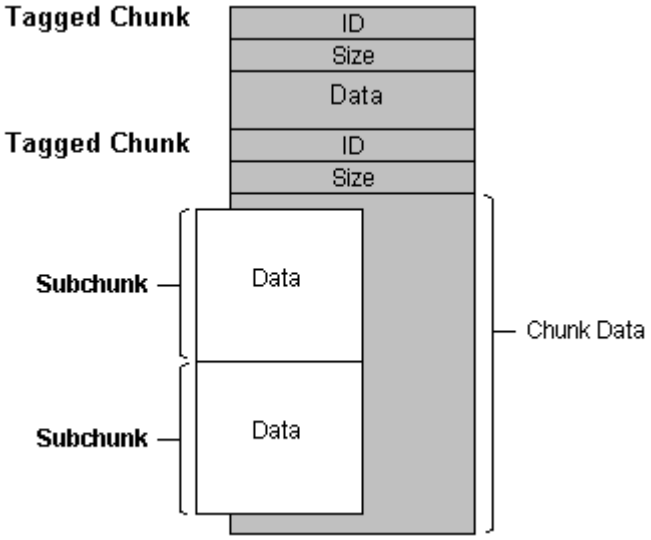
RealMedia File Format is a standard tagged file format that uses four-character codes to identify file elements. These codes are 32-bit, represented by a sequence of one to four ASCII alphanumeric characters, padded on the right with space characters. The data type for four-character codes is FOURCC. Use the `HX_FOURCC` macro to convert four characters into a four-character code.

The basic building block of a RealMedia File is a *chunk* , which is a logical unit of data, such as a stream header or a packet of data. Each chunk contains the following fields:

- Four-character code specifying the chunk identifier
- 32-bit value specifying the size of the data member in the chunk
- Blob of opaque chunk data

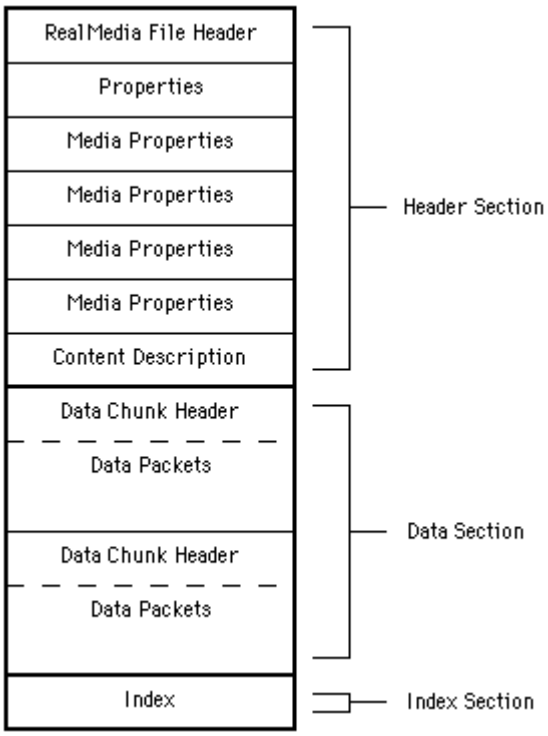
Depending on its type, a top-level chunk can contain subobjects. This document describes the tagged chunks contained in RMFF, as well as the format of the data stored in each type of tagged chunk.

Tagged File Formats



RealMedia File Format organizes tagged chunks into a header section, a data section, and an index section. The organization of these tagged chunks is shown in the following figure.

Sections of a RealMedia File



Header Section

Because RMFF is a tagged file format, the order of the chunks is not explicit, except that the RealMedia File Header must be the first chunk in the file. However, most applications write the standard headers into the file's header section. The following chunks are typically found in the header

section of RMFF:

- RealMedia File Header (this must be the first chunk of the file)
- Properties Header
- Media Properties Header
- Content Description Header

After the RealMedia File Header object, the other headers may appear in any order. All headers are required except the Index Header. The following sections describe the individual header objects.

