Advanced Systems Format (ASF) Specification

Revision 01.20.06

Microsoft Corporation

January 2012

End User License Agreement

Microsoft Advanced Systems Format (ASF) Specification version 1.2

IMPORTANT—READ CAREFULLY: This Microsoft Agreement ("Agreement") is a legal agreement between you (either an individual or a single entity) and Microsoft Corporation ("Microsoft") for the version of the Microsoft specification identified above which you are about to download ("Specification"). By downloading, copying, or otherwise using the Specification, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, do not download, copy or otherwise use the Specification.

The Specification is owned by Microsoft or its suppliers and is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties.

The capitalized terms used in this Agreement are defined as follows:

- "Advanced Systems Format" or "ASF" means version 1.2 of the extensible file storage format developed by or for Microsoft for authoring, editing, archiving, distributing, streaming, playing, referencing, or otherwise manipulating content.
- "Necessary Claims" means any and all claim(s), but only such claim(s), of a patent or patent application that (a) are owned, controlled, or sublicensable (without payment of royalties to an unaffiliated third party) by Microsoft now or at any future time; and (b) are necessarily infringed by implementing ASF as set forth in the Specification. Notwithstanding the foregoing sentence, Necessary Claims do not include any claims: (i) to any enabling technologies that may be necessary to make or use any product or portion thereof that includes ASF (e.g., enabling semiconductor manufacturing technology, compiler technology, object oriented technology, operating system technology, protocols, programming interfaces, etc.); or (ii) covering the implementation of other specifications, technical documentation or technology merely referred to in the Specification.
- "Windows Media Codecs" means all versions of the audio, video, and data compression/decompression algorithms ("codecs") that have a ASF Codec Description that begins with "Windows Media" or "Microsoft". Windows Media Codecs includes, but is not limited to, the "Windows Media Audio" codecs, "Windows Media Video" codecs, and "Microsoft MPEG-4" codec.

All other initially capitalized terms have the meanings assigned to them elsewhere in this Agreement.

1. LICENSE.

(a) **Specification.** Provided you comply with all terms and conditions of this Agreement, including without limitation Section 2 below, Microsoft grants you the following limited, non-exclusive, world-wide, royalty-free, non-assignable, nontransferable, non-sublicenseable license during the Term (defined below), under any copyrights owned or licensable by Microsoft without payment of consideration to unaffiliated third parties, to: (i) reproduce and internally use a reasonable number of copies of the Specification in its entirety as a reference for the sole purpose of implementing ASF in your hardware, application, or utilities (your "Solutions"); (ii) reproduce and internally use your implementations of ASF made pursuant to the terms of this Agreement (your "Implementations") in source code form solely for internal development and testing of your Solutions, and (iii) reproduce and have reproduced in object code form only, your Implementations and distribute, directly and indirectly, your Implementations (only in object code form) solely as part of and for use with your Solutions.

(b) **Necessary Claims.** Provided you comply with all terms and conditions of this Agreement, including without limitation Section 2 below, Microsoft grants you the following limited, non-exclusive, world-wide, royalty-free, non-assignable, nontransferable, non-sublicenseable license during the Term under its Necessary Claims to make, use and distribute in object code form (in accordance with the distribution criteria set forth in Section 1(a)(iii) above) your Implementations that fully comply with the Specification.

(c) **Reserved Rights.** The foregoing license is applicable only to the version of the Specification which you are about to download. This Agreement does not grant you any rights to any additional versions of or extensions to the Specification. Microsoft and its suppliers retain title and all ownership rights to the Specification and the information contained therein. All rights not expressly granted are reserved to Microsoft. Microsoft may have patents or pending patent applications, trademarks, copyrights, trade secrets or other intellectual property rights covering the subject matter in the Specification. The furnishing of this Specification does not give you any license

to these patents, trademarks, trade secrets, copyrights, or other intellectual property rights, except as specifically set forth in Sections 1(a) and 1(b) above.

2. **DESCRIPTION OF ADDITIONAL LIMITATIONS.** Without limiting the conditions set forth in Section 1 above, your rights under Section 1 are expressly conditioned upon your compliance with each of the following limitations:

   (a) You may not alter or remove any copyright, trademark or other protective notices or legends from any copy of the Specification.

   (b) You may not provide, publish or otherwise distribute the Specification to any third party. Further, you shall use commercially reasonable efforts to ensure that the use or distribution of your Solutions, including your Implementations as incorporated into your Solutions, shall not in any way disclose or reveal the information contained in the Specification.

   (c) Your Implementations as incorporated into your Solutions must implement the Specification in its entirety. By way of clarification of the foregoing, you are not required to implement any portion of the Specification that is identified as "optional". However, if you elect to implement a portion of the Specification that is identified as optional, you must also implement that optional portion of the Specification in its entirety.

   (d) Your Solutions shall not circumvent or compromise the protection of content protected with Microsoft's digital rights management.

   (e) To promote interoperability with legacy and future solutions built on Windows Media technologies, Microsoft recommends that your Solutions follow these guidelines: (i) if your Solutions save to permanent storage, write to a network, or otherwise export content compressed with Windows Media Codecs, such content should be contained within ASF and (ii) if your Solutions create ASF files, or register for any file types or MIME types associated with ASF files, your Solutions should adhere to the ASF file naming and registration conventions posted on the http://go.microsoft.com/fwlink/?LinkId=11652 web site.

   (f) For a variety of reasons, including without limitation, because you do not have the right to sublicense the Necessary Claims, your license rights to the Specification are conditioned upon your not creating or distributing your Implementations in any manner that would cause ASF (whether embodied in your Implementation or otherwise) to become subject to any of the terms of an Excluded License. An "Excluded License" is any license that requires as a condition of use, modification and/or distribution of software subject to the Excluded License, that such software or other software combined and/or distributed with such software be (x) disclosed or distributed in source code form; (y) licensed for the purpose of making derivative works; or (z) redistributable at no charge;

3. **TERM**. The term of this Agreement ("Term") commences upon your downloading of the Specification and expires without notice on January 1, 2012, unless terminated sooner as provided in Section 4. After the expiration or termination of the Term, you must cease creating any new Implementations. If you have complied with the terms and conditions of this Agreement and Microsoft has not terminated this Agreement as provided in Section 4, you may continue to exercise the license rights granted herein after the expiration of the Term, including the right to use a reasonable number of copies of the Specification, subject to the conditions and restrictions herein and solely to the limited extent needed to (i) continue to distribute your Solutions containing your Implementations where such Solutions were first commercially released in a general, non-beta form during the Term ("Legacy Solutions") and (ii) develop bug fixes and provide product support for your Legacy Solutions. You may not continue to exercise the license rights granted herein after the expiration of the Term (i) if Microsoft has terminated this Agreement as provided in Section 4 or (ii) if you do not continue to comply with any of the conditions and restrictions herein that were applicable to your exercise of the license rights during the Term.

4. **TERMINATION**. Without prejudice to any other rights, Microsoft may terminate this Agreement if you fail to comply with any of the terms and conditions of this Agreement ("Termination for Cause"). Immediately following a Termination for Cause, you must destroy all copies of the Specification in your possession or under your control and cease any further distribution of your Solutions that use or incorporate Implementations.

5. **SUPPORT.** Microsoft is not obligated to provide technical or other support for the Specification. You are responsible for any and all maintenance, end-user support, technical support and updates for your Solutions.

6. **EXPORT RESTRICTIONS.** You acknowledge that the Specification is subject to U.S. export jurisdiction. You agree to comply with all applicable international and national laws that apply to the Specification, including the U.S. Export Administration Regulations, as well as end-user, end-use and country destination restrictions issued by U.S. and other governments. For additional information on exporting Microsoft products, see http://www.microsoft.com/exporting/.

7. **DISCLAIMER OF WARRANTIES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, MICROSOFT AND ITS SUPPLIERS PROVIDE TO YOU THE SPECIFICATION (AND ALL INTELLECTUAL PROPERTY THEREIN), AND ANY (IF ANY) SUPPORT SERVICES RELATED TO THE SPECIFICATION ("SUPPORT SERVICES") AS IS AND WITH ALL FAULTS; AND MICROSOFT AND ITS SUPPLIERS HEREBY DISCLAIM WITH RESPECT TO THE SPECIFICATION AND SUPPORT SERVICES ALL WARRANTIES AND CONDITIONS, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY (IF ANY) WARRANTIES OR CONDITIONS OF OR RELATED TO: MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OR COMPLETENESS OF**

RESPONSES, RESULTS, WORKMANLIKE EFFORT AND LACK OF NEGLIGENCE. ALSO THERE IS NO WARRANTY, DUTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NONINFRINGEMENT. THE ENTIRE RISK ARISING OUT OF USE OF THE SPECIFICATION (AND ALL INTELLECTUAL PROPERTY THEREIN) AND ANY SUPPORT SERVICES REMAINS WITH YOU.

8. **EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL MICROSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SPECIFICATION, ANY INTELLECTUAL PROPERTY THEREIN, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS AGREEMENT, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF MICROSOFT OR ANY SUPPLIER, AND EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**

9. **LIMITATION OF LIABILITY AND REMEDIES. NOTWITHSTANDING ANY DAMAGES THAT YOU MIGHT INCUR FOR ANY REASON WHATSOEVER (INCLUDING, WITHOUT LIMITATION, ALL DAMAGES REFERENCED ABOVE AND ALL DIRECT OR GENERAL DAMAGES), THE ENTIRE LIABILITY OF MICROSOFT AND ANY OF ITS SUPPLIERS UNDER ANY PROVISION OF THIS AGREEMENT AND YOUR EXCLUSIVE REMEDY FOR ALL OF THE FOREGOING SHALL BE LIMITED TO THE GREATER OF THE AMOUNT ACTUALLY PAID BY YOU FOR THE SPECIFICATION OR U.S.$5.00. THE FOREGOING LIMITATIONS, EXCLUSIONS AND DISCLAIMERS SHALL APPLY TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, EVEN IF ANY REMEDY FAILS ITS ESSENTIAL PURPOSE.**

10. **APPLICABLE LAW**. This Agreement is governed by the laws of the State of Washington.

11. **ENTIRE AGREEMENT**. This Agreement is the entire agreement between you and Microsoft relating to the Specification and it supersedes all prior or contemporaneous oral or written communications, proposals and representations with respect to the Specification.

Should you have any questions concerning this EULA, or if you desire to contact Microsoft for any reason, please contact the Microsoft subsidiary serving your country, or write: Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399.

1. Introduction

Advanced Systems Format (ASF) is an extensible file format designed to store coordinated digital media data. It supports data delivery over a wide variety of networks and is also suitable for local playback.

ASF was designed with the following goals:

- To support efficient playback from digital media servers, HTTP servers, and local storage devices.
- To support scalable digital media types such as audio and video.
- To permit a single digital media composition to be presented over a wide range of bandwidths.
- To allow authoring control over digital media stream relationships, especially in constrained-bandwidth scenarios.
- To be independent of any particular digital media composition system, computer operating system, or data communications protocol.

Each ASF file is composed of one or more digital media streams. The file header specifies the properties of the entire file, along with stream-specific properties. Digital media data, stored after the file header, references a particular digital media stream number to indicate its type and purpose. The delivery and presentation of all digtial media stream data is aligned to a common timeline.

The ASF file definition includes the specification of commonly used media types. If an implementation supports media types from within this set of standard media types, then that media type must be supported in the manner described in this document if the resulting content is to be considered "content compliant" with the ASF specification.

ASF supports the transmission of "live content" over a network. Live content refers to digital media content, which may or may not ever be recorded upon a persistent medium (for example, a disk, CD, DVD, and so on). This use explicitly and solely means that information describing the digital media content must have been received before the digital media data itself is received (in order to interpret the digital media data), and that this information must convey the semantics of the ASF Header Object. Similarly, the received data must conform to the format of the ASF Data Packets. No additional information should be conveyed. Specifically, this use explicitly does not refer to or contain any information about network control protocols or network transmission protocols. It refers solely to the order of information arrival (header semantics before data) and the data format.

A partially downloaded ASF file may still be perfectly functional. As long as the required header objects and some complete set of the data objects are available, it is possible to seek to any position (backward and forward) within the partially downloaded file. Seeking in an ASF file does not require the use of an index object; however, many implementations will require the index in order to gain efficient access to the data.

ASF is a digital media presentation file format. It supports live and on-demand digital media content. ASF files may be edited, but ASF is specifically designed for streaming and/or local playback.

1.1 About this specification

This specification is intended to be compatible with previous versions of the ASF Specification. The format defined herein is an extension of the format defined in previous versions of this specification and is not a new format. Content created according to this specification should be compatible with earlier implementations of ASF (subject to some issues that are outlined in section 8).

1.2 Conventions

In this document the following conventions are used:

- "Shall" or "must" indicates a mandatory requirement
- "Should" indicates a recommended, but optional, course of action
- "May" indicates an optional course of action

All structure definitions assume 1-byte packing.

All references to Unicode strings imply NULL terminated strings unless otherwise indicated. The term "WCHAR" is used to indicate Unicode characters. ASF uses UTF-16, little endian, and the Byte-Order Marker (BOM) character is not present.

Fields marked "reserved" indicate that reading implementations should ignore the value but that writing implementations must set the value to whatever value is indicated in the specification for that field.

The following basic data types are used in this document.

Type	Size (bits)	Signed
BYTE	8	no
WCHAR	16	no
WORD	16	no
DWORD	32	no
QWORD	64	no
GUID	128	no

1.3 Document organization

- Section 1 provides an introduction to ASF and how to navigate this document.
- Section 2 provides a high-level overview of the ASF file format organization.
- Sections 3 and 4 define the various objects that comprise the ASF **Header Object**.
- Section 5 describes the organization of the ASF **Data Object**.
- Section 6 documents the internals of the ASF top-level index objects.

- Section 7 documents various features of ASF and how to use them.
- Section 8 details how to create content that is compatible with many existing ASF reading implementations; this concept is referred to as "content reach."
- Section 9 lists and describes standard ASF media types.
- Section 10 contains a list of the ASF GUIDs used throughout this document.
- Section 11 contains detailed information for specific codec data in ASF.

2. File format organization

This section provides a high-level overview of the ASF file format organization.

2.1 ASF object definition

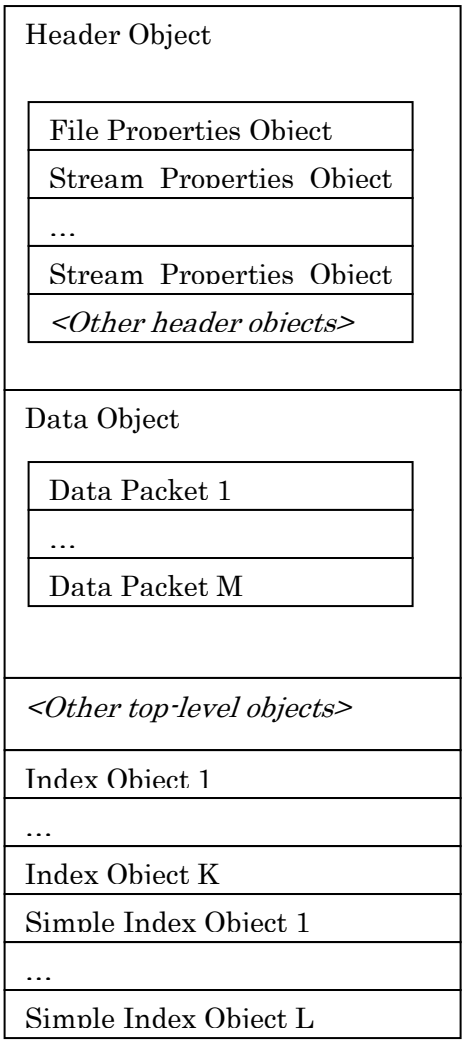
The base unit of organization for ASF files is called the ASF object. It consists of a 128-bit GUID for the object, a 64-bit integer object size, and the variable-length object data. The value of the object size field is the sum of 24 bytes plus the size of the object data in bytes. The following table illustrates the ASF object structure.

Field	Size (bytes)
Object GUID	16
Object Size	8
Object Data	varies (≥0)

All ASF objects and structures (including data packet headers) are stored in little-endian byte order (the inverse of network byte order). However, ASF files can contain digital media stream data in either byte order within the data packets.

2.2 Top-level file structure

ASF files are logically composed of three types of top-level objects: the **Header Object**, the **Data Object**, and the **Index Object(s)**. The **Header Object** is mandatory and must be placed at the beginning of every ASF file. The **Data Object** is also mandatory and must follow the **Header Object**. The **Index Object(s)** are optional, but they are useful in providing time-based random access into ASF files. When present, the **Index Object(s)** must be the last object in the ASF file. The following diagram illustrates the top-level ASF file structure.



Implementations shall ignore any standard or non-standard object that they do not know how to handle. New top-level objects should be added only between the **Data Object** and **Index Object(s)**.

2.3 ASF top-level Header Object

The role of the **Header Object** is to provide a well-known byte sequence at the beginning of ASF files (the ASF_Header_Object GUID) and to contain all the information that is needed to properly interpret the information within the data object. The **Header Object** can optionally contain metadata such as bibliographic information.

Of the three top-level ASF objects, the **Header Object** is the only one that contains other ASF objects. The **Header Object** may include a number of standard objects including, but not limited to:

- **File Properties Object**. Contains global file attributes.
- **Stream Properties Object**. Defines a digital media stream and its characteristics.
- **Header Extension Object**. Allows additional functionality to be added to an ASF file while maintaining backward compatibility.
- **Content Description Object**. Contains bibliographic information.
- **Script Command Object**. Contains commands that can be executed on the playback timeline.
- **Marker Object**. Provides named jump points within a file.

Note that objects in the **Header Object** may appear in any order.

The complete list of header objects defined by this specification can be found in sections 3 and 4.

To be valid, the **Header Object** must contain a **File Properties Object**, a **Header Extension Object**, and at least one **Stream Properties Object**.

2.4 ASF top-level Data Object

The **Data Object** contains all the digital media data for an ASF file. This data is stored in the form of ASF Data Packets. For the context of this specification, Data Packets are stored with a fixed length. Each Data Packet contains data for one or several digital media streams. Data Packets are sorted within the **Data Object** based on the time when they should be delivered (send time). This sorting results in an interleaved data format.

2.5 ASF top-level index objects

There are two varieties of index objects: the **Simple Index Object** and the **Index Object** (of which there are a few varieties).

The **Simple Index Object** contains a time-based index of the video data in an ASF file. The time interval between index entries is constant and is stored in the **Simple Index Object**. For each video stream in an ASF file, there should be one instance of the **Simple Index Object**. The order in which those instances appear in the file is significant. The order of the **Simple Index Object**s should be identical to the order of the video streams based on their stream numbers. As described in detail in section 8.2.12, these objects need to be the last top-level objects in the file.