RealMedia File Header

Each RealMedia file begins with the RealMedia File Header, which identifies the file as RMFF. There is only one RealMedia File Header in a RealMedia file. Because the contents of the RealMedia File Header may change with different versions of RMFF, the header structure supports an object version field for determining what additional fields exists. The following pseudo-structure describes the RealMedia File Header:

```
RealMedia_File_Header
{
  UINT32    object_id;
  UINT32    size;
  UINT16    object_version
;
  if ((object_version == 0) || (object_version == 1))
  {
    UINT32   file_version;
    UINT32   num_headers;
  }
}
```

The RealMedia File Header contains the following fields:

object_id
: The unique object ID for a RealMedia File (`.RMF`). All RealMedia files begin with this identifier. The size of this member is 32 bits.

size
: The size of the RealMedia header section in bytes. The size of this member is 32 bits.

object_version
: The version of the RealMedia File Header object. All files created

according to this specification have an `object_version` number of 0 (zero) or 1. The size of this member is 16 bits.

file_version

The version of the RealMedia file. The Helix Client and Server SDK only covers files with a file version of either 0 (zero) or 1. This member is present on all `RealMedia_File_Header` objects with an `object_version` of 0 (zero) or 1. The size of this member is 32 bits.

num_headers

The number of headers in the header section that follow the RealMedia File Header. This member is present on all `RealMedia_File_Header` objects with an `object_version` of 0 (zero) or 1. The size of this member is 32 bits.

Properties Header

The Properties Header describes the general media properties of the RealMedia File. Components of the RealMedia system use this object to configure themselves for handling the data in the RealMedia file or stream. There is only one Properties Header in a RealMedia file. The following pseudo-structure describes the Properties header:

```
Properties
{
  UINT32    object_id;
  UINT32    size;
  UINT16    object_version;
  if (object_version == 0)
  {
    UINT32   max_bit_rate;
    UINT32   avg_bit_rate;
    UINT32   max_packet_size;
    UINT32   avg_packet_size;
    UINT32   num_packets;
    UINT32   duration;
    UINT32   preroll;
    UINT32   index_offset;
    UINT32   data_offset;
    UINT16   num_streams;
    UINT16   flags;
  }
}
```

The Properties Header contains the following fields:

object_id

The unique object ID for a Properties Header ('PROP'). The size of this member is 32 bits.

size

The 32-bit size of the Properties Header in bytes. The size of this member is 32 bits.

object_version

The version of the RealMedia File Header object. All files created according to this specification have an `object_version` number of 0 (zero). The size of this member is 16 bits.

max_bit_rate

The maximum bit rate required to deliver this file over a network. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

avg_bit_rate

The average bit rate required to deliver this file over a network. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

max_packet_size

The largest packet size (in bytes) in the media data. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

avg_packet_size

The average packet size (in bytes) in the media data. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

num_packets

The number of packets in the media data. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

duration

The duration of the file in milliseconds. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

preroll

The number of milliseconds to prebuffer before starting playback. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

index_offset

The offset in bytes from the start of the file to the start of the index header object. This value can be 0 (zero), which indicates that no index

chunks are present in this file. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

data_offset

The offset in bytes from the start of the file to the start of the Data Section. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

Note: There can be a number of Data_Chunk_Headers in a RealMedia file. The `data_offset` value specifies the offset in bytes to the first Data_Chunk_Header. The offsets to the other Data_Chunk_Headers can be derived from the `next_data_header` field in a Data_Chunk_Header.

num_streams

The total number of media properties headers in the main headers section. This member is present on all Properties objects with an `object_version` of 0 (zero). The size of this member is 16 bits.

flags

Bit mask containing information about this file. The following bits carry information—all of the rest should be zero:

Bit	Flag	Description
0	Save_Enabled	If 1, clients are allowed to save this file to disk.
1	Perfect_Play	If 1, clients are instructed to use extra buffering.
2	LIve	If 1, these streams are from a live broadcast.

The size of this member is 16 bits.