The category **Index Object** refers to the **Index Object**, the **Media Object Index Object**, and the **Timecode Index Object**, whose formats are all similar. The **Index Object**, like the **Simple Index Object**, indexes by time with a fixed time interval, but is not limited to video streams. The **Media Object Index Object** is a frame-based index that facilitates seeking by frame (or object number). The **Timecode Index Object** facilitates seeking by timecode in content that contains timecodes (see section 7.3.2.1).

3. ASF top-level Header Object

This section defines the various objects that comprise the ASF **Header Object**. Of the objects listed in this section, those described in sections 3.1 through 3.12 were also described in previous versions of this specification.

3.1 Header Object (mandatory, one only)

The **Header Object** is a container that holds any combination of the standard objects listed in the following sections. A **File Properties Object**, a **Header Extension Object** and at least one **Stream Properties Object** are required to be present. Implementations shall ignore any objects that they do not understand.

The **Header Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Number of Header Objects	DWORD	32
Reserved1	BYTE	8
Reserved2	BYTE	8

The fields are defined as follows:

Object ID

Specifies the GUID for the **Header Object**. This field shall be set to **ASF_Header_Object**.

Object Size

Specifies the size of the **Header Object**. This includes the sum of 24 bytes plus the size of the objects contained in the **Header Object**. Valid values are at least 30 bytes.

Number of Header Objects

Specifies the number of objects contained within the **Header Object**, not including this one. In addition, the **Header Extension Object** (sections 3.4 and 4) is counted as exactly one object regardless of the number of subobjects contained within it.

Reserved1

This field must be set to the value 0x01. ASF parsers may safely ignore this value.

Reserved2

This field must be set to the value 0x02. If the value is different when read, the application should fail to source the content.

3.2 File Properties Object (mandatory, one only)

The **File Properties Object** defines the global characteristics of the combined digital media streams found within the **Data Object**.

The **File Properties Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
File ID	GUID	128
File Size	QWORD	64
Creation Date	QWORD	64
Data Packets Count	QWORD	64
Play Duration	QWORD	64
Send Duration	QWORD	64
Preroll	QWORD	64
Flags	DWORD	32
Broadcast Flag		1 (LSB)
Seekable Flag		1
Reserved		30
Minimum Data Packet Size	DWORD	32
Maximum Data Packet Size	DWORD	32
Maximum Bitrate	DWORD	32

The fields are defined as follows:

Object ID

Specifies the GUID for the **File Properties Object**. This field shall be set to **ASF_File_Properties_Object**.

Object Size

Specifies the size, in bytes, of the **File Properties Object**. Valid values are at least 104 bytes.

File ID

Specifies the unique identifier for this file. The value of this field shall be regenerated every time the file is modified in any way. The value of this field shall be identical to the value of the **File ID** field of the **Data Object**.

File Size

Specifies the size, in bytes, of the entire file. The value of this field is invalid if the **Broadcast Flag** bit in the **Flags** field is set to 1.

Creation Date

Specifies the date and time of the initial creation of the file. The value is given as the number of 100-nanosecond intervals since January 1, 1601, according to Coordinated Universal Time (Greenwich Mean Time). The value of this field may be invalid if the **Broadcast Flag** bit in the **Flags** field is set to 1.

Data Packets Count

Specifies the number of Data Packet entries that exist within the **Data Object**. The value of this field is invalid if the **Broadcast Flag** bit in the **Flags** field is set to 1.

Play Duration

Specifies the time needed to play the file in 100-nanosecond units. This value should include the duration (estimated, if an exact value is unavailable) of the last media object in the presentation. The value of this field is invalid if the **Broadcast Flag** bit in the **Flags** field is set to 1.

Send Duration

Specifies the time needed to send the file in 100-nanosecond units. This value should include the duration of the last packet in the content. The value of this field is invalid if the **Broadcast Flag** bit in the **Flags** field is set to 1. Players can ignore this value.

Preroll

Specifies the amount of time to buffer data before starting to play the file, in millisecond units. If this value is nonzero, the **Play Duration** field and all of the payload **Presentation Time** fields have been offset by this amount. Therefore,

player software must subtract the value in the preroll field from the play duration and presentation times to calculate their actual values.

It follows that all payload **Presentation Time** fields need to be at least this value.

Flags

The flags are stored in Least Significant Byte (LSB) order.

Broadcast Flag (bit 0)

Specifies, if set, that a file is in the process of being created (for example, for recording applications), and thus that various values stored in the header objects are invalid. It is highly recommended that post-processing be performed to remove this condition at the earliest opportunity.

Seekable Flag (bit 1)

Specifies, if set, that a file is seekable. Note that for files containing a single audio stream and a **Minimum Data Packet Size** field equal to the **Maximum Data Packet Size** field, this flag shall always be set to 1. For files containing a single audio stream and a video stream or mutually exclusive video streams, this flag is only set to 1 if the file contains a matching **Simple Index Object** for each regular video stream (that is, video streams that are not hidden according to the method described in section 8.2.2).

Reserved (bits 2 – 31)

Remaining 30 reserved flags shall be set to 0.

Minimum Data Packet Size

Specifies the minimum Data Packet size in bytes. In general, the value of this field is invalid if the **Broadcast Flag** bit in the **Flags** field is set to 1. However, for the purposes of this specification, the values for the **Minimum Data Packet Size** and **Maximum Data Packet Size** fields shall be set to the same value, and this value should be set to the packet size, even when the **Broadcast Flag** in the **Flags** field is set to 1.

Maximum Data Packet Size

Specifies the maximum Data Packet size in bytes. In general, the value of this field is invalid if the **Broadcast Flag** bit in the **Flags** field is set to 1. However, for the purposes of this specification, the values of the **Minimum Data Packet Size** and **Maximum Data Packet Size** fields shall be set to the same value, and this value should be set to the packet size, even when the **Broadcast Flag** field is set to 1.

Maximum Bitrate

Specifies the maximum instantaneous bit rate in bits per second for the entire file. This shall equal the sum of the bit rates of the individual digital media streams. It shall be noted that the digital media stream includes ASF data packetization overhead as well as digital media data in payloads. Only those streams that have a free-standing **Stream Properties Object** in the header shall have their bit rates included in the sum; streams whose **Stream Properties Object** exists as part of an **Extended Stream Properties Object** in the **Header Extension Object** shall not have their bit rates included in this sum, except when this value would otherwise be 0.

3.3 Stream Properties Object (mandatory, one per stream)

The **Stream Properties Object** defines the specific properties and characteristics of a digital media stream. This object defines how a digital media stream within the **Data Object** is interpreted, as well as the specific format (of elements) of the Data Packet itself (for more information, see section 5.2).

Whereas every stream in an ASF presentation, including each stream in a mutual exclusion relationship, must be represented by a **Stream Properties Object**, in certain cases, this object might be found embedded in the **Extended Stream Properties Object** (section 4.1). See section 8.2 for situations in which that should occur.

The **Stream Properties Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Stream Type	GUID	128
Error Correction Type	GUID	128
Time Offset	QWORD	64
Type-Specific Data Length	DWORD	32
Error Correction Data Length	DWORD	32
Flags	WORD	16
Stream Number		7 (LSB)
Reserved		8
Encrypted Content Flag		1
Reserved	DWORD	32
Type-Specific Data	BYTE	varies
Error Correction Data	BYTE	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Stream Properties Object**. The value of this field shall be set to **ASF_Stream_Properties_Object**.

Object Size

Specifies the size, in bytes, of the **Stream Properties Object**. Valid values are at least 78 bytes.

Stream Type

Specifies the type of the stream (for example, audio, video, and so on). Use one of the GUIDs defined in section 10.4. Any streams with unrecognized **Stream Type** values should be ignored.

Error Correction Type

Specifies the error correction type used by this digital media stream. For streams other than audio, this value should be set to **ASF_No_Error_Correction**. For audio streams, this value should be set to **ASF_Audio_Spread**. The possible GUIDs are defined in section 10.5, and the standard error correction schemes are described in section 9.1.

Time Offset

Specifies the presentation time offset of the stream in 100-nanosecond units. The value of this field is added to all of the timestamps of the samples in the stream. This value shall be equal to the send time of the first interleaved packet in the data section. The value of this field is typically 0. It is non-zero in the case when an ASF file is edited and it is not possible for the editor to change the presentation times and send times of ASF packets. Note that if more than one stream is present in an ASF file the offset values of all stream properties objects must be equal.

Type-Specific Data Length

Specifies the number of bytes in the **Type-Specific Data** field.

Error Correction Data Length

Specifies the number of bytes in the **Error Correction Data** field.

Flags

The flags are stored in LSB order.

Stream Number (bits 0-6)

Specifies the number of this stream. 0 is an invalid stream. Valid values are between 1 and 127. The numbers assigned to streams in an ASF presentation may be any combination of unique values; parsing logic must not assume that streams are numbered sequentially.

Reserved (bits 7-14)

These bits are reserved and should be set to 0.

Encrypted Content Flag (bit 15)

Specifies, if set, that the data contained in this stream is encrypted and will be unreadable unless there is a way to decrypt the stream.

Reserved

This field is reserved and should be set to 0.

Type-Specific Data

Specifies type-specific format data. The structure for the **Type-Specific Data** field is determined by the value stored in the **Stream Type** field. The structure for the **Type-Specific Data** field for standard ASF media types is detailed in section 9.

Error Correction Data

Specifies data specific to the error correction type. The structure for the **Error Correction Data** field is determined by the value stored in the **Error Correction Type** field. For example, an audio data stream might need to know how codec chunks were redistributed, or it might need a sample of encoded silence. For detailed information, see section 9.1.

3.4 Header Extension Object (mandatory, one only)

The **Header Extension Object** allows additional functionality to be added to an ASF file while maintaining backward compatibility. The **Header Extension Object** is a container containing 0 or more additional extended header objects. Extended header objects should conform to the ASF Object Structure (per section 2.1). For more information, see section 8.2.1.

The **Header Extension Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Reserved Field 1	GUID	128
Reserved Field 2	WORD	16
Header Extension Data Size	DWORD	32
Header Extension Data	BYTE	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Header Extension Object**. The value of this field shall be set to **ASF_Header_Extension_Object**.

Object Size

Specifies the size, in bytes, of the **Header Extension Object**. The value of this field shall be set to 46 bytes plus the value of Header Extension Data Size.

Reserved Field 1

This value shall be set to **ASF_Reserved_1** as defined in section 10.6.

Reserved Field 2

This value should be set to 6.

Header Extension Data Size

Specifies the number of bytes stored in the **Header Extension Data** field. This value may be 0 bytes or 24 bytes and larger. It should also be equal to the **Object Size** field minus 46 bytes.

Header Extension Data

Specifies an array of bytes containing additional extended header objects. This data should be interpreted as 0 or more extended header objects stored consecutively within the array of bytes. No empty space, padding, leading, or trailing bytes are allowed.

3.5 Codec List Object (optional, one only)

The **Codec List Object** provides user-friendly information about the codecs and formats used to encode the content found in the ASF file. The **Codec List Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Reserved	GUID	128
Codec Entries Count	DWORD	32
Codec Entries	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Codec List Object**. The value of this field shall be set to **ASF_Codec_List_Object**.

Object Size

Specifies the size, in bytes, of the **Codec List Object**. Valid values are at least 44 bytes.

Reserved

Reserved. This field shall be set to **ASF_Reserved_2** as defined in section 10.8.

Codec Entries Count

Specifies the number of entries listed in the **Codec Entries** field.

Codec Entries

Codec Entries are described in the following table.

Field name	Field type	Size (bits)
Type	WORD	16
Codec Name Length	WORD	16
Codec Name	WCHAR	varies
Codec Description Length	WORD	16
Codec Description	WCHAR	varies
Codec Information Length	WORD	16
Codec Information	BYTE	varies

The fields are defined as follows:

Type

Specifies the type of the codec used. Use one of the values in the following table.

Values	Meaning

0x0001	Video codec
0x0002	Audio codec
0xFFFF	Unknown codec

Codec Name Length

Specifies the number of Unicode characters stored in the **Codec Name** field.

Codec Name

Specifies an array of Unicode characters that contains the name of the codec used to create the content.

Codec Description Length

Specifies the number of Unicode characters stored in the **Codec Description** field.

Codec Description

Specifies an array of Unicode characters that contains the description of the format used to create the content.

Codec Information Length

Specifies the number of bytes stored in the **Codec Information** field.

Codec Information

Specifies an opaque array of information bytes about the codec used to create the content. The meaning of these bytes is determined by the codec.

3.6 Script Command Object (optional, one only)

The **Script Command Object** provides a list of type/parameter pairs of Unicode strings that are synchronized to the ASF file's timeline. Types can include URL or FILENAME values. Other type values may also be freely defined and used. The semantics and treatment of this set of types are defined by the local implementations. The parameter value (referred to as Commands in the follwoing table) is specific to the type field. You can use this type/parameter pairing for many purposes, including sending URLs to be launched by a client into an HTML frame (in other words, the URL type) or launching another ASF file for the chained continuous play of audio or video presentations (in other words, the FILENAME type). This object is also used as a method to stream text, as well as to provide script commands that you can use to control elements within the client environment.

The **Script Command Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Reserved	GUID	128
Commands Count	WORD	16
Command Types Count	WORD	16
Command Types	See below	varies
Commands	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Script Command Object**. The value of this field shall be set to **ASF_Script_Command_Object**.

Object Size

Specifies the size, in bytes, of the **Script Command Object**. Valid values are at least 44 bytes.

Reserved

Reserved. This field shall be set to **ASF_Reserved_3** as defined in section 10.9.

Commands Count

Specifies the number of **Command** structures in the **Script Command Object**.

Command Types Count

Specifies the number of **Command Type** structures in the **Script Command Object**.

Command Types

The structure of each **Command Type** entry is shown in the following table.

Field name	Field type	Size (bits)
Command Type Name Length	WORD	16
Command Type Name	WCHAR	varies

Command Type Name Length

Specifies the number of Unicode characters that are found within the **Command Type Name** field.

Command Type Name

Specifies the name of a type of command. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

Commands

The structure of each **Command** entry is shown in the following table.

Field name	Field type	Size (bits)
Presentation Time	DWORD	32
Type Index	WORD	16
Command Name Length	WORD	16
Command Name	WCHAR	varies

Presentation Time

Specifies the presentation time of the command, in milliseconds.

Type Index

Specifies the type of this command, as a zero-based index into the array of **Command Types** of this object.

Command Name Length

Specifies the number of Unicode characters that are found within the **Command Name** field.

Command Name

Specifies the name of this command. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

3.7 Marker Object (optional, 0 or 1)

The **Marker Object** is represented using the following structure.

Field name	Field type	Size (bits)

Object ID	GUID	128
Object Size	QWORD	64
Reserved	GUID	128
Markers Count	DWORD	32
Reserved	WORD	16
Name Length	WORD	16
Name	WCHAR	varies
Markers	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Marker Object**. The value of this field shall be set to **ASF_Marker_Object**.

Object Size

Specifies the size, in bytes, of the **Marker Object**. Valid values are at least 48 bytes.

Reserved

Reserved. This field shall be set to **ASF_Reserved_4** as defined in section 10.10.

Markers Count

Specifies the number of **Marker** structures in the **Marker Object**.

Reserved

Specifies a reserved field. This field shall be set to 0.

Name Length

Specifies the number of bytes that are found within the **Name** field.

Name

Specifies the name of the **Marker Object**. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

Markers

The structure of each **Marker** entry is shown in the following table.

Field name	Field type	Size (bits)
Offset	QWORD	64
Presentation Time	QWORD	64
Entry Length	WORD	16
Send Time	DWORD	32
Flags	DWORD	32
Marker Description Length	DWORD	32
Marker Description	WCHAR	varies

The fields are defined as follows:

Offset

Specifies a byte offset into the **Data Object** to the actual position of the marker in the **Data Object**. ASF parsers must seek to this position to properly display data at the specified marker **Presentation Time**.

Presentation Time

Specifies the presentation time of the marker, in 100-nanosecond units.

Entry Length

Specifies the length in bytes of the **Send Time**, **Flags**, and **Marker Description Length** fields, and the number of bytes stored in the **Marker Description** field. (Note that the last version of this specification allowed for this field to optionally include some padding bytes at the end of the **Marker** entry in addition to

the sizes of the aforementioned fields. This was not correct; the value of this field needs to be exactly the size of the **Send Time**, **Flags**, and **Marker Description Length** fields plus the number of bytes stored in the **Marker Description** field.)

Send Time

Specifies the send time of the marker entry, in milliseconds.

Flags

Flags are reserved and should be set to 0.

Marker Description Length

Specifies the number of WCHARs that are found within the **Marker Description** field (including the nul-terminating character, if present). (Note that the last version of this specification indicated that this value is equal to the number of bytes that are found within the **Marker Description** field. This was not correct; this value is the number of WCHARs.)

Marker Description

Specifies an array of WCHARs containing a description of the marker entry. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

3.8 Bitrate Mutual Exclusion Object (optional, 0 or 1)

The **Bitrate Mutual Exclusion Object** identifies video streams that have a mutual exclusion relationship to each other (in other words, only one of the streams within such a relationship can be streamed at any given time and the rest are ignored). One instance of this object must be present for each set of objects that contains a mutual exclusion relationship. All video streams in this relationship must have the same frame size. The exclusion type is used so that implementations can allow user selection of common choices, such as bit rate.

The **Bitrate Mutual Exclusion Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Exclusion Type	GUID	128
Stream Numbers Count	WORD	16
Stream Numbers	WORD	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Mutual Exclusion Object**. The value of this field shall be set to **ASF_Bitrate_Mutual_Exclusion_Object**.

Object Size

Specifies the size, in bytes, of the **Bitrate Mutual Exclusion Object**. Valid values are at least 42 bytes.

Exclusion Type

Specifies the nature of the mutual exclusion relationship. Use one of the GUIDs defined in section 10.11.

Stream Numbers Count

Specifies the number of video streams listed in the **Stream Numbers** field.

Stream Numbers

Specifies the list of mutually exclusive video stream numbers. Valid values are between 1 and 127 as defined in the **Stream Properties Object**.

3.9 Error Correction Object (optional, one only)

The **Error Correction Object** defines the error correction method. This enables different error correction schemes to be used during content creation. The **Error Correction Object** contains provisions for opaque information needed by the error correction engine for recovery. For example, if the error correction scheme were a simple N+1 parity scheme, then the value of N would have to be available in this object.

Note that this does not refer to the same thing as the **Error Correction Type** field in the **Stream Properties Object**. Rather, this object indicates the error correction scheme used for the ASF packets as described in section 5.2.1.

The **Error Correction Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Error Correction Type	GUID	128
Error Correction Data Length	DWORD	32
Error Correction Data	BYTE	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Error Correction Object**. The value of this field shall be set to **ASF_Error_Correction_Object**.

Object Size

Specifies the size, in bytes, of the **Error Correction Object**. Valid values are at least 44 bytes.

Error Correction Type

Specifies the type of error correction.

Error Correction Data Length

Specifies the length, in bytes, of the **Error Correction Data** field.

Error Correction Data

Specifies data specific to the error correction scheme. The structure for the **Error Correction Data** field is determined by the value stored in the **Error Correction Type** field.

3.10 Content Description Object (optional, one only)

The **Content Description Object** lets authors record well-known data describing the file and its contents. This object is used to store standard bibliographic information such as title, author, copyright, description, and rating information. This information is pertinent to the entire file.

The **Content Description Object** is represented using the following structure.

Field name	Field type	Size (bits)

Object ID	GUID	128
Object Size	QWORD	64
Title Length	WORD	16
Author Length	WORD	16
Copyright Length	WORD	16
Description Length	WORD	16
Rating Length	WORD	16
Title	WCHAR	Varies
Author	WCHAR	Varies
Copyright	WCHAR	Varies
Description	WCHAR	Varies
Rating	WCHAR	Varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Content Description Object**. The value of this field shall be set to **ASF_Content_Description_Object**.

Object Size

Specifies the size, in bytes, of the **Content Description Object**. Valid values are at least 34 bytes.

Title Length

Specifies the number of bytes that comprise the title information stored in the **Title** field.

Author Length

Specifies the number of bytes that comprise the author information stored in the **Author** field.

Copyright Length

Specifies the number of bytes that comprise the copyright information stored in the **Copyright** field.

Description Length

Specifies the number of bytes that comprise the description information stored in the **Description** field.

Rating Length

Specifies the number of bytes that comprise the rating information stored in the **Rating** field.

Title

Specifies an array of WCHARs that contains the title information. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

Author

Specifies an array of WCHARs that contains the author information. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

Copyright

Specifies an array of WCHARs that contains the copyright information. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

Description

Specifies an array of WCHARs that contains the description information. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

Rating

Specifies an array of WCHARs that contains the rating information. Whereas it is highly recommended that this string include a nul-terminator, the nul-terminator may not be present.

3.11 Extended Content Description Object (optional, one only)

The **Extended Content Description Object** lets authors record data describing the file and its contents that is beyond the standard bibliographic information such as title, author, copyright, description, or rating information. This information is pertinent to the whole file. Each **Content Descriptor** stored in this object uses a name/value pair metaphor.

For more information about what types of attributes belong in the **Extended Content Description Object**, see section 7.4.

The **Extended Content Description Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Content Descriptors Count	WORD	16
Content Descriptors	See text	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Extended Content Description Object**. The value of this field shall be set to **ASF_Extended_Content_Description_Object**.

Object Size

Specifies the size, in bytes, of the **Extended Content Description Object**. Valid values are at least 26 bytes.

Content Descriptors Count

Specifies the number of entries in the **Content Descriptors** list.

Content Descriptors

The structure of each **Content Descriptor** entry is shown in the following table.

Field name	Field type	Size (bits)
Descriptor Name Length	WORD	16
Descriptor Name	WCHAR	varies
Descriptor Value Data Type	WORD	16
Descriptor Value Length	WORD	16
Descriptor Value	See text	varies

The fields are defined as follows:

Descriptor Name Length

Specifies the size, in bytes, of the **Descriptor Name** field.

Descriptor Name

Specifies an array of Unicode characters that contains the name of the descriptor.

Descriptor Value Data Type

Specifies the type of data stored in the **Descriptor Value** field. The types are defined in the following table.

Value	Type	Descriptor value length
0x0000	Unicode string	varies

0x0001	BYTE array	varies
0x0002	BOOL	32
0x0003	DWORD	32
0x0004	QWORD	64
0x0005	WORD	16

Descriptor Value Length

Specifies the number of bytes stored in the **Descriptor Value** field. For sizes, see the table in the **Descriptor Value Data Type** section.

Descriptor Value

Specifies the value for the **Content Descriptor** field. The type for this value is determined by the **Descriptor Value Data Type** field.

3.12 Stream Bitrate Properties Object (optional but recommended, one only)

The **Stream Bitrate Properties Object** defines the average bit rate of each digital media stream. It is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Bitrate Records Count	WORD	16
Bitrate Records	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Stream Bitrate Properties Object**. The value of this field shall be set to **ASF_Stream_Bitrate_Properties_Object**.

Object Size

Specifies the size, in bytes, of the **Stream Bitrate Properties Object**. Valid values are at least 26 bytes.

Bitrate Records Count

Specifies the number of records listed in the **Bitrate Records** field.

Bitrate Records

The structure of each **Bitrate Record** entry is shown in the following table.

Field name	Field type	Size (bits)
Flags	WORD	16
Stream Number		7 (LSB)
Reserved		9
Average Bitrate	DWORD	32

The fields are defined as follows:

Flags

The flags are stored in LSB order.

Stream Number (bits 0-6)

Specifies the number of this stream described by this record. 0 is an invalid stream. Valid values are between 1 and 127.

Reserved (bits 7-15)

These bits are reserved and should be set to 0.

Average Bitrate

Specifies the average bit rate of the stream in bits per second. This value should include an estimate of ASF packet and payload overhead associated with this stream.

3.13 Content Branding Object (optional, one only)

The **Content Branding Object** stores branding data for an ASF file, including information about a banner image and copyright associated with the file. It is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Banner Image Type	DWORD	32
Banner Image Data Size	DWORD	32
Banner Image Data	BYTE	varies
Banner Image URL Length	DWORD	32
Banner Image URL	char	varies
Copyright URL Length	DWORD	32
Copyright URL	char	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Content Branding Object**. The value of this field shall be set to **ASF_Content_Branding_Object**.

Object Size

Specifies the size, in bytes, of the **Content Branding Object**. Valid values are at least 40 bytes.

Banner Image Type

Specifies the type of data contained in the **Banner Image Data**. Valid values are 0 to indicate that there is no banner image data; 1 to indicate that the data represent a bitmap; 2 to indicate that the data represents a JPEG; and 3 to indicate that the data represents a GIF. If this value is set to 0, then the **Banner Image Data Size** field must be set to 0, and the **Banner Image Data** field must be empty.

Banner Image Data Size

Specifies the number of bytes in the **Banner Image Data** field.

Banner Image Data

This field contains the entire banner image, including the header for the appropriate image format.

Banner Image URL Length

Specifies the number of bytes in the **Banner Image URL** field.

Banner Image URL

Contains an ASCII (not WCHAR) string. If present, it is a link to more information about the banner image.

Copyright URL Length

Specifies the number of bytes in the **Copyright URL** field.

Copyright URL

Contains an ASCII (not WCHAR) string. If present, it is a link to more information about the copyright for the content.

3.14 Content Encryption Object (optional, 0 or 1)

The **Content Encryption Object** lets authors protect content by using Microsoft® Digital Rights Manager version 1. It is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Secret Data Length	DWORD	32
Secret Data	BYTE	varies
Protection Type Length	DWORD	32
Protection Type	char	varies
Key ID Length	DWORD	32
Key ID	char	varies
License URL Length	DWORD	32
License URL	char	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Content Encryption Object**. The value of this field shall be set to **ASF_Content_Encryption_Object**.

Object Size

Specifies the size, in bytes, of the **Content Encryption Object**. Valid values are larger than 40 bytes.

Secret Data Length

Specifies the number of bytes stored in the **Secret Data** field.

Secret Data

Specifies an array of bytes containing secret data.

Protection Type Length

Specifies the length in bytes of the **Protection Type** field.

Protection Type

Specifies a nul-terminated array of ASCII characters (not WCHARs) describing the type of protection mechanism used. The value of this field shall be set to **"DRM"**.

Key ID Length

Specifies the length in bytes of the **Key ID** field.

Key ID

Specifies a nul-terminated array of ASCII characters (not WCHARs) describing the key ID used.

License URL Length

Specifies the length in bytes of the **License URL** field.

License URL

Specifies a nul-terminated array of ASCII characters (not WCHARs) containing the URL from which a license to manipulate the content can be acquired.

3.15 Extended Content Encryption Object (optional, 0 or 1)

The **Extended Content Encryption Object** lets authors protect content by using the Windows Media Rights Manager 7 Software Development Kit (SDK). It is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Data Size	DWORD	32
Data	BYTE	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Extended Content Encryption Object**. The value of this field shall be set to **ASF_Extended_Content_Encryption_Object**.

Object Size

Specifies the size, in bytes, of the **Extended Content Encryption Object**. Valid values are larger than 24 bytes.

Data Size

Specifies the length, in bytes, of the data contained in the **Data** field.

Data

Specifies an array of bytes required by the DRM client to manipulate the protected content.

3.16 Digital Signature Object (optional, 0 or 1)

The **Digital Signature Object** lets authors sign the portion of their header that lies between the end of the **File Properties Object** and the beginning of the **Digital Signature Object**. It is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Signature Type	DWORD	32
Signature Data Length	DWORD	32
Signature Data	BYTE	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Digital Signature Object**. The value of this field shall be set to **ASF_Digital_Signature_Object**.

Object Size

Specifies the size, in bytes, of the **Digital Signature Object**. Valid values are larger than 32 bytes.

Signature Type

Specifies the type of digital signature used. This field shall be set to 2.

Signature Data Length

Specifies the number of bytes stored in the **Signature Data** field.

Signature Data

Specifies an array of bytes containing the digital signature.

3.17 Padding Object (optional, 0 to many)

The **Padding Object** is a dummy object that is used to pad the size of the **Header Object**. This object enables the size of any object stored in the **Header Object** to grow or shrink without having to rewrite the entire **Data Object** and **Index Object** sections of the ASF file. For instance, if entries in the **Content Description Object** or **Extended Content Description Object** need to be removed or shortened, the size of the **Padding Object** can be increased to compensate for the reduction in size of the **Content Description Object**. The ASF file can then be updated by overwriting the previous **Header Object** with the edited **Header Object** of identical size, without having to move or rewrite the data contained in the **Data Object**. Playback applications shall simply ignore **Padding Objects**. The **Padding Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Padding Data	BYTE	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Padding Object**. The value of this field shall be set to **ASF_Padding_Object**.

Object Size

Specifies the size, in bytes, of the **Padding Object**. Valid values are at least 24 bytes.

Padding Data

Specifies an array of bytes used for padding purposes. The length of the padding object data is calculated as Object Size – 24 bytes (sizeof(Object ID) + sizeof(Object Size)). As the name implies, the **Padding Data** field contains padding instead of meaningful data and shall always be ignored.

4. Objects in the ASF Header Extension Object

This section describes objects that, if present, must be embedded in the **Header Extension Object** as described in section 2.3. Custom ASF objects with user-defined meanings can be added to the **Header Extension Object** in addition to the objects in this list, but there is no guarantee of interoperability between implementations for those objects.

Note that objects defined in section 3 may appear in the **Header Extension Object**, so reading implementations should be able to understand those objects from within this object as well. However, a writing implementation should not, in general, place objects from section 3 into the

Header Extension Object, except when it wants to hide these objects from earlier reading implementations. For a discussion of when this is appropriate, see section 8.2.1.

4.1 Extended Stream Properties Object (optional, 1 per media stream)

The **Extended Stream Properties Object** defines additional optional properties and characteristics of a digital media stream that are not described in the **Stream Properties Object**.

Typically, the basic **Stream Properties Object** is present in the **Header Object**, and the **Extended Stream Properties Object** is present in the **Header Extension Object**. Sometimes, however, the **Stream Properties Object** for a stream may be embedded inside the **Extended Stream Properties Object** for that stream. This approach facilitates the creation of backward-compatible content and is appropriate for some of the scenarios discussed in section 8.

This object has an optional provision to include application-specific or implementation-specific data attached to the payloads of each digital media sample stored within a Data Packet. This data can be looked at as digital media sample properties and is stored in the **Replicated Data** field of a payload header. The **Payload Extension Systems** fields of the **Extended Stream Properties Object** describes what this data is and is necessary for that data to be parsed, if present.

The **Extended Stream Properties Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Start Time	QWORD	64
End Time	QWORD	64
Data Bitrate	DWORD	32
Buffer Size	DWORD	32
Initial Buffer Fullness	DWORD	32
Alternate Data Bitrate	DWORD	32
Alternate Buffer Size	DWORD	32
Alternate Initial Buffer Fullness	DWORD	32
Maximum Object Size	DWORD	32
Flags	DWORD	32
Reliable Flag		1 (LSB)
Seekable Flag		1
No Cleanpoints Flag		1
Resend Live Cleanpoints Flag		1
Reserved Flags		28
Stream Number	WORD	16
Stream Language ID Index	WORD	16
Average Time Per Frame	QWORD	64
Stream Name Count	WORD	16
Payload Extension System Count	WORD	16
Stream Names	See below	varies
Payload Extension Systems	See below	varies
Stream Properties Object	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Extended Stream Properties Object**. The value of this field shall be set to **ASF_Extended_Stream_Properties_Object**.

Object Size

Specifies the size, in bytes, of the **Extended Stream Properties Object**. Valid values are at least 88 bytes. If there is a **Stream Properties Object** embedded in this object, then the size of the **Stream Properties Object** must be included in this value.

Start Time

Specifies the presentation time of the first object, indicating where this digital media stream starts within the context of the timeline of the ASF file as a whole. This time value corresponds to presentation times as they appear in the data packets (adjusted by the preroll). This field is given in units of milliseconds and can optionally be set to 0, in which case it will be ignored.

End Time

Specifies the presentation time of the last object plus the duration of play, indicating where this digital media stream ends within the context of the timeline of the ASF file as a whole. This time value corresponds to presentation times as they appear in the data packets (adjusted by the preroll). This field is given in units of milliseconds and can optionally be set to 0, in which case it will be ignored.

Data Bitrate

Specifies the leak rate *R*, in bits per second, of a leaky bucket that contains the data portion of the stream without overflowing, excluding all ASF Data Packet overhead. The size of the leaky bucket is specified by the value of the **Buffer Size** field. This field must be non-zero. For more information about leaky buckets, see section 7.1.

Buffer Size

Specifies the size *B*, in milliseconds, of the leaky bucket used in the **Data Bitrate** definition.

Initial Buffer Fullness

Specifies the initial fullness, in milliseconds, of the leaky bucket used in the **Data Bitrate** definition. This is the fullness of the buffer at the instant before the first bit in the stream is dumped into the bucket. Typically, this value is set to 0. This value shall not exceed the value in the **Buffer Size** field.

Alternate Data Bitrate

Specifies the leak rate *RAlt*, in bits per second, of a leaky bucket that contains the data portion of the stream without overflowing, excluding all ASF Data Packet overhead. The size of the leaky bucket is specified by the value of the **Alternate Buffer Size** field. This value is relevant in most scenarios where the bit rate is not exactly constant, but it is especially useful for streams that have highly variable bit rates. This field can optionally be set to the same value as the **Data Bitrate** field.

Alternate Buffer Size

Specifies the size *BAlt*, in milliseconds, of the leaky bucket used in the **Alternate Data Bitrate** definition. This value is relevant in most scenarios where the bit rate is not exactly constant, but it is especially useful for streams that have highly variable bit rates. This field can optionally be set to the same value as the **Buffer Size** field.

Alternate Initial Buffer Fullness

Specifies the initial fullness, in milliseconds, of the leaky bucket used in the **Alternate Data Bitrate** definition. This is the fullness of the buffer at the instant before the first bit in the stream is dumped into the bucket. Typically, this value is set to 0. This value shall not exceed the value of the **Alternate Buffer Size** field.

Maximum Object Size

Specifies the maximum size of the largest sample stored in the data packets for a stream. A value of 0 means “unknown”.

Flags

The flags are stored in LSB order.

Reliable Flag

Specifies, if set, that this digital media stream, if sent over a network, must be carried over a reliable data communications transport mechanism. This should be set for streams that cannot recover after a lost media object.

Seekable Flag

This flag should be set only if the stream is seekable, either by using an index object or by estimating according to bit rate (as can sometimes be done with audio). This flag pertains to this stream only rather than to the entire file.

No Cleanpoint Flag

Specifies, if set, that the stream does not contain any cleanpoints. A cleanpoint is any point at which playback could begin without having seen the previous media objects. For streams that use key frames, the key frames would be the cleanpoints.

Resend Live Cleanpoints Flag

Specifies, if set, that when a stream is joined in mid-transmission, all information from the most recent cleanpoint up to the current time should be sent before normal streaming begins at the current time. The default behavior (when this flag is not set) is to send only the data starting at the current time. This flag should only be set for streams that are coming from a “live” source.

Reserved Flags

Specifies reserved flags. Shall be set to 0.

AverageTime Per Frame

Specifies the average time duration, measured in 100-nanosecond units, of each frame. This number should be rounded to the nearest integer. This field can optionally be set to 0 if the average time per frame is unknown or unimportant. It is recommended that this field be set for video.

Stream Number

Specifies the number of this stream. 0 is an invalid stream number (that is, other Header Objects use stream number 0 to refer to the entire file as a whole rather than to a specific media stream within the file). Valid values are between 1 and 127.

Stream Language ID Index

Specifies the language, if any, which the content of the stream uses or assumes. Refer to the **Language List Object** description for the details concerning how the **Stream Language ID Index** and **Language ID Index** fields should be used. Note that this is an index into the languages listed in the **Language List Object** rather than a language identifier.

Stream Name Count

Specifies how many **Stream Names** are present. Each stream name instance is potentially localized into a specific language. The **Language ID Index** field indicates the language in which the **Stream Name** has been written in Unicode characters. This shall be set to 0 if there are no **Stream Names**, in which case the **Stream Names** field should be omitted.

Payload Extension System Count

Specifies how many **Payload Extension Systems** there are for this stream. This shall be set to 0 if there are no **Payload Extension Systems**, in which case the **Payload Extension Systems** field should be omitted.

Stream Names

The structure of each **Stream Name** entry is shown in the following table.

Field name	Field type	Size (bits)
Language ID Index	WORD	16
Stream Name Length	WORD	16
Stream Name	WCHAR	varies

where the **Stream Name Length** field indicates the number of valid bytes that are found within the **Stream Name** field.

Payload Extension Systems

Payload extensions provide a way for content creators to specify kinds of data that will appear in the payload header for every payload from this stream. This system is used when stream properties must be conveyed at the media

object level. The **Replicated Data** bytes in the payload header will contain these properties in the order in which the **Payload Extension Systems** appear in this object. A **Payload Extension System** must appear in the **Extended Stream Properties Object** for each type of per-media-object properties that will appear with the payloads for this stream. For more information, see section 7.3.

Payload extension systems are described in the following table.

Field name	Field type	Size (bits)
Extension System ID	GUID	128
Extension Data Size	WORD	16
Extension System Info Length	DWORD	32
Extension System Info	BYTE	varies

The fields are defined as follows:

Extension System ID

Specifies a unique identifier for the extension system. Five standard GUIDs are defined in section 10.13, although custom extension system types can be defined as well.

Extension Data Size

Specifies the fixed size of the extension data for this system that will appear in the replicated data alongside every payload for this stream. If this extension system uses variable-size data, then this should be set to **0xffff**. Note, however, that replicated data length is limited to 255 bytes, which limits the total size of all extension systems for a particular stream.