Media Properties Header

The Media Properties Header describes the specific media properties of each stream in a RealMedia file. Components of the RealMedia system use this object to configure themselves for handling the media data in each stream. There is one Media Properties Header for each media stream in a RealMedia file. The following pseudo-structure describes the Media Properties header:

```
Media_Properties
{
  UINT32     object_id;
```

```
  UINT32     size;
  UINT16     object_version;
  if (object_version == 0)
  {
    UINT16                      stream_number;
    UINT32                      max_bit_rate;
    UINT32                      avg_bit_rate;
    UINT32                      max_packet_size;
    UINT32                      avg_packet_size;
    UINT32                      start_time;
    UINT32                      preroll;
    UINT32                      duration;
    UINT8                       stream_name_size;
    UINT8[stream_name_size]     stream_name;
    UINT8                       mime_type_size;
    UINT8[mime_type_size]       mime_type;
    UINT32                      type_specific_len;
    UINT8[type_specific_len]    type_specific_data;
  }
}
```

The Media Properties Header contains the following members:

object_id

The unique object ID for a Media Properties Header ("MDPR"). The size of this member is 32 bits.

size

The size of the Media Properties Header in bytes. The size of this member is 32 bits.

object_version

The version of the Media Properties Header object. The size of this member is 16 bits.

stream_number

The `stream_number` (synchronization source identifier) is a unique value that identifies a physical stream. Every data packet that belongs to a physical stream contains the same `STREAM_NUMBER` . The `STREAM_NUMBER` enables a receiver of multiple physical streams to distinguish which packets belong to each physical stream. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

max_bit_rate

The maximum bit rate required to deliver this stream over a network. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

avg_bit_rate

The average bit rate required to deliver this stream over a network.

This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

max_packet_size

The largest packet size (in bytes) in the stream of media data. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

avg_packet_size

The average packet size (in bytes) in the stream of media data. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

start_time

The time offset in milliseconds to add to the time stamp of each packet in a physical stream. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

preroll

The time offset in milliseconds to subtract from the time stamp of each packet in a physical stream. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

duration

The duration of the stream in milliseconds. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

stream_name_size

The length of the following `stream_name` member in bytes. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 8 bits.

stream_name

A nonunique alias or name for the stream. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). This size of this member is variable.

mime_type_size

The length of the following `mime_type` field in bytes. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). This size of this member is 8 bits.

mime_type

A nonunique MIME style type/subtype string for data associated with the stream. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). This size of this member is variable.

type_specific_len

The length of the following `type_specific_data` in bytes. The `type_specific_data` is typically used by the data type renderer to initialize itself in order to process the physical stream. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

type_specific_data

The `type_specific_data` is typically used by the data type renderer to initialize itself in order to process the physical stream. This member is present on all MediaProperties objects with an `object_version` of 0 (zero). The size of this member is variable.

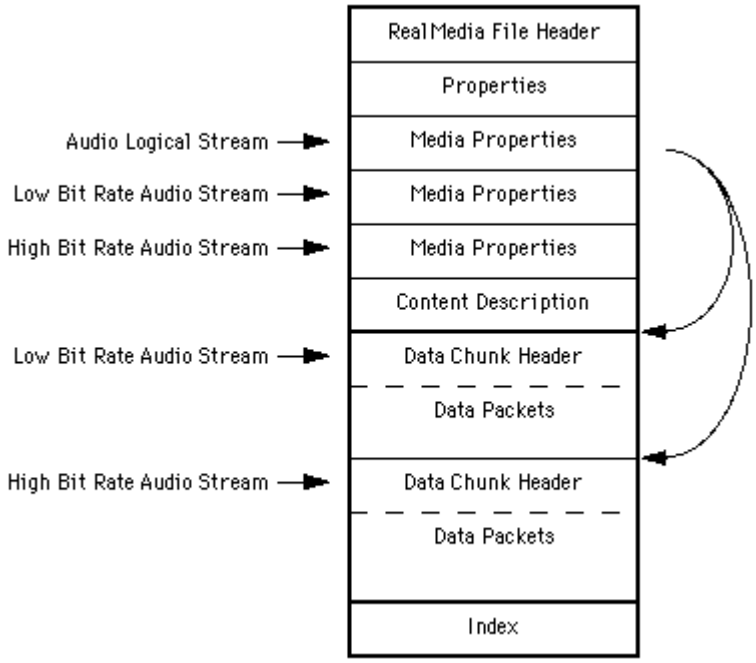
Logical Stream Organization

A RealMedia file can contain a higher-level grouping of physical streams. This grouping is called a logical stream. Logical streams contain the following information:

- Identifies which physical streams are grouped together into a logical stream.
- Contains name value properties that can be used to idnetify properties of the logical stream. (such as language, packet grouping, and so on.)

A logical stream is represented with a Media Properties Header. The mime type of the physical stream is preceeded with "`logical-` ". For example, the mime type for an ASM-compatible RealAudio stream is `audio/x-pn-multirate-realaudio` . A logical stream consisting of a set of RealAudio physical streams would therefore have the mime type `logical-audio/x-pn-multirate-realaudio` . An example of a logical stream is shown in the following figure.

Logical Stream Organization



In this example there is one logical stream, one low bit rate audio stream and one high bit rate audio stream. This results in a RealMedia file with three Media Property Headers and three data sections. The `type_specific_data` field of the logical stream's Media Property Header contains a `LogicalStream` structure. This structure contains all of the information required to interpret the logical stream and its collection of physical streams. The structure refers to the low bit rate and high bit rate audio streams. The `LogicalStream` structure also contains the `data_offset` s to the start of the data section for each physical stream.

The logical stream number assigned to the logical stream is determined from the `stream_number` field in the Media Properties Header.

There is also one special logical stream of MIME type "`logical-fileinfo` " containing information about the entire file. There should only be one media header with this type. Behavior of players and editing tools is undefined if you have more than one.

The ASM rules contained in the `logical-fileinfo` stream are used to define precisely how bandwidth will be divided between the streams in the file. The `logical-fileinfo` may also contain a name-value pair that specifies which stream combinations should be served to older players.

LogicalStream Structure

The following sample shows the `LogicalStream` structure:

```
LogicalStream
{
  ULONG32       size;
  UINT16        object_version;
  if (object_version == 0)
  {
    UINT16      num_physical_streams;
    UINT16      physical_stream_numbers[num_physical_streams];
    ULONG32     data_offsets[num_physical_streams];
    UINT16      num_rules;
    UINT16      rule_to_physical_stream_number_map[num_rules];
    UINT16      num_properties;
    NameValueProperty   properties[num_properties];
  }
};
```

The `LogicalStream` structure contains the following fields:

size

The size of the `LogicalStream` structure in bytes. The size of this structure member is 32 bits.

object_version

The version of the LogicalStream structure. The size of this structure member is 16 bits.

num_physical_streams

The number of physical streams that make up this logical stream. The physical stream numbers are stored in a list immediately following this field. These physical stream numbers refer to the `stream_number` field found in the Media Properties Object for each physical stream belonging to this logical stream. The size of this structure member is 16 bits

physical_stream_numbers[]

The list of physical stream numbers that comprise this logical stream. The size of this structure member is variable.

data_offsets[]

The list of data offsets indicating the start of the data section for each physical stream. The size of this structure member is variable.

num_rules

The number of ASM rules for the logical stream. Each physical stream in the logical stream has at least one ASM rule associated with it or it will never get played. The mapping of ASM rule numbers to physical stream numbers is stored in a list immediately following this member. These physical stream numbers refer to the `stream_number` field found in the Media Properties Object for each physical stream

belonging to this logical stream. The size of this structure member is 16 bits.

rule_to_physical_stream_map[]

The list of physical stream numbers that map to each rule. Each entry in the map corresponds to a 0-based rule number. The value in each entry is set to the physical stream number for the rule. For example:

```
rule_to_physical_stream_map[0] = 5
```

This example means physical stream 5 corresponds to rule 0. All of the ASM rules referenced by this array are stored in the first name-value pair of this logical stream which must be called "ASMRuleBook" and be of type "string". Each rule is separated by a semicolon.