Extension System Info Length

Specifies the length of the **Extension System Info** field. This field shall be set to 0 if there is no value in the **Extension System Info** field.

Extension System Info

Specifies additional information to describe this extension system (optional).

Stream Properties Object

Specifies an optional **Stream Properties Object**. For more information, see the beginning of this section.

4.2 Advanced Mutual Exclusion Object (optional, 0 to many)

The **Advanced Mutual Exclusion Object** identifies streams that have a mutual exclusion relationship to each other (in other words, only one of the streams within such a relationship can be streamed—the rest are ignored). There should be one instance of this object for each set of objects that contain a mutual exclusion relationship. The exclusion type is used so that implementations can allow user selection of common choices, such as language. This object must be used if any of the streams in the mutual exclusion relationship are hidden according to the manner described in section 8.2.2. For more details about scenarios in which you should use this object, see section 8.2.

The **Advanced Mutual Exclusion Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Exclusion Type	GUID	128
Stream Numbers Count	WORD	16
Stream Numbers	WORD	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Advanced Mutual Exclusion Object**. The value of this field shall be set to **ASF_Advanced_Mutual_Exclusion_Object**.

Object Size

Specifies the size, in bytes, of the **Advanced Mutual Exclusion Object**. Valid values are at least 42 bytes.

Exclusion Type

Specifies the nature of the mutual exclusion relationship. Use one of the GUIDs defined in section 10.11.

Stream Numbers Count

Specifies the number of streams listed in the **Stream Numbers** field. This value should be at least 2.

Stream Numbers

Specifies the list of mutually exclusive stream numbers. Valid values are between 1 and 127.

4.3 Group Mutual Exclusion Object (optional, 0 to many)

The **Group Mutual Exclusion Object** is used to describe mutual exclusion relationships between groups of streams. This object is organized in terms of "records," each containing one or more streams, where a stream in record N cannot coexist with a stream in record M for N != M (however, streams in the same record can coexist). This mutual exclusion object would be used typically for the purpose of language mutual exclusion, and a record would consist of all streams for a particular language.

The **Group Mutual Exclusion Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Exclusion Type	GUID	128
Record Count	WORD	16
Records	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Group Mutual Exclusion Object**. The value of this field shall be set to **ASF_Group_Mutual_Exclusion_Object**.

Object Size

Specifies the size, in bytes, of the **Group Mutual Exclusion Object**. Valid values are at least 42 bytes.

Exclusion Type

Specifies the nature of the mutual exclusion relationship. Use one of the GUIDs defined in section 10.11.

Record Count

Specifies the number of entries in the **Records** list. This value should be at least 2.

Records

The structure of each **Record** entry is shown in the following table.

Field name	Field type	Size (bits)
Stream Count	WORD	16
Stream Numbers	WORD	varies

The fields are defined as follows:

Stream Count

Specifies the number of streams in this record. Must be at least 1.

Stream Numbers

Specifies the stream numbers for this record. Valid values are between 1 and 127.

For an example regarding the use of the **Group Mutual Exclusion Object**, see section 7.2.

4.4 Stream Prioritization Object (optional, 0 or 1)

The **Stream Prioritization Object** indicates the author's intentions as to which streams should or should not be dropped in response to varying network congestion situations. There may be special cases where this preferential order may be ignored (for example, the user hits the "mute" button). Generally it is expected that implementations will try to honor the author's preference.

The priority of each stream is indicated by how early in the list that stream's stream number is listed (in other words, the list is ordered in terms of decreasing priority).

The **Mandatory flag** field shall be set if the author wants that stream kept "regardless". If this flag is not set, then that indicates that the stream should be dropped in response to network congestion situations. Non-mandatory streams must never be assigned a higher priority than mandatory streams.

The **Stream Prioritization Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Priority Records Count	WORD	16
Priority Records	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Stream Prioritization Object**. The value of this field shall be set to **ASF_Stream_Prioritization_Object**.

Object Size

Specifies the size, in bytes, of the **Stream Prioritization Object**. Valid values are at least 26 bytes.

Priority Records Count

Specifies the number of entries in the **Priority Records** list. This value should be at least 2.

Priority Records

Priority Records are listed in order of decreasing priority. The structure of each **Priority Record** entry is shown in the following table.

Field name	Field type	Size (bits)
Stream Number	WORD	16
Priority Flags	WORD	16
Mandatory Flag		1 (LSB)
Reserved Flags		15

The fields are defined as follows:

Stream Number

Specifies the stream number. Valid values are between 1 and 127.

Mandatory flag

Specifies, if set, that the stream is mandatory.

Reserved flags

Specifies reserved flags. Shall be set to 0.

For an example using the **Stream Prioritization Object**, see section 7.2.

4.5 Bandwidth Sharing Object (optional, 0 to many)

The **Bandwidth Sharing Object** indicates streams that share bandwidth in such a way that the maximum bandwidth of the set of streams is less than the sum of the maximum bandwidths of the individual streams. There should be one instance of this object for each set of objects that share bandwidth. Whether or not this object can be used meaningfully is content-dependent.

The **Bandwidth Sharing Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Sharing Type	GUID	128
Data Bitrate	DWORD	32
Buffer Size	DWORD	32
Stream Numbers Count	WORD	16
Stream Numbers	WORD	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Bandwidth Sharing Object**. The value of this field shall be set to **ASF_Bandwidth_Sharing_Object**.

Object Size

Specifies the size, in bytes, of the **Bandwidth Sharing Object**. Valid values are at least 50 bytes.

Sharing Type

Specifies the type of sharing relationship for this object. Two types are predefined: **ASF_Bandwidth_Sharing_Partial**, in which any number of the streams in the relationship may be streaming data at any given time; and **ASF_Bandwidth_Sharing_Exclusive**, in which only one of the streams in the relationship may be streaming data at any given time. For these GUID values, see section 10.12.

Data Bitrate

Specifies the leak rate *R*, in bits per second, of a leaky bucket that contains the data portion of all of the streams, excluding all ASF Data Packet overhead, without overflowing. The size of the leaky bucket is specified by the value of the **Buffer Size** field. This value can be less than the sum of all of the data bit rates in the **Extended Stream Properties Objects** for the streams contained in this bandwidth-sharing relationship. For more information about leaky buckets, see section 7.1.

Buffer Size

Specifies the size *B*, in bits, of the leaky bucket used in the **Data Bitrate** definition. This value can be less than the sum of all of the buffer sizes in the **Extended Stream Properties Objects** for the streams contained in this bandwidth-sharing relationship.

Stream Numbers Count

Specifies the number of entries in the **Stream Numbers** list. This value should be at least 2.

Stream Numbers

Specifies the list of digital media stream numbers in a bandwidth sharing relationship. Valid values are between 1 and 127.

4.6 Language List Object (optional, only 1)

The **Language List Object** contains an array of Unicode-based language IDs. All other header objects refer to languages through zero-based positions in this array.

The **Language List Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Language ID Records Count	WORD	16
Language ID Records	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Language List Object**. The value of this field shall be set to **ASF_Language_List_Object**.

Object Size

Specifies the size, in bytes, of the **Language List Object**. Valid values are at least 26 bytes.

Language ID Records Count

Specifies the number of entries in the **Language ID Records** list.

Language ID Records

The structure of each **Language ID Record** entry is shown in the following table.

Field name	Field type	Size (bits)
Language ID Length	BYTE	8
Language ID	WCHAR	varies

The fields are defined as follows:

Language ID Length

Specifies the size, in bytes, of the **Language ID** field.

Language ID

Specifies the **Language ID** string, identifying a language supported in the file. It is strongly recommended that the strings stored in the **Language List Object** be compliant with RFC-1766 for specifying languages.

Note that other objects refer to the **Language List Object** by means of their own **Language List ID Index** fields. The value in the **Language ID Index** field explicitly provides an index into the **Language ID Records** structure in order to identify a specific language. The first entry into this structure has an index value of 0 (zero). Index values that are greater than the number of entries within the **Language ID Records** structure are interpreted as signifying the default language for the system (on the Windows operating system, this would correspond to the LCID returned by IMultiLanguage->GetRfc1766FromLcid(GetUserDefaultLCID()).

The first entry (entry 0) in the **Language List Object** is the default language, and systems should exhibit a preference for that language when no other has been explicitly specified. Language applies both to objects that specify a language and to objects that do not have a provision for specifying language.

4.7 Metadata Object (optional, 0 or 1)

The **Metadata Object** permits authors to store stream-based metadata in a file.

This object supports the same types of metadata information as the **Extended Content Description Object** except that it also allows a stream number to be specified. For more information about what types of attributes belong in the **Metadata Object**, see section 7.4.

The **Metadata Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Description Records Count	WORD	16
Description Records	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Metadata Object**. The value of this field shall be set to **ASF_Metadata_Object**.

Object Size

Specifies the size, in bytes, of the **Metadata Object**. Valid values are larger than 26 bytes.

Description Records Count

Specifies the number of entries in the **Description Records** list.

Description Records

The structure of each **Description Record** entry is shown in the following table.

Field name	Field type	Size (bits)
Reserved (Must Be Zero)	WORD	16
Stream Number	WORD	16
Name Length	WORD	16
Data Type	WORD	16
Data Length	DWORD	32
Name	WCHAR	varies
Data	See below	

The fields are defined as follows:

Reserved (Must Be Zero)

This field must contain the value 0.

Stream Number

Specifies whether the entry applies to a specific digital media stream or whether it applies to the whole file. A value of 0 in this field indicates that it applies to the whole file; otherwise, the entry applies only to the indicated stream number and must be between 1 and 127.

Name Length

Specifies the number of valid bytes stored in the **Name** field. Valid values are even numbers.

Data Type

Specifies the type of information being stored. The following values are recognized.

Value type	Description
0x0000	Unicode string. The data consists of a sequence of Unicode characters.
0x0001	BYTE array. The type of data is implementation-specific.
0x0002	BOOL. The data is 2 bytes long and should be interpreted as a 16-bit unsigned integer. Only 0x0000 or 0x0001 are permitted values.
0x0003	DWORD. The data is 4 bytes long and should be interpreted as a 32-bit unsigned integer.
0x0004	QWORD. The data is 8 bytes long and should be interpreted as a 64-bit unsigned integer.
0x0005	WORD. The data is 2 bytes long and should be interpreted as a 16-bit unsigned integer.

Data Length

Specifies the length in bytes of the **Data** field. Valid values are less than 65536.

Name

Specifies the name that uniquely identifies the attribute being described. Names are case-sensitive.

Data

Specifies the actual metadata being stored. The **Data** field should be interpreted according to the value stored in the **Data Type** field.

4.8 Metadata Library Object (optional, 0 or 1)

The **Metadata Library Object** lets authors store stream-based, language-attributed, multiply defined, and large metadata attributes in a file.

This object supports the same types of metadata as the **Metadata Object**, as well as attributes with language IDs, attributes that are defined more than once, large attributes, and attributes with the GUID data type For more information about what types of attributes belong in the **Metadata Library Object**, see section 7.4.

The ASF **Metadata Library Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Description Records Count	WORD	16
Description Records	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Metadata Library Object**. The value of this field shall be set to **ASF_Metadata_Library_Object**.

Object Size

Specifies the size, in bytes, of the **Metadata Library Object**. Valid values are at least 26 bytes.

Description Records Count

Specifies the number of entries in the **Description Records** list.

Description Records

The structure of each **Description Record** entry is shown in the following table.

Field name	Field type	Size (bits)
Language List Index	WORD	16
Stream Number	WORD	16
Name Length	WORD	16
Data Type	WORD	16
Data Length	DWORD	32
Name	WCHAR	varies
Data	See below	varies

The fields are defined as follows:

Language List Index

Specifies the index into the **Language List Object** that identifies the language of this attribute. If there is no **Language List Object** present, this field must be zero.

Stream Number

Specifies whether the entry applies to a specific digital media stream or whether it applies to the whole file. A value of 0 in this field indicates that it applies to the whole file; otherwise, the entry applies only to the indicated stream number. Valid values are between 1 and 127.

Name Length

Specifies the number of valid bytes stored in the **Name** field. Valid values are even numbers.

Data Type

Specifies the type of information being stored. The following values are recognized.

Value type	Description
0x0000	Unicode string. The data consists of a sequence of Unicode characters.
0x0001	BYTE array. The type of the data is implementation-specific.
0x0002	BOOL. The data is 2 bytes long and should be interpreted as a 16-bit unsigned integer. Only 0x0000 or 0x0001 are permitted values.
0x0003	DWORD. The data is 4 bytes long and should be interpreted as a 32-bit unsigned integer.
0x0004	QWORD. The data is 8 bytes long and should be interpreted as a 64-bit unsigned integer.
0x0005	WORD. The data is 2 bytes long and should be interpreted as a 16-bit unsigned integer.
0x0006	GUID. The data is 16 bytes long and should be interpreted as a 128-bit GUID.

Data Length

Specifies the length, in bytes, of the **Data** field.

Name

Specifies the name that identifies the attribute being described.

Data

Specifies the actual metadata being stored. The **Data** field should be interpreted according to the value stored in the **Data Type** field.

4.9 Index Parameters Object (mandatory only if the Index Object is present in file, 0 or 1)

The **Index Parameters Object** supplies information about those streams that are actually indexed (there must be at least one stream in an index) by the **Index Object** and how they are being indexed. This object shall be present in the **Header Object** if there is an **Index Object** present in the file.

An **Index Specifier** is required for each stream that will be indexed by the **Index Object**. These specifiers must exactly match those in the **Index Object**.

The **Index Parameters Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Index Entry Time Interval	DWORD	32
Index Specifiers Count	WORD	16
Index Specifiers	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Index Parameters Object**. The value of this field shall be set to **ASF_Index_Parameters_Object**.

Object Size

Specifies the size, in bytes, of the **Index Parameters Object**. Valid values are at least 34 bytes.

Index Entry Time Interval

Specifies the time interval between index entries in milliseconds. This value cannot be 0.

Index Specifiers Count

Specifies the number of entries in the **Index Specifiers** list. Valid values are 1 and greater.

Index Specifiers

Index Specifiers are defined as follows:

Field name	Field type	Size (bits)
Stream Number	WORD	16
Index Type	WORD	16

The fields are defined as follows:

Stream Number

Specifies the stream number that the **Index Specifiers** refer to. Valid values are between 1 and 127.

Index Type

Specifies the type of index. Values are as follows: 1 = Nearest Past Data Packet, 2 = Nearest Past Media Object, and 3 = Nearest Past Cleanpoint. The Nearest Past Data Packet indexes point to the data packet whose presentation time is closest to the index entry time. The Nearest Past Object indexes point to the closest data packet containing an entire object or first fragment of an object. The Nearest Past Cleanpoint indexes point to the closest data packet containing an entire object (or first fragment of an object) that has the Cleanpoint Flag set. Nearest Past Cleanpoint is the most common type of index.

4.10 Media Object Index Parameters Object (mandatory only if media object index is present in file, 0 or 1)

The **Media Object Index Parameters Object** supplies information about those streams that actually indexed (there must be at least one stream in an index) by media objects. This object shall be present in the **Header Object** if there is a **Media Object Index Object** present in the file.

An **Index Specifier** is required for each stream that will be indexed by the **Media Object Index Object**. These specifiers must exactly match those in the **Media Object Index Object**.

The **Media Object Index Parameters Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Index Entry Count Interval	DWORD	32
Index Specifiers Count	WORD	16
Index Specifiers	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Media Object Index Parameters Object**. The value of this field shall be set to **ASF_Media_Object_Index_Parameters_Object**.

Object Size

Specifies the size, in bytes, of the **Media Object Index Parameters Object**. Valid values are at least 34 bytes.

Index Entry Count Interval

Specifies the interval between each index entry by the number of media objects. This value cannot be 0.

Index Specifiers Count

Specifies the number of entries in the **Index Specifiers** list. Valid values are 1 and greater.

Index Specifiers

The structure of each **Index Specifier** entry is shown in the following table.

Field name	Field type	Size (bits)
Stream Number	WORD	16
Index Type	WORD	16

The fields are defined as follows:

Stream Number

Specifies the stream number that the **Index Specifiers** refer to. Valid values are between 1 and 127.

Index Type

Specifies the type of index. Values are defined as follows: 1 = Nearest Past Data Packet, 2 = Nearest Past Media Object, 3 = Nearest Past Cleanpoint, 0xff = Frame Number Offset. For a video stream, the Nearest Past Media Object and Nearest Past Data Packet indexes point to the closest data packet containing an entire video frame or first fragment of a video frame; Nearest Past Cleanpoint indexes point to the closest data packet containing an entire video frame (or first fragment of a video frame) that is a key frame; and Frame Number Offset indicates how many more frames need to be read for the given stream, starting with the first frame in the packet pointed to by the index entry, in order to get to the requested frame. Nearest Past Media Object is the most common value. Because ASF payloads do not contain the full frame number, there is often a Frame Number Offset index alongside one of the other types of indexes to allow the user to identify the exact frame being seeked to.

4.11 Timecode Index Parameters Object (mandatory only if TIMECODE index is present in file, 0 or 1)

The **Timecode Index Parameters Object** supplies information about those streams that are actually indexed (there must be at least one stream in an index) by timecodes. All streams referred to in the **Timecode Index Parameters Object** must have timecode **Payload Extension**

Systems associated with them in the **Extended Stream Properties Object** (for more details, see section 7.30). This object shall be present in the **Header Object** if there is a **Timecode Index Object** present in the file.

An **Index Specifier** is required for each stream that will be indexed by the **Timecode Index Object**. These specifiers must exactly match those in the **Timecode Index Object**.

The **Timecode Index Parameters Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Index Entry Count Interval	DWORD	32
Index Specifiers Count	WORD	16
Index Specifiers	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Timecode Index Parameters Object**. The value of this field shall be set to **ASF_Timecode_Index_Parameters_Object**.

Object Size

Specifies the size, in bytes, of the **Timecode Index Parameters Object**. Valid values are at least 34 bytes.

Index Entry Count Interval

This value is ignored for the **Timecode Index Parameters Object**.

Index Specifiers Count

Specifies the number of entries in the **Index Specifiers** list. Valid values are 1 and greater.

Index Specifiers

The structure of each **Index Specifier** entry is shown in the following table.

Field name	Field type	Size (bits)
Stream Number	WORD	16
Index Type	WORD	16

The fields are defined as follows:

Stream Number

Specifies the stream number that the **Index Specifiers** refer to. Valid values are between 1 and 127.

Index Type

Specifies the type of index. Values are defined as follows: 2 = Nearest Past Media Object, 3 = Nearest Past Cleanpoint (1 is not a valid value). For a video stream, The Nearest Past Media Object indexes point to the closest data packet containing an entire video frame or the first fragment of a video frame, and the Nearest Past Cleanpoint indexes point to the closest data packet containing an entire video frame (or first fragment of a video frame) that is a key frame. Nearest Past Media Object is the most common value.

4.12 Compatibility Object (optional, only 1)

The Compatibility Object is reserved for future use. It is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128

Object Size	QWORD	64
Profile	BYTE	8
Mode	BYTE	8

Object ID

Specifies the GUID for the **Compatibility Object**. The value of this field shall be set to **ASF_Compatibility_Object**.

Object Size

Specifies the size, in bytes, of the **Compatibility Object**. Valid values are at least 28 bytes.

Profile

This field is reserved and should be set to 2.

Mode

This field is reserved and should be set to 1.

4.13 Advanced Content Encryption Object (optional, 0 or 1)

The **Advanced Content Encryption Object** lets authors protect content by using Next Generation Windows Media Digital Rights Management for Network Devices. It is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Content Encryption Records Count	WORD	16
Content Encryption Records	See Text	Varies

Object ID

Specifies the GUID for the Advanced Content Encryption Object. This field shall be set to ASF_Advanced_Content_Encryption_Object.

Object Size

Specifies the size of the Advanced Content Encryption Object. This includes the sum of 26 bytes plus the length of the content encryption records.

Content Encryption Records Count

Specifies the number of content encryption records that are contained in this object.

Content Encryption Records

Content Encryption Records are described as follows.

Field name	Field type	Size (bits)
System ID	GUID	128
System Version	DWORD	32
Encrypted Object Record Count	WORD	16
Encrypted Object Records	See text	varies
Data Size	DWORD	32
Data	BYTE	varies

System ID

Specifies the unique identifier for the content encryption system. The possible GUIDs are defined in section 10.6.

System Version

Specifies the version of the content encryption system.

Encrypted Object Record Count

Specifies the number of encrypted object records that follow.

Encrypted Object Records

Encrypted Object Records indicate what objects (for example, streams) in the ASF file a particular **Content Encryption Record** is associated with. **Encrypted Object Records** are described as follows:

Field name	Field type	Size (bits)
Encrypted Object ID Type	WORD	16
Encrypted Object ID Length	WORD	16
Encrypted Object ID	See Text	Varies

Encrypted Object ID Type

Specifies the type of data stored in the **Encrypted Object ID** field. The following values are recognized.

Value type	Description
0x0001	WORD. This value indicates that the **Encrypted Object ID** is an ASF stream number. The **Object ID Length** should be set to 0x0002 when this value type is set.

Encrypted Object ID Length

Specifies the size, in bytes, of the **Encrypted Object ID** field.

Encrypted Object ID

Specifies the ID of the encrypted object. The length of this field is determined by the value of the **Encrypted Object ID Length** field. When the **Encrypted Object ID Type** is set to 0x0001, this field specifies the ASF stream number that the Content Encryption Record is associated with. A value of 0 in this field indicates that it applies to the whole file; otherwise, the entry applies only to the indicated stream number.

Data Size

Specifies the size, in bytes, of the **Data** field.

Data

The content protection data for this **Content Encryption Record**.

5. ASF top-level Data Object

The **Data Object** contains all of the Data Packets for a file. These Data Packets are organized in terms of increasing send times. A Data Packet can contain interleaved data from several digital media streams. This data can consist of entire objects from one or more streams. Alternatively, it can consist of partial objects (fragmentation).

Capabilities provided within the interleave packet definition include:

- Single or multiple payload types per Data Packet
- Fixed-size Data Packets
- Error correction information (optional)
- Clock information (optional)
- Redundant sample information, such as presentation time stamp (optional)

The packet definition is the same as that in previous versions of this specification.

5.1 ASF Data Object (mandatory, one only)

The **Data Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
File ID	GUID	128
Total Data Packets	QWORD	64
Reserved	WORD	16
Data Packets	See section 5.2	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Data Object**. The value of this field shall be set to **ASF_Data_Object**.

Object Size

Specifies the size of the **Data Object**. Valid values are at least 50 bytes. Note that if the **Broadcast Flag** bit of the **Flags** field is set on the **File Properties Object**, then the **Object Size** field may be 0. This is a special case that indicates the size of the **Data Object** is unknown. It is not valid to have a value of 0 for the **Object Size** field if the **Broadcast Flag** is not set.

File ID

Specifies the unique identifier for this ASF file. The value of this field shall be changed every time the file is modified in any way. The value of this field shall be identical to the value of the **File ID** field of the **Header Object**.

Total Data Packets

Specifies the number of ASF Data Packet entries that exist within the **Data Object**. It must be equal to the **Data Packet Count** field in the **File Properties Object** (section 3.2). The value of this field is invalid if the **Broadcast Flag** field of the **File Properties Object** is set to 1.

Reserved

Specifies a reserved field. The value of this field shall set to 0x0101.

Data Packets

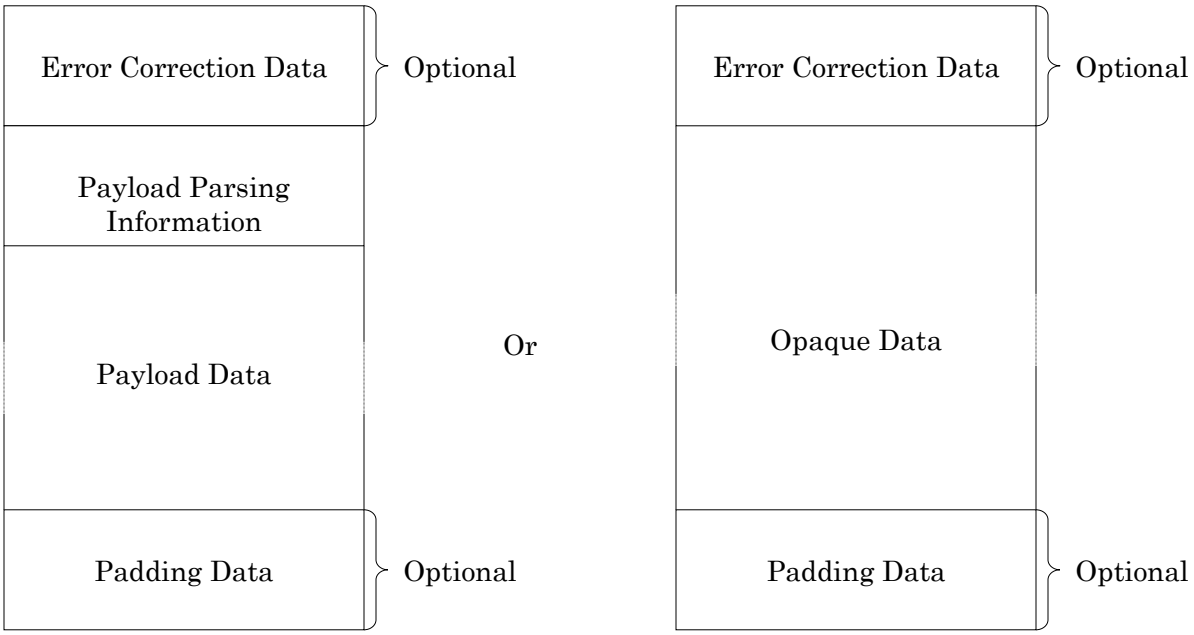
Specifies a list of Data Packets, as defined in section 5.2.

This structure is immediately followed by one or more Data Packets.

5.2 ASF data packet definition

In general, ASF media types logically consist of subelements that are referred to as media objects. What a media object happens to be in a given digital media stream is entirely stream-dependent (for example, it is a frame within a video stream). An ASF Data Packet is a conveniently sized grouping of complete or fragmented media objects from several digital media streams.

ASF Data Packets are structured as shown in the following diagram:



The following sections detail the content of each block shown in the previous diagram.

5.2.1 Error correction data

ASF Data Packets may start with some error correction data. This is signaled by the high order bit (**Error Correction Present** bit) of the first byte of the Data Packet being set. If this bit isn't set, a Data Packet starts with the payload data described in section 5.2.2. If this bit is set to 1, a Data Packet starts with the following error correction data.

Field name	Field type	Size (bits)
Error Correction Flags	BYTE	8
Error Correction Data Length		4 (LSB)
Opaque Data Present		1
Error Correction Length Type		2
Error Correction Present		1
Error Correction Data	BYTE	varies

The fields are defined as follows:

Error Correction Flags

The flags are stored in LSB order.

Error Correction Data Length (bits 0-3)

The value of this field is valid only if the value of the **Error Correction Length Type** field is 00. If the **Error Correction Length Type** field is 00, this field represents the size of the **Error Correction Data** field, in bytes. The value of this field should be set to 0010. If the **Error Correction Length Type** field is different than 00, this field shall be set to 0000.

Opaque Data Present (bit 4)

Specifies, if set, that the **Error Correction Data** field is followed by opaque data. The value of this field should be set to 0.

Error Correction Length Type (bits 5-6)

Specifies the number of bits used to code the size of the error correction data. The value of this field should be set to 00, indicating that the size of the error correction data is stored in the **Error Correction Data Length** field. Values other than 00 are reserved for future use.

Error Correction Present (bit 7)

Specifies, if set, that this Data Packet starts with error correction information. If set, the structure of the Data Packet is as described above. If it is not set, the Data Packet starts with the payload structure, as shown in section 5.2.2.

Error Correction Data

Specifies an array of bytes containing error correction data. The format of this data is based on the presence of the **Error Correction Object** (see section 3.9), and the value of the **Error Correction Type** field of the **Error Correction Object**. In all cases, the format of the first two bytes of **Error Correction Data** are defined as follows:

Field name	Field type	Size (bits)
First byte – Type	BYTE	8
Type		4 (LSB)
Number		4
Second byte – Cycle	BYTE	8

The fields are defined as follows:

First byte – Type

The type values are stored in LSB order.

Type (bits 0-3)

Specifies the type of the error correction data. The values are defined in the following table.

Value (bits)	Description
0000	The data is uncorrected. If there is no **Error Correction Object** in the **Header Object**, this is the value that shall be used for this field.
0001	The type of error correction is XOR data. This field can only be set to this value if there is an **Error Correction Object** in the **Header Object**.
0010	The type of error correction is parity data. This field can only be set to this value if there is an **Error Correction Object** in the **Header Object**.

Number (bits 4-7)

This number shall be set to 0 if the value of the **Type** field indicates that the data is uncorrected. If there is an **Error Correction Object** in the **Header Object**, this field shall be set to 1.

Second byte – Cycle

Specifies the cycle used by the error correction algorithm. This number shall be set to 0 if the value of the **Type** field indicates that the data is uncorrected.

5.2.2 Payload parsing information

If any error correction data is present, payload parsing information follows it. Payload parsing information has the following structure.

Field name	Field type	Size (bits)
Length Type Flags	BYTE	8
Multiple Payloads Present		1 (LSB)
Sequence Type		2
Padding Length Type		2
Packet Length Type		2
Error Correction Present		1
Property Flags	BYTE	8
Replicated Data Length Type		2 (LSB)
Offset Into Media Object Length Type		2
Media Object Number Length Type		2
Stream Number Length Type		2
Packet Length	BYTE, WORD or DWORD	0, 8, 16, 32
Sequence	BYTE, WORD or DWORD	0, 8, 16, 32
Padding Length	BYTE, WORD or DWORD	0, 8, 16, 32
Send Time	DWORD	32
Duration	WORD	16

The fields are defined as follows:

Length Type Flags

The flags are stored in LSB order.

Multiple Payloads Present (bit 0)

Specifies, if set, that the Data Packet contains multiple payloads. Whenever this flag is set, there will be data from multiple digital media stream samples in the Data Packet.

Sequence Type (bits 1-2)

Specifies the number of bits used to code the **Sequence** field. The values are defined in the following table.

Value type	Description
00	The **Sequence** field does not exist.
01	The **Sequence** field is coded using a BYTE.
10	The **Sequence** field is coded using a WORD.
11	The **Sequence** field is coded using a DWORD.

The value of field should be set to 00.

Padding Length Type (bits 3-4)

Specifies the number of bits used to code the **Padding Length** field. The values are defined in the following table.

Value type	Description
00	The **Padding Length** field does not exist.
01	The **Padding Length** field is coded using a BYTE.
10	The **Padding Length** field is coded using a WORD.
11	The **Padding Length** field is coded using a DWORD.

Packet Length Type (bits 5-6)

Specifies the number of bits used to code the **Packet Length** field. The values are defined in the following table.

Value type	Description
00	The **Packet Length** field does not exist.
01	The **Packet Length** field is coded using a BYTE.
10	The **Packet Length** field is coded using a WORD.
11	The **Packet Length** field is coded using a DWORD.

The value of field should be set to 00 when creating content.

Error Correction Present (bit 7)

Specifies, if set, that this Data Packet starts with error correction information. If not set, the structure of the Data Packet starts with the payload data as described above. If it is set, the Data Packet starts with the **Error Correction Data** structure, as shown in section 5.2.1 and this bit in the **Length Type Flags** field following error correction data in the Data Packet is not used and shall be ignored.

Property Flags

The flags are stored in LSB order.

Replicated Data Length Type (bits 0-1)

Specifies the number of bits used to code the **Replicated Data Length** field. The following values are defined:

Value type	Description
00	The **Replicated Data Length** field does not exist.
01	The **Replicated Data Length** field is coded using a BYTE.
10	The **Replicated Data Length** field is coded using a WORD.
11	The **Replicated Data Length** field is coded using a DWORD.

The value of this field shall be set to 01.

Offset Into Media Object Length Type (bits 2-3)

Specifies the number of bits used to code the **Offset Into Media Object Length** field. The following values are defined:

Value type	Description
00	The **Offset Into Media Object Length** field does not exist.
01	The **Offset Into Media Object Length** field is coded using a BYTE.
10	The **Offset Into Media Object Length** field is coded using a WORD.
11	The **Offset Into Media Object Length** field is coded using a DWORD.

The value of this field shall be set to 11.

Note that for compressed payloads (see sections 5.2.3.2 and 5.2.3.4) this field takes on a different meaning; instead—it is a presentation time.

Media Object Number Length Type (bits 4-5)

Specifies the number of bits used to code the **Media Object Number Length** field. The following values are defined:

Value type	Description
00	The **Media Object Number Length** field does not exist.
01	The **Media Object Number Length** field is coded using a BYTE.
10	The **Media Object Number Length** field is coded using a WORD.
11	The **Media Object Number Length** field is coded using a DWORD.

The value of this field shall be set to 01.

Stream Number Length Type (bits 6-7)

Specifies the number of bits used to code the **Stream Number Length** field. The value of this field shall be set to 01 to indicate that the **Stream Number Length** field is coded in a BYTE.

Packet Length

This field specifies the length of the data packet. This field exists only if the value of the **Packet Length Type** field is not 00. Whenever present, the **Packet Length** field can be coded using either a BYTE, a WORD, or a DWORD. This is specified by the value of the **Packet Length Type** field.

Sequence

This field is reserved for future use, and should be ignored. This field exists if and only if the value of the **Sequence Type** field is not 00. Whenever present, the **Sequence** field can be coded using either a BYTE, WORD or a DWORD. This is specified by the value of the **Sequence Type** field.

Padding Length

This field specifies the length of the padding at the end of a data packet. This field exists only if the value of the **Padding Length Type** field is not 00. Whenever present, the **Padding Length** field can be coded using either a BYTE, WORD, or DWORD and should contain the correct padding length. This is specified by the value of the **Padding Length Type** field.
In the multiple-payload case, where the amount of padding can be inferred by the packet length and the size of the final payload, this value can be zero even if there is some padding in the packet. For more details, see section 8.2.15.

Send Time

Specifies the send time of the Data Packet. The **Send Time** field must be coded using a DWORD and is specified in millisecond units.

Duration

Specifies the duration of the Data Packet. The **Duration** field is coded using a WORD and is specified in millisecond units.

5.2.3 Payload data

The actual digital media data follows the payload parsing information. This data can contain one or several payloads of data, depending on the value of the **Multiple Payloads Present** flag in the structure described in the previous section. If the **Multiple Payloads Present** flag is set to 1, the actual data is composed of multiple payloads, as described in section 5.2.3.3.

5.2.3.1 Single payload

Payload data in a Data Packet with a unique payload has the following structure.

Field name	Field type	Size (bits)
Payload	See next table	See next table

Payload

Field name	Field type	Size (bits)
Stream Number	BYTE	8
Media Object Number	BYTE, WORD, or DWORD	0, 8, 16, 32
Offset Into Media Object	BYTE, WORD, or DWORD	0, 8, 16, 32
Replicated Data Length	BYTE, WORD, or DWORD	0, 8, 16, 32
Replicated Data	BYTE	varies
Payload Data	BYTE	varies

The fields are defined as follows:

Stream Number

Specifies the stream number of the stream this data payload belongs to as well as whether the payload belongs to a media object that is a key frame.

Field name	Field type	Size (bits)
Stream Number and Key Frame Bit	BYTE	8
Stream Number		7 (LSB)

Key Frame Bit		1

Valid values for the low seven bits range from 1 to 127.

Media Object Number

Specifies the number of the media object this data payload belongs to. This field shall not be present if the **Media Object Number Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Media Object Number** field shall be coded using a BYTE. This is specified by the value of the **Media Object Number Length Type** field.

Offset Into Media Object

Specifies the byte offset in the media object this data payload belongs to. This field shall not be present if the **Offset Into Media Object Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Offset Into Media Object** field shall be coded using a DWORD. This is specified by the value of the **Offset Into Media Object Length Type** field.

Replicated Data Length

Specifies the size, in bytes, of the **Replicated Data** field. This field shall not be present if the **Replicated Data Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Replicated Data Length** field shall be coded using a BYTE. This is specified by the value of the **Replicated Data Length Type** field. If the value of this field is set to 1, the payload should be interpreted as a compressed payload, as described in section 5.2.3.2. Otherwise, valid values are 0 or values greater than or equal to 8.

Replicated Data

Specifies an array of replicated data. The number of bytes in this array is specified by the **Replicated Data Length** field. This data will be identical in value for all payloads for the same **Media Object**. Whenever present, this data always starts with a DWORD that contains the size, in bytes, of the **Media Object** this payload belongs to, immediately followed by a DWORD that contains the presentation time, in milliseconds, of the **Media Object** this payload belongs to. Following those two DWORDs is optional extension data for media samples. For details on how to use this data, see section 7.3.1.

Payload Data

Specifies an array containing the actual data for the payload. The number of bytes in this array can be calculated from the overall **Packet Length** field, and is equal to the **Packet Length** minus the packet header length, minus the payload header length (including **Replicated Data)**, minus the **Padding Length**.

5.2.3.2 Single payload, compressed payload data

This section shows the compressed payload interpretation of a single payload, as determined when the **Replicated Data Length** field of a single payload has a value of 1. A compressed payload contains one or more sub-payloads.

Compressed payloads can allow some space in the data packets to be saved. They can be used to represent a group of payloads only when all of the following conditions are met:

- All payloads are for the same stream.
- Either all payloads need to be marked as a key frame, or (more commonly) none of the frames needs to be marked as a key frame.
- Each payload represents an entire media object (rather than a fragment thereof).
- No payload has more than 256 bytes of data.
- No payload has more than the standard 8 bytes of replicated data.
- The values of the **Media Object Number** field in the **Paylod Data** stuctures of the payloads are consecutive.
- The presentation times of the payloads are spaced in constant intervals.