The size of this structure member is variable.

num_properties

The number of `NameValueProperty` structures contained in this structure. These name/value structures can be used to identify properties of this logical stream (for example, language). The size of this structure member is 16 bits.

properties[]

The list of `NameValueProperty` structures (see NameValueProperty Structure below for more details). As mentioned above, it is required that the first name-value pair be a string named "ASMRuleBook" and contain the ASM rules for this logical stream. The size of this structure member is variable.

NameValueProperty Structure

The following sample shows the `NameValueProperty` structure:

```
NameValueProperty
{
  ULONG32               size;
  UINT16                object_version;
  if (object_version == 0)
  {
    UINT8       name_length;
    UINT8       name[namd_length];
    INT32       type;
    UINT16      value_length;
    UINT8       value_data[value_length];
  }
}
```

The `NameValueProperty` structure contains the following fields:

size

The size of the `NameValueProperty` structure in bytes. The size of this structure member is 32 bits.

object_version

The version of the `NameValueProperty` structure. The size of this structure member is 16 bits.

name_length

The length of the name data. The size of this structure member is 8 bits.

name[]

The name string data. The size of this structure member is 8 bits.

type

The type of the value data. This member can take on one of three values (any other value is undefined), as shown in the following table:

type	Description	value_length
0	32-bit unsigned integer property	4
1	buffer	variable
2	string	variable

The size of this structure member is 32 bits.

value_length

The length of the value data. The size of this structure member is 16 bits.

value_data[]

The value data. The size of this structure member is 8 bits.

Content Description Header

The Content Description Header contains the title, author, copyright, and comments information for the RealMedia file. All text data is in ASCII format. The following pseudo-structure describes the Content Description Header:

```
Content_Description
{
  UINT32     object_id;
  UINT32     size;
  UINT16      object_version
;
  if (object_version == 0)
```

```
  {
    UINT16    title_len;
    UINT8[title_len]  title;
    UINT16    author_len;
    UINT8[author_len]  author;
    UINT16    copyright_len;
    UINT8[copyright_len]  copyright;
    UINT16    comment_len;
    UINT8[comment_len]  comment;
  }
}
```

The Content Description Header contains the following fields:

object_id
: The unique object ID for the Content Description Header ("CONT"). The size of this member is 32 bits.

size
: The size of the Content Description Header in bytes. The size of this member is 32 bits.

object_version
: The version of the Content Description Header object. The size of this member is 16 bits.

title_len
: The length of the title data in bytes. Note that the title data is not null-terminated. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is 16 bits.

title
: An array of ASCII characters that represents the title information for the RealMedia file. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is variable.

author_len
: The length of the author data in bytes. Note that the author data is not null-terminated. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is 16 bits.

author
: An array of ASCII characters that represents the author information for the RealMedia file. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is variable.

copyright_len

The length of the copyright data in bytes. Note that the copyright data is not null-terminated. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is 16 bits

copyright

An array of ASCII characters that represents the copyright information for the RealMedia file. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is variable.

comment_len

The length of the comment data in bytes. Note that the comment data is not null-terminated. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is 16 bits.

comment

An array of ASCII characters that represents the comment information for the RealMedia file. This member is present on all Content Description Header objects with an `object_version` of 0 (zero). The size of this member is variable.

Data Section

The data section of the RealMedia file consists of a Data Section Header that describes the contents of the data section, followed by a series of interleaved media data packets. Note that the size field of the Data Chunk Header is the size of the entire data chunk, including the media data packets.