Each of the sub-payloads should be treated as an independent payload.

Payload

A **Payload** is described as follows:

Field name	Field type	Size (bits)
Stream Number	BYTE	8
Media Object Number	BYTE, WORD or DWORD	0, 8, 16, 32
Presentation Time	BYTE, WORD or DWORD	0, 8, 16, 32
Replicated Data Length	BYTE, WORD or DWORD	0, 8, 16, 32
Presentation Time Delta	BYTE	8
Sub-Payload Data	BYTE	varies

The fields are defined as follows:

Stream Number

Specifies the stream number of the stream that this data payload belongs to as well as whether the payload belongs to a media object that is a key frame.

Field name	Field type	Size (bits)
Stream Number and Key Frame Bit	BYTE	8
Stream Number		7 (LSB)
Key Frame Bit		1

Valid values for the low seven bits range from 1 to 127.

Media Object Number

Specifies the number of the media object that the first sub-payload belongs to. The media object number of each subsequent sub-payload is the **Media Object Number** field plus the sub-payload number. This field shall not be present if the **Media Object Number Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Media Object Number** field can be coded using either a BYTE, WORD or a DWORD. This is specified by the value of the **Media Object Number Length Type** field. For content created according to this specification, this field is coded using a BYTE.

Presentation Time

Specifies the presentation time, in milliseconds, of the media object that the first sub-payload belongs to. This field must be present; the **Offset Into Media Object Length Type** field of the payload parsing information structure must not be set to 00. The **Presentation Time** field can be coded using either a BYTE, WORD or a DWORD. This is specified by the value of the **Offset Into Media Object Length Type** field. For content created according to this specification, this field is coded using a DWORD.

Note that this field is used to express the **Offset Into Media Object** value for non-compressed payloads; however, for compressed payloads it is used to express the **Presentation Time** value.

Replicated Data Length

Specifies the size, in bytes, of the **Replicated Data** field. This field must be present; the **Replicated Data Length Type** field of the payload parsing information structure must not be set to 00. Whenever present, the **Replicated Data Length** field can be coded using either a BYTE, WORD or a DWORD. This is specified by the value of the **Replicated Data Length Type** field. For content created according to this specification, this field is coded using a BYTE. The **Replicated Data Length** field must contain the value 1 or the payload should be interpreted as a normal, non-compressed payload.

Presentation Time Delta

Specifies the presentation time delta, in milliseconds, to be applied to sub-payloads after the first. The presentation time of a media object in a sub-payload shall be interpreted as the **Presentation Time** field plus the **Presentation Time Delta** multiplied by the sub-payload number. This value is ignored if there is only one sub-payload.

Sub-Payload Data

Specifies an array containing the sub-payloads. The number of bytes in this array can be calculated from the overall **Packet Length** field.

Contains one or more sub-payloads, described as follows. Note that the number of sub-payloads is not explicitly specified; the last sub-payload should be detected by the end of the sub-payload data matching the end of the **Sub-Payload Data**, as indicated by the **Payload Data Length** field.

Field name	Field type	Size (bits)
Sub-Payload #0 Data Length	BYTE	8
Sub-Payload #0 Data	BYTE	varies
Sub-Payload #1 Data Length	BYTE	8
Sub-Payload #1 Data	BYTE	varies
...		varies

Each sub-payload should be interpreted as an independent payload, with properties described above.

5.2.3.3 Multiple payloads

Payload data in a Data Packet with multiple payloads has the following structure.

Field name		Field type	Size (bits)
Payload Flags		BYTE	8
	Number of Payloads		6 (LSB)
	Payload Length Type		2
Payloads		See below	

The fields are defined as follows:

Payload Flags

The flags are stored in LSB order.

Number of Payloads (bits 0-5)

Specifies the number of payloads contained in the **Payloads** field. This field must not contain the value 0.

Payload Length Type (bits 6-7)

Specifies the number of bits used to code the **Payload Length** field contained in each of the payloads of this packet. The values are defined in the following table.

Value type	Description
01	The **Payload Length** field is coded using a BYTE.
10	The **Payload Length** field is coded using a WORD.
11	The **Payload Length** field is coded using a DWORD.

The value of this field should be set to 10.

Payloads

This field contains an array of payloads. The number of entries in this array is specified by the **Number of Payloads** field. The **Payload** is commonly described as follows; however, if the **Replicated Data Length** field contains a value of 1, the payload should be interpreted as described in section 5.2.3.4:

Field name	Field type	Size (bits)
Stream Number	BYTE	8
Media Object Number	BYTE, WORD or DWORD	0, 8, 16, 32
Offset Into Media Object	BYTE, WORD or DWORD	0, 8, 16, 32
Replicated Data Length	BYTE, WORD or DWORD	0, 8, 16, 32
Replicated Data	BYTE	varies
Payload Length	BYTE, WORD or DWORD	8, 16, 32
Payload Data	BYTE	varies

The **Payload** fields are defined as follows:

Stream Number

Specifies the stream number of the stream this data payload belongs to as well as whether the payload belongs to a media object that is a key frame.

Field name	Field type	Size (bits)
Stream Number and Key Frame Bit	BYTE	8
Stream Number		7 (LSB)
Key Frame Bit		1

Valid values for the low seven bits range from 1 to 127.

Media Object Number

Specifies the number of the media object this data payload belongs to. This field shall not be present if the **Media Object Number Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Media Object Number** field can be coded using either a BYTE, a WORD, or a DWORD. This is specified by the value of the **Media Object Number Length Type** field. For content created according to this specification, this field is coded using a BYTE.

Offset Into Media Object

Specifies the byte offset in the media object this data payload belongs to. This field shall not be present if the **Offset Into Media Object Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Offset Into Media Object** field can be coded using either a BYTE, a WORD, or a DWORD. This is specified by the value of the **Offset Into Media Object Length Type** field. For content created according to this specification, this field is coded using a DWORD.

Replicated Data Length

Specifies the size, in bytes, of the **Replicated Data** field. This field shall not be present if the **Replicated Data Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Replicated Data Length** field can be coded using either a BYTE, a WORD, or a DWORD. This is specified by the value of the **Replicated Data Length Type** field. For content created according to this specification, this field is coded using a BYTE. If the value of this field is set to 1, the payload should be interpreted as a compressed payload, as described in section 5.2.3.4. Otherwise, valid values are 0 or values greater than or equal to 8.

Replicated Data

Specifies an array of replicated data. The number of bytes in this array is specified by the **Replicated Data Length** field. This data will be identical in value for all payloads for the same **Media Object**. Whenever present, this data always starts with a DWORD that contains the size, in bytes, of the **Media Object** this payload belongs to, immediately followed by a DWORD that contains the presentation time, in milliseconds, of the **Media Object** this payload belongs to. Following those two DWORDs is optional extension data for digital media samples. For details on how to use this data, see section 7.3.1.

Payload Length

Specifies the number of bytes in the **Payload Data** array. The value of this field must not be 0. The **Payload Length** field can be coded using either a BYTE, a WORD, or a DWORD. This is specified by the value of the **Payload Length Type** field. For content created according to this specification, this field is coded using a WORD.

Payload Data

Specifies an array containing the actual data for the payload.

5.2.3.4 Multiple payloads, compressed payload data

The following describes the structure of a compressed payload in a multiple payload packet, as determined when the **Replicated Data Length** field of a payload has a value of 1. A compressed payload contains one or more sub-payloads.

Compressed payloads can allow some space in the data packets to be saved. They can be used to represent a group of payloads only when all of the following conditions are met:

- All payloads are for the same stream.
- None of the payloads needs to be marked as a key frame.
- Each payload represents an entire media object (rather than a fragment thereof).
- No payload has more than 256 bytes of data.
- No payload has more than the standard 8 bytes of replicated data.
- The values of the **Media Object Number** field in the **Paylod Data** stuctures of the payloads are consecutive.
- The presentation times of the payloads are spaced in constant intervals.

Each of the sub-payloads should be treated as an independent payload.

Field name	Field type	Size (bits)
Payload Flags	BYTE	8
Number of Payloads		6 (LSB)
Payload Length Type		2
Compressed Payloads	See below	

The fields are defined as follows:

Payload Flags

The flags are stored in LSB order.

Number of Payloads (bits 0-5)

Specifies the number of payloads contained in the **Payloads** field. This field must not contain the value 0.

Payload Length Type (bits 6-7)

Specifies the number of bits used to code the **Payload Length** field contained in each of the payloads of this packet. The values are defined in the following table.

Value type	Description
01	The **Payload Length** field is coded using a BYTE.
10	The **Payload Length** field is coded using a WORD.
11	The **Payload Length** field is coded using a DWORD.

The value of this field should be set to 10.

Compressed Payloads

Field name	Field type	Size (bits)
Stream Number	BYTE	8
Media Object Number	BYTE, WORD or DWORD	0, 8, 16, 32
Presentation Time	BYTE, WORD or DWORD	0, 8, 16, 32
Replicated Data Length	BYTE, WORD or DWORD	0, 8, 16, 32
Presentation Time Delta	BYTE	8
Payload Length	BYTE, WORD or DWORD	8, 16, 32
Sub-Payload Data	BYTE	varies

The fields are defined as follows:

Stream Number

Specifies the stream number of the stream this data payload belongs to as well as whether the payload belongs to a media object that is a key frame.

Field name	Field type	Size (bits)
Stream Number and Key frame Bit	BYTE	8
Stream Number		7 (LSB)
Key frame Bit		1

Valid values for the low seven bits range from 1 to 127.

Media Object Number

Specifies the number of the media object that the first sub-payload belongs to. The media object number of each subsequent sub-payload is the **Media Object Number** field plus the sub-payload number. This field shall not be present if the **Media Object Number Length Type** field of the payload parsing information structure is set to 00. Whenever present, the **Media Object Number** field can be coded using either a BYTE, WORD or a DWORD. This is specified by the value of the **Media Object Number Length Type** field. For content created according to this specification, this field is coded using a BYTE.

Presentation Time

Specifies the presentation time, in milliseconds, of the media object that the first sub-payload belongs to. This field must be present; the **Offset Into Media Object Length Type** field of the payload parsing information structure must not be set to 00. The **Presentation Time** field can be coded using either a BYTE, WORD or a DWORD. This is specified by the value of the **Offset Into Media Object Length Type** field. For content created according to this specification, this field is coded using a DWORD.

Note that this field is used to express the **Offset Into Media Object** value for non-compressed payloads; however, for compressed payloads it is used to express the **Presentation Time** value.

Replicated Data Length

Specifies the size, in bytes, of the **Replicated Data** field. This field must be present; the **Replicated Data Length Type** field of the payload parsing information structure must not be set to 00. Whenever present, the **Replicated Data Length** field can be coded using either a BYTE, WORD or a DWORD. This is specified by the value of the **Replicated Data Length Type** field. For content created according to this specification, this field is coded using a BYTE. The **Replicated Data Length** field must contain the value 1, or else the payload should be interpreted as a normal, non-compressed payload.

Presentation Time Delta

Specifies the presentation time delta, in milliseconds, to be applied to sub-payloads after the first. The presentation time of a media object in a sub-payload shall be interpreted as the **Presentation Time** field plus the **Presentation Time Delta** multiplied by the sub-payload number. This value is ignored if there is only one sub-payload.

Payload Length

Specifies the number of bytes in the **Sub-Payload Data** array. The value of this field must not be 0. The **Payload Length** field can be coded using either a BYTE, a WORD, or a DWORD. This is specified by the value of the **Payload Length Type** field. For content created according to this specification, this field is coded using a WORD.

Sub-Payload Data

Specifies an array containing the sub-payloads.

Contains one or more sub-payloads, described as follows. Note that the number of sub-payloads is not explicitly specified; the last sub-payload should be detected by the end of the sub-payload data matching the end of the **Sub-Payload Data**, as indicated by the **Payload Length** field.

Field name	Field type	Size (bits)
Sub-Payload #0 Data Length	BYTE	8
Sub-Payload #0 Data	BYTE	varies
Sub-Payload #1 Data Length	BYTE	8
Sub-Payload #1 Data	BYTE	varies
...		varies

Each sub-payload should be interpreted as an independent payload, with properties described above.

5.2.4 Padding data

Following the payload data, an ASF Data Packet may contain padding data. Padding data in a Data Packet has the following structure.

Field name	Field type	Size (bits)
Padding Data	BYTE	varies

The fields are defined as follows:

Padding Data

Specifies an array of bytes containing padding data. The size of this array is specified by the **Padding Length** field, when present in the payload parsing information structure. If the **Padding Length** field is not present, the **Padding Data** array is empty. When present, these bytes should be set to 0.

6. ASF top-level index objects

This section describes the various types of index objects that can appear at the end of an ASF file. More than one of these can coexist. For guidelines concerning which type of index to use for maximum backward compatibility, see section 8.2.12.

6.1 ASF top-level Simple Index Object (optional but recommended when appropriate, 1 for each non-hidden video stream)

For each video stream in an ASF file, there should be one instance of the **Simple Index Object**. Additionally, the instances of the **Simple Index Object** shall be ordered by stream number.

Index entries in the **Simple Index Object** are in terms of **Presentation Times**. The corresponding **Packet Number** field values (of the **Index Entry**, see below) indicate the packet number of the ASF Data Packet with the closest past key frame. Note that for video streams that contain both key frames and non-key frames, the **Packet Number** field will always point to the closest past key frame.

For a description of how to determine which video streams should be indexed with the **Simple Index Object**, see section 8.2.12.

The **Simple Index Object** was defined in previous versions of this specification.

The **Simple Index Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
File ID	GUID	128
Index Entry Time Interval	QWORD	64
Maximum Packet Count	DWORD	32
Index Entries Count	DWORD	32

Index Entries	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Simple Index Object**. The value of this field shall be set to **ASF_Simple_Index_Object**.

Object Size

Specifies the size, in bytes, of the **Simple Index Object**. Valid values are at least 56 bytes.

File ID

Specifies the unique identifier for this ASF file. The value of this field should be changed every time the file is modified in any way. The value of this field may be set to 0 or set to be identical to the value of the **File ID** field of the **Data Object** and the **Header Object**. ASF parsers may safely ignore this value.

Index Entry Time Interval

Specifies the time interval between each index entry in 100-nanosecond units. The most common value is 10000000, to indicate that the index entries are in 1-second intervals, though other values can be used as well.

Maximum Packet Count

Specifies the maximum **Packet Count** value of all **Index Entries**.

Index Entries Count

Specifies the number of **Index Entries** structures contained in the next field.

Index Entries

The structure of each Index Entry is shown in the following table.

Field name	Field type	Size (bits)
Packet Number	DWORD	32
Packet Count	WORD	16

The fields are defined as follows:

Packet Number

Specifies the number of the Data Packet associated with this index entry. Note that for video streams that contain both key frames and non-key frames, this field will always point to the closest key frame prior to the time interval.

Packet Count

Specifies the number of Data Packets to send at this index entry. If a video key frame has been fragmented into two Data Packets, the value of this field will be equal to 2.

6.2 ASF top-level Index Object (optional but recommended when appropriate, 0 or 1)

This top-level ASF object supplies the necessary indexing information for an ASF file that contains more than just a plain script-audio-video combination. It includes stream-specific indexing information based on an adjustable index entry time interval. The index is designed to be broken into blocks to facilitate storage that is more space-efficient by using 32-bit offsets relative to a 64-bit base. That is, each index block has a full 64-bit offset in the block header that is added to the 32-bit offsets found in each index entry. If a file is larger than 2^32 bytes, then multiple index blocks can be used to fully index the entire large file while still keeping index entry offsets at 32 bits.

Indices into the **Index Object** are in terms of presentation times. The corresponding **Offset** field values of the **Index Entry** (see the following table) are byte offsets that, when combined with the

Block Position value of the **Index Block**, indicate the starting location in bytes of an ASF Data Packet relative to the start of the first ASF Data Packet in the file.

An offset value of 0xFFFFFFFF is used to indicate an invalid offset value. Invalid offsets signify that this particular index entry does not identify a valid indexible point. Invalid offsets may occur for the initial index entries of a digital media stream whose first ASF Data Packet has a non-zero send time. Invalid offsets may also occur in the case where a digital media stream has a large gap in the presentation time of successive objects.

The **Index Object** is not recommended for use with files where the Send Time of the first Data Packet within the **Data Object** has a Send Time value significantly greater than zero (otherwise the index itself will be sparse and inefficient).

Any ASF file containing an **Index Object** shall also contain an **Index Parameters Object** in its ASF Header.

The **Index Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Index Entry Time Interval	DWORD	32
Index Specifiers Count	WORD	16
Index Blocks Count	DWORD	32
Index Specifiers	See section 4.9	varies
Index Blocks	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Index Object**. The value of this field shall be set to **ASF_Index_Object**.

Object Size

Specifies the size, in bytes, of the **Index Object**. Valid values are at least 34 bytes.

Index Entry Time Interval

Specifies the time interval between each index entry in ms.

Index Specifiers Count

Specifies the number of **Index Specifiers** structures in this **Index Object**.

Index Blocks Count

Specifies the number of **Index Blocks** structures in this **Index Object**.

Index Specifiers

Index Specifiers are described in section 4.9. The ones in this field shall be exactly the same as those listed in the corresponding **Index Parameters Object**.

Index Blocks

The structure of individual **Index Block** entries is shown in the following table.

Field name	Field type	Size (bits)
Index Entry Count	DWORD	32
Block Positions	QWORD	varies
Index Entries	See below	varies

The fields are defined as follows:

Index Entry Count

Specifies the number of **Index Entries** in the block.

Block Positions

Specifies a list of byte offsets of the beginnings of the blocks relative to the beginning of the first Data Packet (for example, the beginning of the Data Object + 50 bytes). The number of entries in this list is specified by the value of the **Index Specifiers Count** field. The order of those byte offsets is tied to the order in which **Index Specifiers** are listed.

Index Entries

The structure of individual Index Entries is shown in the following table.

Field name	Field type	Size (bits)
Offsets	DWORD	varies

The field is defined as follows:

Offsets

The size of the **Offsets** field within each **Index Entry** structure is 32 bits multiplied by the value of the **Index Specifiers Count** field. For example, if the **Index Specifiers Count** is 3, then there are three 32-bit offsets in each **Index Entry**. **Index Entry** offsets are ordered according to the ordering specified by the **Index Parameters Object**, thereby permitting the same stream to be potentially indexed by multiple **Index Types** (that is, Nearest Past Cleanpoint, Nearest Past Object, Nearest Past Data Packet). An offset value of 0xffffffff indicates an invalid offset value; see the beginning of this section for information about invalid offset values.

6.3 ASF top-level Media Object Index Object (optional, 0 or 1)

This top-level ASF object supplies media object indexing information for the streams of an ASF file. It includes stream-specific indexing information based on an adjustable index entry media object count interval. This object can be used to index all the video frames or key frames in a video stream. The index is designed to be broken into blocks to facilitate storage that is more space-efficient by using 32-bit offsets relative to a 64-bit base. That is, each index block has a full 64-bit offset in the block header that is added to the 32-bit offset found in each index entry. If a file is larger than 2^32 bytes, then multiple index blocks can be used to fully index the entire large file while still keeping index entry offsets at 32 bits.

Indices into the **Media Object Index Object** are in terms of media object numbers, with the first frame for a given stream in the ASF file corresponding to entry 0 in the **Media Object Index Object**. The corresponding **Offset** field values of the **Index Entry** (see the following table) are byte offsets that, when combined with the **Block Position** value of the **Index Block**, indicate the starting location in bytes of an ASF Data Packet relative to the start of the first ASF Data Packet in the file.

Any ASF file containing a **Media Object Index Object** shall also contain a **Media Object Index Parameters Object** in its ASF Header.

The ASF **Media Object Index Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64

Index Entry Count Interval	DWORD	32
Index Specifiers Count	WORD	16
Index Blocks Count	DWORD	32
Index Specifiers	See section 4.10	varies
Index Blocks	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Media Object Index Object**. The value of this field shall be set to **ASF_Media_Object_Index_Object**.

Object Size

Specifies the size, in bytes, of the **Media Object Index Object**. Valid values are at least 34 bytes.

Index Entry Count Interval

Specifies the interval between each index entry in number of media objects.

Index Specifiers Count

Specifies the number of **Index Specifiers** structures in this **Media Object Index Object**.

Index Blocks Count

Specifies the number of **Index Block** structures in this **Media Object Index Object**.

Index Specifiers

Index Specifiers are described in Section 4.10. The ones in this field shall be exactly the same as those listed in the corresponding **Media Object Index Parameters Object**.

Index Blocks

The structure of each **Index Block** is shown in the following table.

Field name	Field type	Size (bits)
Index Entry Count	DWORD	32
Block Positions	QWORD	varies
Index Entries	See below	varies

The fields are defined as follows:

Index Entry Count

Specifies the number of **Index Entries** in the block.

Block Positions

Specifies a list of byte offsets of the beginning of the blocks relative to the beginning of the first Data Packet (for example, the beginning of the Data Object + 50 bytes). The number of entries in this list is specified by the value of the **Index Specifiers Count** field. The order of those byte offsets is tied to the order in which **Index Specifiers** are listed.

Index Entries

The structure of each **Index Entry** is shown in the following table.

Field name	Field type	Size (bits)
Offsets	DWORD	varies

The field is defined as follows:

Offsets

The size of the **Offsets** field within each **Index Entry** structure is 32 bits multiplied by the value of the **Index Specifiers Count** field. For example, if the **Index Specifiers Count** is 2, then there are two 32-bit offsets in each **Index Entry**. **Index Entry** offsets are ordered according to the ordering specified by the **Media Object**

Index Parameters Object, thereby permitting the same stream to be potentially indexed by multiple **Index Types** (that is, Nearest Past Media Object, or Nearest Past Cleanpoint).

6.4 ASF top-level Timecode Index Object (optional, 0 or 1)

This top-level ASF object supplies timecode indexing information for the streams of an ASF file. It includes stream-specific indexing information based on the timecodes found in the file. If the **Timecode Index Object** is used, it is recommended that timecodes be stored as a **Payload Extension System** on the appropriate stream (see section 7.3.2.1). It is also recommended that every timecode appearing in the ASF file have a corresponging index entry.

The index is designed to be broken into blocks to facilitate storage that is more space-efficient by using 32-bit offsets relative to a 64-bit base. That is, each index block has a full 64-bit offset in the block header that is added to the 32-bit offsets found in each index entry. If a file is larger than 2^32 bytes, then multiple index blocks can be used to fully index the entire large file while still keeping index entry offsets at 32 bits.

To locate an object with a particular timecode in an ASF file, one would typically look through the **Timecode Index Object** in blocks of the appropriate range and try to locate the nearest possible timecode. The corresponding **Offset** field values of the **Index Entry** (see the following table) are byte offsets that, when combined with the **Block Position** value of the **Index Block**, indicate the starting location in bytes of an ASF Data Packet relative to the start of the first ASF Data Packet in the file.

Any ASF file containing a **Timecode Index Object** shall also contain a **Timecode Index Parameters Object** in its ASF Header.

The **Timecode Index Object** is represented using the following structure.

Field name	Field type	Size (bits)
Object ID	GUID	128
Object Size	QWORD	64
Reserved	DWORD	32
Index Specifiers Count	WORD	16
Index Blocks Count	DWORD	32
Index Specifiers	See section 4.11	varies
Index Blocks	See below	varies

The fields are defined as follows:

Object ID

Specifies the GUID for the **Media Object Index Object**. The value of this field shall be set to **ASF_Timecode_Index_Object**.

Object Size

Specifies the size, in bytes, of the **Timecode Index Object**. Valid values are at least 34 bytes.

Reserved

Reserved, must be 1.

Index Specifiers Count

Specifies the number of **Index Specifiers** structures in this **Timecode Index Object**.

Index Blocks Count

Specifies the number of **Index Block** structures in this **Timecode Index Object**.

Index Specifiers

Index Specifiers are described in section 4.11. The ones in this field shall be exactly the same as those listed in the corresponding **Timecode Index Parameters Object**.

Index Blocks

The structure of indiviudal **Index Block** entries is shown in the following table.

Field name	Field type	Size (bits)
Index Entry Count	DWORD	32
Timecode Range	WORD	16
Block Positions	QWORD	varies
Index Entries	See below	varies

The fields are defined as follows:

Index Entry Count

Specifies the number of **Index Entries** in the block.

Timecode Range

Specifies the timecode range for this block. Subsequent blocks must contain range numbers greater than or equal to this one. For an explanation of ASF timecodes, see section 7.3.2.1.

Block Positions

Specifies a list of byte offsets of the beginning of the blocks relative to the beginning of the first Data Packet (for example, the beginning of the Data Object + 50 bytes). The number of entries in this list is specified by the value of the **Index Specifiers Count** field. The order of those byte offsets is tied to the order in which **Index Specifiers** are listed.

Index Entries

The structure of individual Index Entries is shown in the following table.

Field name	Field type	Size (bits)
Timecode	See below	32
Offsets	DWORD	varies

The fields are defined as follows:

Timecode

This is the 4-byte timecode for these entries. For an explanation of ASF timecodes, see section 7.3.2.1.

Offsets

The size of the **Offsets** field within each **Index Entry** structure is 32 bits multiplied by the value of the **Index Specifiers Count** field. For example, if the **Index Specifiers Count** is 2, then there are two 32-bit offsets in each **Index Entry**. **Index Entry** offsets are ordered according to the ordering specified by the **Media Object Index Parameters Object**, thereby permitting the same stream to be potentially indexed by multiple **Index Types** (that is, Nearest Past Media Object, or Nearest Past Cleanpoint).

7. ASF feature implementation guidelines

This section contains important information about how to use the various content features provided by the ASF file format. These guidelines should be followed for all reading and writing implementations.

7.1 Bit rate and the leaky bucket model

The ASF leaky bucket model is what makes ASF content streamable over a network, even when the streams do not maintain an exactly constant bit rate. The parameters of the leaky bucket are specified in the **Extended Stream Properties Object**, and the send times of the ASF Packets are used to keep the presentation in line with those parameters. For transmission of a stream over a fixed-rate channel, a leaky bucket that can contain the stream is necessary to smooth the traffic into the network at a constant rate.

The following discussion deals with a single-stream presentation for the sake of clarity and is used to make the simple generalization to a multi-stream presentation.

In the **Extended Stream Properties Objects**, the **Data Bitrate** field of a stream corresponds to the leak rate of a "leaky bucket" that contains the stream (which is a convenient metaphor for the rate at which bits can be sent out over a network). A leaky bucket is a buffer that empties at a constant positive leak rate, into which bits may be dumped at irregular times, in irregular amounts. A leaky bucket is said to contain a stream if bits from the stream are dumped into the bucket according to the values of the **Presentation Time** field without ever overflowing the bucket.

Each leaky bucket is specified by its leak rate *R* in bits per second, its size (or capacity) *B* in bits, and its initial fullness *F* in bits. In the **Extended Stream Properties Object**, these values correspond to the **Data Bitrate**, **Buffer Size**, and **Initial Buffer Fullness** fields respectively. The initial fullness of the leaky bucket is the fullness of the buffer at the instant before the first bit in the stream in dumped in (in most implementations, this value is 0, but other values are valid as well).

The **Send Time** value in the ASF packet is an indication of when the payload data inside the packet would be leaked according to the leak rate R. This, of course, has to be no later than the **Presentation Time** of the payload (otherwise the data arrives too late to present). Due to the fact that the instantaneous bit rate as determined by **Presentation Time** values fluctuates around R, though, when the buffer is not completely full, **Send Time** will have to be earlier than the **Presentation Time** to ensure that when the buffer does get close to being full, every payload will still be leaked before or at its **Presentation Time**. Therefore, the **Send Time** values should be seen following the leak rate R relatively constantly, even if the **Presentation Time** values in the presentation are not. Another way to think of it is that the **Send Time** values allow for a head start so that the objects will arrive at the destination before their **Presentation Time** values, even when the **Presentation Time** values are advancing at an instantaneous rate that is higher than R.

Two other related parameters are of interest. The delay *D* is the time it takes for a full bucket to empty, and is equal to B/R. The preroll *T* is the time it takes for a full bucket to reach the initial bucket fullness F, and is equal to (B-F)/R. This is the time that the decoder buffer must build up before playout can begin (the head start needed to ensure that all payloads get leaked at or before their **Presentation Time**).

There is no unique, or even minimal, leaky bucket to contain a given stream. A given stream can be contained by a large bucket with a small leak rate, or a small bucket with a large leak rate. Typically, though, the parameters R, B, and F characterize a reasonable bit rate and buffer size scenario.

A group of streams with parameters Ri, Bi, and Fi can always be contained by a leaky bucket with parameters Σri, Σbi, and Σfi. These are the default parameters for such a group. However, it may be possible to contain the group with a smaller bucket, or at a smaller leak rate. For example, two

variable-rate streams may be coordinated such that together they form a stream that requires a smaller value of R and B than the sum of those values for the constituent streams. Hence, for some groups of streams, it may be desirable to override the default parameters. In ASF, this is possible by using the **Bandwidth Sharing Object**.

7.2 Stream selection process

This section provides a sample stream selection scenario that can be solved by using some of the stream selection objects used in the ASF file format specification. It also describes the stream selection algorithm that should be used when reading an ASF file.

7.2.1 Description of sample content

The content being created is meant to be presented in two different video windows and has an audio track.

The audio track is meant to be played in either French or English.

The first video window is meant to render a video stream showing the main content of the presentation. Whenever the content being played uses the French audio stream, this video window is to also receive and render the content of an additional English closed-caption stream.

The second video window is optional and is to be rendered separately, using a different set of codecs and contains only value-add visual information that was captured from a presentation similar to a Microsoft® PowerPoint® presentation. This video comes in French and English versions.

The video streams have also been authored to allow playback over slow-speed lines by using mutually exclusive streams compressed at different bit rates.

7.2.2 Content authoring

The following table describes the set of streams needed to build the content described previously, as well as their properties:

Stream	Type	Language	Bit rate
#1	Audio – Base Layer	English	10 Kbps
#2	Audio – Base Layer	French	10 Kbps
#3	Main Video – Mutually exclusive with #4 and #5	None	10 Kbps
#4	Main Video – Mutually exclusive with #3 and #5	None	20 Kbps
#5	Main Video – Mutually exclusive with #3 and #4	None	30 Kbps
#6	Closed Caption Script Stream	English	1 Kbps
#7	Extra Video – Mutually exclusive with #8	English	5 Kbps
#8	Extra Video – Mutually exclusive with #7	English	10 Kbps
#9	Extra Video – Mutually exclusive with #10	French	5 Kbps
#10	Extra Video – Mutually exclusive with #9	French	10 Kbps

(Because not all legacy ASF reading implementations support such a complex scenario, the author should choose a default presentation for those implementations. Assuming English is the desired language on legacy systems, the "backward compatible" presentation would likely consist of streams #1, #3, #4, #5 and #6. These streams should be represented in the ASF Header by **Stream Properties** Objects that are separate from those streams' **Extended Stream Properties Objects;** every other stream should be represented only by an **Extended Stream Properties Object** with a **Stream Properties Object** embedded. For more information, see section 8.)

The author uses a **Group Mutual Exclusion Object** to indicate that choosing one language for the presentation will turn off the streams in the other language. There should be a record for all English-specific streams and one for all French-specific streams. The following table shows how the object would look.

Field name	Field size (bytes)	Field value
Object ID	16	ASF_Advanced_Mutual_Exclusion_Object
Object Size	8	60
Exclusion Type	16	ASF_Mutex_Language
Record Count	2	2
Stream Number Count	2	3
Stream Numbers	6	1, 7, 8
Stream Number Count	2	4
Stream Numbers	8	2, 9, 10

Note that although stream #6 was designated as an English-language stream, it was not included in the language **Group Mutual Exclusion Object**. This is because this stream can appear with either presentation. There would typically be some logic to determine whether captioning is or is not desired in the presentation; this logic is outside the scope of the stream selection algorithm described in this section.

The author also wants to specify some default stream selection behavior to handle congestion/packet loss whenever bandwidth over the wire becomes scarce and streams might have to be dropped from the presentation. In this specific case, the author would like to make sure that at least the lowest bit rate audio gets to the user's computer, and then some form of video of the main video output and closed captioning, and then higher quality audio (which is absent from this example), and then higher quality video of the main video output, and finally, the caption stream and the extra video streams, if possible at all. To specify this, the author creates a stream priority list by using the **Stream Prioritization Object**. This stream priority list then contains the following ordered list of streams: #1, #2, #3, #4, #5, #6, #7, #9, #8, #10.

Finally, the author describes the bit rate mutual exclusion relationship between the video streams by using a set of mutual exclusion objects, also based on bit rate in this example. The first object is a **Mutual Exclusion Object** used to describe the mutual exclusion based on bit rate between video streams #3, #4, and #5, because, as mentioned in the parenthetical above, all of these streams are backward-compatible. The second object is an **Advanced Mutual Exclusion Object** used to describe the mutual exclusion based on bit rate between video streams #7 and #8. The last object is an **Advanced Mutual Exclusion Object** used to describe the mutual exclusion based on bit rate between video streams #9 and #10. The **Advanced Mutual Exclusion Object**

should be used because, as mentioned in the parenthetical above, these streams are not backward compatible. For a more detailed discussion on backward compatibility, see section 8.

7.2.3 Exercise of the stream selection process

This section describes how the stream selection process occurs on the client computer. The term "application" has broad meanings. It could mean a custom application that natively understands the ASF file format, or an application such as Windows Media® Player, using the file format and streaming services provided by the Windows Media Format SDK.

First, the client application builds a list *S* of all the streams available in the authored content. In the case described above, this list is initialized as follows:

$S = \{\#1, \#2, \#3, \#4, \#5, \#6, \#7, \#8, \#9, \#10\}$

The next step for the application is to apply all the mutual exclusion constraints that are not based on the bit rate to the main list *S* of streams. In this example, that means the language mutual exclusion relationship. After the language of the presentation is selected by the user or by using automatic means, the languages listed in the **Group Mutual Exclusion Object** records that do not match the selection are removed from the main list *S* of streams. Assuming that French was selected as the presentation language; the main list *S* of streams now contains the following items:

$S = \{\#2, \#3, \#4, \#5, \#6, \#9, \#10\}$

To complete the stream selection process, the application applies the mutual exclusion constraints based on bit rate to generate an ordered list of streams *B* from which the application will chose streams to render, up to the current available bit rate. This step uses the following algorithm:

1. Create an empty list of streams *B*.
2. Take the next stream *s* in the list contained in the **Stream Prioritization Object** list of streams. If the **Stream Prioritization Object** is absent, reading implementations should be able to generate their own priority list, in which lower bit rates precede higher bit rates and higher-priority media types (like audio) precede lower-priority media types.
3. If *s* is not contained in the main list *S* of streams, go to step 2; otherwise proceed to step 4.
4. If *s* is in a **Bitrate Mutual Exclusion Object** with any stream *t* in *B*, remove stream *t* from *B* and add *s* to *B*; otherwise, simply add *s* to *B*.
5. Go to step 2 until there are no more streams in the **Stream Prioritization Object** list (or the implementation-generated priority list) of streams to look at, or end the algorithm if the current available bit rate has been reached.

The following table shows the content of *B*, as well as its cumulative bit rate for the list of streams contained in the **Stream Prioritization Object**, until the end of the **Stream Prioritization** list. The number in the column labeled "Stream" indicates which stream *s* is being considered at step #2 above. Again, the preceding algorithm will stop and use the current set of streams in *B* if the cumulative bit rates of the streams in *B* has reached the current available bit rate over the network:

Stream *	B	Bit rate
#1	{}	0 Kbps

#2	{#2}	10 Kbps
#3	{#2, #3}	20 Kbps
#4	{#2, #4}	30 Kbps
#5	{#2, #5}	40 Kbps
#6	{#2, #5, #6}	41 Kbps
#7	{#2, #5, #6}	41 Kbps
#9	{#2, #5, #6, #9}	46 Kbps
#8	{#2, #5, #6, #9}	46 Kbps
#10	{#2, #5, #6, #10}	51 Kbps

* Streams are considered in the order in which they were listed in the **Stream Prioritization Object**.

7.3 Payload extension systems

The **Replicated Data** in an ASF payload can be used to convey per-media-object properties. There are a few standard properties, and custom implementation-specific properties can be defined as well.

Note that **Payload Extension Systems** are sometimes referred to by the term "**Data Unit Extension Systems**" in other contexts. The two terms are completely interchangeable; in this specification we use the former term.

7.3.1 Parsing the Replicated Data

For every payload of a particular stream, the **Replicated Data** shall contain data for the same **Payload Extension Systems**, in the same order. The GUIDs identifying which properties will be present on the stream's payloads are specified in the **Extended Stream Properties Object** for that stream in the **Payload Extension Systems** (see section 4.1 for the format of this data). The order in which they appear in the **Extended Stream Properties Object** is the order in which they will appear in the **Replicated Data.**

Compressed payloads (see sections 5.2.3.2 and 5.2.3.4) have 1 byte of **Replicated Data** and therefore by definition do not contain **Payload Extension System** data. For all other payloads, **Replicated Data** shall be at least 8 bytes long. The first 4 bytes of data shall contain the size of the media object that the payload belongs to. The next 4 bytes of data shall contain the **Presentation Time** for the media object that the payload belongs to.