Data Chunk Header

The Data Chunk Header marks the start of the data chunk. There is usually only one data chunk in a RealMedia file; however, for extremely large files, there may be multiple data chunks. The following pseudostructure describes the Data chunk header:

```
Data_Chunk_Header
{
  UINT32     object_id;
  UINT32     size;
  UINT16      object_version;
  if (object_version == 0)
  {
```

```
    UINT32    num_packets;
    UINT32    next_data_header;
  }
}
```

The Data Chunk Header contains the following fields:

object_id
: The unique object ID for the Data Chunk Header ('DATA'). The size of this member is 32 bits.

size
: The size of the Data Chunk in bytes. The size includes the size of the header plus the size of all the packets in the data chunk. The size of this member is 32 bits.

object_version
: The version of the Data Chunk Header object. The size of this member is 16 bits.

num_packets
: Number of packets in the data chunk. This member is present on all Data Chunk Header objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

next_data_header
: Offset from start of file to the next data chunk. A non-zero value refers to the file offset of the next data chunk. A value of zero means there are no more data chunks in this file. This field is not typically used. This member is present on all Data Chunk Header objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

Data Packet Header

Following a data chunk header is `num_packet` data packets. These packets can all be from the same stream, or packets from different streams can follow one another. These packets, whether from the same stream or from different streams, should have an increasing value of timestamp. That is, the timestamp of a packet should be greater than or equal to the timestamp of the previous packet in the file.

The following pseudo-structure describes the details of the packet:

```
Media_Packet_Header
{
  UINT16                object_version;
  if ((object_version == 0) || (object_version == 1))
```

```
  {
    UINT16        length;
    UINT16        stream_number;
    UINT32        timestamp;
    if (object_version == 0)
    {
      UINT8        packet_group;
      UINT8        flags;
    }
    else if (object_version == 1)
    {
      UINT16        asm_rule;
      UINT8          asm_flags;
    }
    UINT8[length]        data;
  }
  else
  {
    StreamDone();
  }
}
```

The Media Packet Header contains the following fields:

object_version

The version of the Media Packet Header object. The size of this member is 16 bits.

length

The length of the packet in bytes. This member is present on all Media Packet Header objects with an `object_version` of 0 (zero) or 1. The size of this member is 16 bits.

stream_number

The 16-bit alias used to associate data packets with their associated Media Properties Header. This member is present on all Media Packet Header objects with an `object_version` of 0 (zero) or 1. The size of this member is 16 bits.

timeStamp

The time stamp of the packet in milliseconds This member is present on all Media Packet Header objects with an `object_version` of 0 (zero) or 1. The size of this member is 32 bits.

packet_group

The packet group to which the packet belongs. If packet grouping is not used, set this field to 0 (zero). This member is present on all Media Packet Header objects with an `object_version` of 0 (zero). The size of this member is 8 bits.

flags

Flags describing the properties of the packet. The following flags are

defined:

- HX_RELIABLE_FLAG

  If this flag is set, the packet is delivered reliably.

- HX_KEYFRAME_FLAG

  If this flag is set, the packet is part of a key frame or in some way marks a boundary in your data stream.

This member is present on all Media Packet Header objects with an `object_version` of 0 (zero). The size of this member is 8 bits.

asm_rule

The ASM rule assigned to this packet. Only present if `object_version` equals 1. The size of this member is 16 bits.

asm_flags

Contains `HX_` flags that dictate stream switching points. Only present if `object_version` equals 1. The size of this member is 8 bits.

data

The application-specific media data. This member is present on all Media Packet Header objects with an `object_version` of 0 (zero) or 1. The size of this member is variable.

Index Section

The index section of the RealMedia file consists of a Index Chunk Header that describes the contents of the index section, followed by a series of index records. Note that the size field of the Index Chunk Header is the size of the entire index chunk, including the index records.

Index Section Header

The Index Chunk Header marks the start of the index chunk. There is usually one index chunk per stream in a RealMedia file. The following pseudo-structure describes the Index chunk header.

```
Index_Chunk_Header
{
  u_int32     object_id;
  u_int32     size;
  u_int16      object_version
;
  if (object_version == 0)
```

```
  {
    u_int32     num_indices;
    u_int16     stream_number;
    u_int32     next_index_header;
  }
}
```

The Index Chunk Header contains the following fields:

object_id
: The unique object ID for the Index Chunk Header ("INDX"). The size of this member is 32 bits.

size
: The size of the Index Chunk in bytes. The size of this member is 32 bits.

object_version
: The version of the Index Chunk Header object. The size of this member is 16 bits.

num_indices
: Number of index records in the index chunk. This member is present on all Index Chunk Header objects with an `object_version` of 0 (zero). The size of this member is 32 bits

stream_number
: The stream number for which the index records in this index chunk are associated. This member is present on all Index Chunk Header objects with an `object_version` of 0 (zero). The size of this member is 16 bits.

next_index_header
: Offset from start of file to the next index chunk. This member enables RealMedia file format readers to find all the index chunks quickly. A value of zero for this member indicates there are no more index headers in this file. This member is present on all Index Chunk Header objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

Index Record

The index section of a RealMedia file consists of a series of index record objects. Each index record contains information for quickly finding a packet of a particular time stamp for a physical stream. The following pseudo-structure describes the details of each index record:

```
IndexRecord
{
  UINT16   object_version;
  if (object_version == 0)
  {
    u_int32  timestamp;
    u_int32  offset;
    u_int32   packet_count_for_this_packet;
  }
}
```

An Index Record contains the following fields:

object_version
: The version of the Index Record object. The size of this member is 16 bits.

timestamp
: The time stamp (in milliseconds) associated with this record. This member is present on all Index Record objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

offset
: The offset from the start of the file at which this packet can be found. This member is present on all Index Record objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

packet_count_for_this_packet
: The packet number of the packet for this record. This is the same number of packets that would have been seen had the file been played from the beginning to this point. This member is present on all Index Record objects with an `object_version` of 0 (zero). The size of this member is 32 bits.

Metadata Section

The metadata section of the RealMedia file consists of a tag containing a set of named metadata properties that describe the media file. These properties can be text, integers, or any binary data. The tag is preceded by a header that identifies the size of the entire metadata section. Following the tag, the footer identifies the size of the tag. Since the metadata section is found at the end of the file, the footer can be used to expedite seeking backwards. At the end of the metadata section, and the file itself, is an ID3v1 tag.

Metadata Section Header

The Metadata Section Header marks the start of the metadata section. There is one metadata section in a RealMedia file and it is expected to be at the end of the file. The following structure describes the Metadata section header:

```
MetadataSectionHeader
{
  u_int32        object_id;
  u_int32        size;
}
```

The Metadata Section Header contains the following fields:

object_id

The unique object ID for the Metadata Section Header ("RMMD"). The size of this member is 32 bits.

size

The size of the full metadata section in bytes. The size of this member is 32 bits.

Metadata Tag

The metadata tag of a RealMedia file consists of a series of properties. The properties are represented as a tree hierarchy with one unnamed root property. Each property contains a type and value, as well as multiple (optional) sub-properties. The following structure describes the details of the metadata tag:

```
MetadataTag
{
  u_int32        object_id;
  u_int32        object_version;
  u_int8[]        properties;
}
```

The Metadata Tag contains the following fields:

object_id

The unique object ID for the Metadata Tag ("RJMD"). The size of this member is 32 bits.

object_version

The version of the Metadata Tag. The size of this member is 32 bits.

properties[]

The `MetadataProperty` structure that makes up the metadata tag (see "Metadata Property Structure" for more details). As mentioned above,

the properties will be represented as one unnamed root metadata property with multiple sub-properties, each with their own optional sub-properties. These will be nested, as in a tree.

Metadata Property Structure

The following sample describes the details of the `MetadataProperty` structure:

```
MetadataProperty
{
  u_int32        size;
  u_int32        type;
  u_int32        flags;
  u_int32        value_offset;
  u_int32        subproperties_offset;
  u_int32        num_subproperties;
  u_int32        name_length;
  u_int8[name_length]    name;
  u_int32        value_length;
  u_int8[value_length]    value;
  PropListEntry[num_subproperties]    subproperties_list;
  MetadataProperty[num_subproperties]    subproperties;
}
```

The `MetadataProperty` structure contains the following fields:

size

The size of the `MetadataProperty` structure in bytes. The size of this member is 32 bits.

type

The type of the value data. The data in the value array can be one of the following types:

- MPT_TEXT

  The value is string data.

- MPT_TEXTLIST

  The value is a separated list of strings, separator specified as sub-property/type descriptor.

- MPT_FLAG

  The value is a boolean flag—either 1 byte or 4 bytes, check size value.

- MPT_ULONG

  The value is a four-byte integer.

- MPT_BINARY

  The value is a byte stream.

- MPT_URL

  The value is string data.

- MPT_DATE

  The value is a string representation of the date in the form: YYYYmmDDHHMMSS (m = month, M = minutes).

- MPT_FILENAME

  The value is string data.

- MPT_GROUPING

  This property has subproperties, but its own value is empty.

- MPT_REFERENCE

  The value is a large buffer of data, use sub-properties/type descriptors to identify mime-type.

The size of this member is 32 bits.

flags

Flags describing the property. The following flags are defined these can be used in combination:

- MPT_READONLY

  Read only, cannot be modified.

- MPT_PRIVATE

  Private, do not expose to users.

- MPT_TYPE_DESCRIPTOR

Type descriptor used to further define type of value.

The size of this member is 32 bits.

value_offset

The offset to the `value_length` , relative to the beginning of the `MetadataProperty` structure. The size of this member is 32 bits.

subproperties_offset

The offset to the `subproperties_list` , relative to the beginning of the `MetadataProperty` structure. The size of this member is 32 bits.

num_subproperties

The number of subproperties for this `MetadataProperty` structure. The size of this member is 32 bits.

name_length

The length of the name data, including the null-terminator. The size of this member is 32 bits.

name[]

The name of the property (string data). The size of this member is designated by `name_length` .

value_length

The length of the value data. The size of this member is 32 bits.

value[]

The value of the property (data depends on the type specified for the property). The size of this member is designated by `value_length` .

subproperties_list[]

The list of `PropListEntry` structures. The PropListEntry structure identifies the offset for each property (see "PropListEntry Structure" for more details. The size of this member is num_subproperties * sizeof(PropListEntry).

subproperties[]

The sub-properties. Each sub-property is a `MetadataProperty` structure with its own size, name, value, sub-properties, and so on. The size of this member is variable.

PropListEntry Structure

The following sample describes the details of the `PropListEntry` structure:

```
PropListEntry
{
  u_int32        offset;
  u_int32        num_props_for_name;
}
```

The `PropListEntry` structure contains the following fields:

offset

The offset for this indexed sub-property, relative to the beginning of the containing `MetadataProperty` . The size of this member is 32 bits.

num_props_for_name

The number of sub-properties that share the same name. For example, a lyrics property could have multiple versions as differentiated by the language sub-property type descriptor. The size of this member is 32 bits.

Metadata Section Footer

The metadata section footer marks the end of the metadata section of a RealMedia file. The metadata section footer contains the size of the metadata tag. Since the metadata section is at the end of the file, the section footer lies a fixed offset of 140 bytes from the end of the file. The size of the metadata tag enables a file reader to quickly locate the beginning of the metadata tag relative to the end of the file. The following structure describes the Metadata section footer.

```
MetadataSectionFooter
{
  u_int32        object_id;
  u_int32        object_version;
  u_int32        size;
}
```

The `MetadataSectionFooter` contains the following fields:

object_id

The unique object ID for the Metadata Section Footer ("RMJE"). The size of this member is 32 bits.

object_version

The version of the metadata tag. The size of this member is 32 bits.

size

The size of the preceding metadata tag. The size of this member is 32 bits.

ID3v1 Tag

The ID3v1 Tag is at the end of the metadata section and is expected to be at the end of the entire file. It is a fixed size—128 bytes—and begins with the characters "TAG". Futher information about the informal ID3v1 standard can be found at http://id3.org/id3v1.html .