The remaining bytes in the **Replicated Data**, when present, should be parsed as follows: Step through the payload extension systems for that stream in order. For each **Payload Extension System**, if the **Extension Data Size** (specified in the **Extended Stream Properties Object**) contains a value N not equal to 0xffff, the next N bytes should be interpreted as the value for that property for that media object. If the **Extension Data Size** is equal to 0xffff, then this is a variable-size **Payload Extension System**, and the next 2 bytes shall be interpreted as the size of this particular property for this media object, and the property value will be in as many bytes following the size. (The size does not include the 2 bytes that the size number itself takes up.)

For example, consider a stream that has two payload extension systems associated with it, the first of which has size 0xffff (to indicate that its extensions are variable-sized) and the second of which has a fixed size of 7 bytes. The replicated data would be formatted as illustrated in the following diagram:

Media Object Size (4 bytes)	Pres Time (4 bytes)	Ext Sys Len (2B)	Data for first extension system (size contained in previous field)	Data for second extension system (7 bytes)

Because **Replicated Data Length** is represented by a BYTE, content authoring implementations need to be aware that all **Payload Extension System** data plus the 8 required bytes must fit into 256 bytes.

7.3.2 Standard Payload Extension Systems

A few GUIDs define standard **Payload Extension Systems**, and the GUID values are defined in section 10.13.

7.3.2.1 ASF_Payload_Extension_System_Timecode

The following table shows how timecodes are specified for media objects They are 14-byte values as described in the following table.

Field name	Field type	Size (bits)
Timecode Range	WORD	16
Timecode	DWORD	32
User-defined bits	DWORD	32
Reserved (must be 0)	DWORD	32

Timecode Range

The **Timecode Range** is used as a "source ID" field; for example, if a stream consists of some data sourced from one tape followed by some other data sourced from a different tape, the former segment's timecodes would have 0 as their **Timecode Range** value, while the latter segment's timecodes would have 1 as their **Timecode Range** value. Within a certain **Timecode Range**, the **Timecode** values must be strictly increasing.

If a stream has multiple timecode ranges (that is, it came from multiple sources), the timecode ranges that are assigned to represent the various sources must be strictly increasing and consecutive. The first media object for a stream shall have a **Timecode Range** value of 0.

Timecode

This is the value that shall be used when creating a **Timecode Index Object** for this content. This field contains the timecode for this media object. The recommended format for timecodes should be (from MSB to LSB): Hours:Minutes:Seconds:Frames, with each value represented in one byte.

User-defined bits

The meaning of these bytes is up to the implementation.

Reserved

Must be 0.

7.3.2.2 ASF_Payload_Extension_System_File_Name

This system is useful for **File Transfer** streams because one stream can contain data from numerous files. This shall be a nul-terminated WCHAR string.

7.3.2.3 ASF_Payload_Extension_System_Content_Type

This indicates the type of interlacing in the content. This is a 1-byte value with zero or more of the bits set as described in the following table.

Field name	Field type	Size (bits)
Content Type Flags	BYTE	8
Reserved		4 (LSB)
Interlaced Repeat First Field		1
Interlaced Bottom Field First		1
Interlaced Top Field First		1
Interlaced Content		1

Interlaced Repeat First Field

When set in conjunction with the **Interlaced Bottom Field First** flag or the **Interlaced Top Field First** flag, this flag indicates that the first field in the frame (as specified by the field order flag) should be repeated when rendering.

Interlaced Bottom Field First

When set in conjunction with the **Interlaced Content** flag, this flag indicates that the lower field occurs first in time. If this flag is set, the **Interlaced Top Field First** flag must not be set.

Interlaced Top Field First

When set in conjunction with the **Interlaced Content** flag, this flag indicates that the upper field occurs first in time. If this flag is set, the **Interlaced Bottom Field First** flag must not be set.

Interlaced Content

When set, this flag indicates that the sample contains an interlaced frame consisting of two interleaved fields. If this flag is not set, the frame is progressive. If this flag is set in conjunction with the **Interlaced Bottom Field First** flag or the **Interlaced Top Field First** flag, the other flag indicates the field order of the fields. If neither of the field order flags is set, then the field order is unknown and should be assumed to be top field first.

7.3.2.4 ASF_Payload_Extension_System_Pixel_Aspect_Ratio

This field indicates the media object's pixel aspect ratio because it can vary between frames in the same stream. This is a 2-byte value as described in the following table.

Field name	Field type	Size (bits)
Pixel Aspect Ratio: "x"	BYTE	8
Pixel Aspect Ratio: "y"	BYTE	8

7.3.2.5 ASF_Payload_Extension_System_Sample_Duration

This indicates the duration, in milliseconds, of the media object. This is a 2-byte value.

7.3.2.6 ASF_Payload_Extension_System_Encryption_Sample_ID

This extension system is used when encrypting samples for the purposes of protecting the content over a link.

This is a variable-length piece of data that will appear on at least the first payload of every sample in the content when this extension system is being used. It may not appear on subsequent payloads of a frame.

The length of this extension will be less than or equal to 8 bytes, and it is a network-byte-ordered sample ID that should increment for each frame in the content. ASF parsing implementations that wish to consume this value should be aware that this value may have had its leading zeroes stripped in order to optimize for space; therefore, if this extension appears to be fewer than 8 bytes long for some payload, the parser should fill in the leading zeroes before interpreting this number.

7.3.2.7 ASF_Payload_Extension_System_Degradable_JPEG

This extension system is used with degradable JPEG image streams (9.4.2). The Windows Media® Format SDK appends this data to each sample. The format of the data is shown in the following table.

Field name	Field type	Size (bits)
Width	WORD	16
Height	WORD	16
Restart Interval	WORD	16
Restart Count	WORD	16
First Interval	WORD	16
First Total Size	WORD	16
First Offset To End	WORD	16
Last Interval	WORD	16
Last Total Size	WORD	16
Last Offset to Start	WORD	16

Width

Width of the entire image

Height

Height of the entire image

Restart Interval

The restart interval for the entire image.

Restart Count

The number of restarts in the image.

First Interval

The interval number at the start of the packet data.

First Total Size

The size of the entire interval that is specified by First Interval.

First Offset To End

The offset from the start of the payload to the end of the interval.

Last Interval

The interval number at the end of the packet data.

Last Total Size

The size of the entire interval that is specified by Last Interval.

Last Offset To Start

The offset from the end of the payload to the start of the interval.

Note: The Format SDK does not include the GUID for this extension system in the **Header Extension Object**.

In addition, GUIDs for custom properties can be defined by implementations.

7.4 Metadata

ASF provides five objects for storing metadata in an ASF file. To determine which object should be used for a particular metadata attribute, choose the first object in the following list that can support that attribute:

- **Content Description Object**. This object stores non-language-specific, non-stream-specific WCHAR string values for the following attributes: Title, Author, Copyright, Description, and Rating. Values are limited to 65535 bytes in length.

- **Content Branding Object**. This object stores non-language-specific, non-stream-specific values for the following attributes: Banner Image Type, Banner Image Data, Banner Image URL, and Copyright URL.

- **Extended Content Description Object**. This object stores non-language-specific, non-stream-specific attributes of any name for the following types: WCHAR strings, BYTE arrays, Boolean values, DWORDs, QWORDs, or WORDs. Values are limited to 65535 bytes in length.

- **Metadata Object**. This object stores non-language-specific attributes for any name. It allows data to be stored in the same types as the **Extended Content Description Object**. Attributes can be stream-specific. Values are limited to 65535 bytes in length.

- **Metadata Library Object**. This object stores attributes for any name. It allows data to be stored in the same types as the **Extended Content Description Object** as well as a GUID data type. Attributes can be stream-specific and are associated with a particular language. Values can exceed the 65535-byte limit imposed by the other attribute-storage objects because value length is stored in a DWORD.

7.5 Pixel aspect ratio

Video that is used as a source for video streams in an ASF file may have non-square pixels. To reproduce such video frames on a computer monitor or other device, the aspect ratio of the pixels must be known.

There are two methods of storing pixel aspect ration information in an ASF file. The first option is to identify the pixel aspect ratio by including the AspectRatioX and AspectRatioY metadata attributes in the **Metadata Object**. This option is approprate only if the pixel aspect ratio is constant for the duration of the stream.

The second option is to attach the pixel aspect ratio of each sample as a payload extension. For more information about the second option, see section 7.3.2.4.

Parsing applications use this information to determine whether any image transformation needs to be performed on video samples to maintain the original picture aspect ratio on the target device. If a file contains both the metadata attributes and the payload extensions, the parsing application should use the payload extensions.

8. Content reach guidelines

8.1 How to use this section

When creating ASF content, it is important to do so with emphasis toward making it as widely distributable as possible. This section provides guidelines about how to author content for compatibility with the maximum number of ASF reading implementations that currently exist.

It is important to note that unlike the rest of this specification, this section does not cover what constitutes valid ASF. Implementations that read ASF content must not assume that the guidelines listed have been followed; however, implementations that write ASF content must follow them.

8.2 Compatibility issues

This section considers the following implementations of ASF readers in its discussion of content reach:

- Windows Media Format SDK (and all software built on top of it, including Windows Media Player 7 and later): version 7 up to and including 9 Series.
- Windows Media Services: versions up to and including 9 Series.
- Windows Media Player 6.4 (**Note:** Windows Media Player 6.4 is not built on top of the Windows Media Format SDK.)

8.2.1 Header extension object and custom header objects

8.2.1.1 Guideline

All header objects except for those listed in section 3 should be written inside the **Header Extension Object**, as described in section 3.4.

8.2.1.2 ASF objects affected

This guideline pertains to the ASF **Header Object**.

8.2.1.3 Readers affected

All of the Microsoft products covered in this section can play content whether or not this guideline is followed. However, unless the guideline is followed, editing the content's metadata in products build on any versions of the Windows Media Format SDK before 9 Series will result in permanently corrupted content.

8.2.2 Handling complex stream configurations

8.2.2.1 Guideline

There are cases in which it is desirable to keep all products prior to but not including Windows Media Player 9 Series from recognizing a particular stream. For some scenarios, see sections 8.2.3 through 8.2.9.

This is done by embedding the **Stream Properties Object** for that stream inside the **Extended Stream Properties Object** for the same stream (see section 4.1). There must be no separate **Stream Properties Object** for that stream.

For the ramifications of "hiding" a stream this way, see section 8.2.7.

8.2.2.2 ASF objects affected

This guideline pertains to the **Stream Properties Object** and the **Extended Stream Properties Object**.

8.2.2.3 Readers affected

A stream that is declared this way in the ASF header will be ignored by all of the implementations covered in this section prior to but not including the Windows Media 9 Series technologies.

Note the caveat regarding Windows Media Services 4.1 in section 8.2.7 ("Unknown stream IDs in the payloads").

8.2.3 Media types other than audio, video, image and script

8.2.3.1 Guideline

Media types other than audio, video, image and script can be found in ASF files. However, streams using these media types shall be hidden according to section 8.2.2.

8.2.3.2 ASF objects affected

This guideline pertains to the **Extended Stream Properties Object**.

8.2.3.3 Readers affected

Implementations of Windows Media technologies prior to Windows Media 9 Series do not support media types other than audio, video, and script. Following this guideline will hide the streams with the other media types from these readers.

8.2.4 Bitrate mutually exclusive video streams, different frame sizes

8.2.4.1 Guideline

When creating content with bit rate mutually-exclusive video streams in which the streams have different frame sizes, all but one of the streams in the mutual exclusion relationship shall be hidden

from earlier implementations according to the manner described in section 8.2.2, and the **Advanced Mutual Exclusion Object** shall be used to describe the mutual exclusion.

Note, however, when all video streams have the same frame size, the **Bitrate Mutual Exclusion Object** shall be used, and no streams need to be hidden.

8.2.4.2 ASF objects affected

This guideline pertains to the **Advanced Mutual Exclusion Object** and the **Extended Stream Properties Object**.

8.2.4.3 Readers affected

Bit rate mutual exclusion among video streams with different frame sizes is supported only by the components of Windows Media 9 Series. Following this guideline will hide the mutual exclusion relationship as well as all but one of the streams from previous versions of Windows Media technologies.

8.2.5 Bitrate mutually exclusive non-video streams

8.2.5.1 Guideline

When creating content with bit rate mutually-exclusive streams of a media type other than video (the most common such scenario would be bit rate mutually-exclusive audio), all but one of the streams in the mutual exclusion relationship shall be hidden from earlier implementations according to the manner described in section 8.2.2, and the **Advanced Mutual Exclusion Object** shall be used to describe the mutual exclusion.

8.2.5.2 ASF objects affected

This guideline pertains to the **Advanced Mutual Exclusion Object** and the **Extended Stream Properties Object**.

8.2.5.3 Readers affected

Bit rate mutual exclusion among streams other than video is supported only by the components of Windows Media 9 Series. Following this guideline will hide the mutual exclusion relationship as well as all but one of the streams from previous versions of Windows Media technologies.

8.2.6 Multiple independent audio or video streams

8.2.6.1 Guideline

When creating content that has multiple audio or video streams that are not in a mutual exclusion relationship, all but one stream for each media type should be hidden from earlier implementations according to the manner described in section 8.2.2.

8.2.6.2 ASF objects affected

This guideline pertains to the **Extended Stream Properties Object**.

8.2.6.3 Readers affected

The presence of multiple independent audio or video streams is supported only by the components of Windows Media 9 Series. Following this guideline will hide all but one stream for each media type from previous version of Windows Media technologies.

8.2.7 Unknown stream IDs in the payloads

8.2.7.1 Guideline

Stream IDs in the packet payloads of the ASF **Data Object** do not necessarily need to correspond to the streams declared in the ASF header. All products covered in this section support "unknown" stream IDs; however, be aware that Windows Media Services 4.1 sends these payloads over the network, so they do take up bandwidth.

8.2.7.2 ASF objects affected

This guideline pertains to the ASF packets in the **Data Object**.

8.2.7.3 Readers affected

As already mentioned, this guideline points out how "unknown" stream IDs in the payloads affect the performance of Windows Media Services 4.1.

8.2.8 Multi-language presentations

8.2.8.1 Guideline

When creating content with language mutual exclusion relationships, the guidelines in section 8.2.6 shall be followed to ensure that for each media type, all but one stream is hidden from earlier implementations according to the manner described in section 8.2.2. The stream that is not hidden from earlier implementations shall be in the default language for the presentation.

8.2.8.2 ASF objects affected

This guideline pertains to the **Advanced Mutual Exclusion Object** or the **Group Mutual Exclusion Object** as well as the **Extended Stream Properties Object**.

8.2.8.3 Readers affected

Language-based mutual exclusion is supported only by the components of Windows Media 9 Series. Following this guideline will hide all but one stream for each media type from previous versions of Windows Media technologies.

8.2.9 Group mutual exclusion

8.2.9.1 Guideline

When creating content with the **Group Mutual Exclusion Object**, the streams in all but one record of the object shall be hidden from earlier implementations according to the manner described in section 8.2.2.

8.2.9.2 ASF objects affected

This guideline pertains to the **Group Mutual Exclusion Object** and the **Extended Stream Properties Object**.

8.2.9.3 Readers affected

The ability to create and recognize mutually exclusive relationships between groups of streams is supported only by the components of Windows Media 9 Series. Following this guideline will hide the streams in all but one group ("record") from previous versions of Windows Media technologies.

8.2.10 Presence of Stream Bitrate Properties Object

8.2.10.1 Guideline

When creating content, the optional **Stream Bitrate Properties Object** should be included as specified in section 3.12.

8.2.10.2 ASF objects affected

This guideline pertains to the **Stream Bitrate Properties Object.**

8.2.10.3 Readers affected

For implementations that do not recognize the **Header Extension Object**, this object aids in stream selection.

8.2.11 Custom top-level objects

8.2.11.1 Guideline

Custom ASF top-level objects may be added directly after the **Data Object** and before any of the index objects. They must not be added anywhere else. If the content will be streamed by Windows Media® Services 4.1, there must be no custom ASF top-level objects anywhere.

In addition, some earlier ASF implementations on devices cannot handle top-level objects besides the **Header Object**, the **Data Object**, and the **Simple Index Object** at all, so content targeting these devices should use these three top-level objects only.

8.2.11.2 ASF objects affected

This guideline pertains to custom top-level ASF objects.

8.2.11.3 Readers affected

Of the implementations covered in this section, Windows Media Services 4.1 cannot open content with any custom top-level objects. All other implementations covered in this section will ignore custom top-level objects added directly after the **Data Object**.

As already mentioned, content containing top-level objects besides the **Header Object**, the **Data Object**, and the **Simple Index Object** may be rejected altogether by earlier ASF implementations on devices.

8.2.12 Index objects

8.2.12.1 Guideline

For indexing content, the **Simple Index Object** shall be used, alone, whenever possible. **Simple Index Object**s can index video streams that are declared with separate **Stream Properties Objects** in the ASF Header. For instance, content with one audio stream and multiple bit rate mutually-exclusive video streams (all with the same frame size) can be indexed by using one **Simple Index Object** for each video stream.

When any of the other types of **Index Object**s (including the **Media Object Index Object** and the **Timecode Index Object**) are needed for indexing the content, they shall be after the **Data Object** and before the **Simple Index Object**. The **Index Object** (or the group of **Index Objects**, when there are more than one) shall be followed by at least one **Simple Index Object**. These **Simple Index Objects** shall index all video streams that are declared by separate **Stream Properties Objects** in the ASF header. If there are no such streams, then there shall be a single **Simple Index Object** with zero entries.

When a **Frame Index Object** is included, writer implementations should write the **NumberOfFrames** attribute along with the index. **NumberOfFrames** is a QWORD metadata attribute in the **Extended Content Description** object.

8.2.12.2 ASF objects affected

This guideline affects the various ASF index objects.

8.2.12.3 Readers affected

- All implementations covered in this section prior to the components of Windows Media 9 Series recognize only the **Simple Index Object**.
- Any content with any index objects besides the **Simple Index Object**(s) cannot be opened by Windows Media Services 4.1. It is also known that certain earlier ASF implementations on devices cannot handle index objects besides the **Simple Index Object**(s) (for more information, see section 8.2.11).
- The Windows Media Format 7 SDK requires that the **Index Objects** be followed by at least one **Simple Index Object**.
- The Windows Media Format 9 SDK and later require the presence of the **NumberOfFrames** attribute in addtion to a **Frame Index Object** to perform frame-based seeking in a specified piece of content.

8.2.13 Do not create content with variable-size packets

8.2.13.1 Guideline

Packet size shall be fixed in all content that is created.

8.2.13.2 ASF objects affected

This guideline pertains to the packets in the **Data Object**.

8.2.13.3 Readers affected

None of the implementations covered in this section support variable-size packets.

8.2.14 Packet size must be under 64 KB

8.2.14.1 Guideline

Packet sizes shall not exceed 64 KB.

8.2.14.2 ASF objects affected

This guideline pertains to the packets in the **Data Object**.

8.2.14.3 Readers affected

None of the implementations covered in this section are guaranteed to support packet sizes larger than 64 KB.

8.2.15 Padding length must be accurate

8.2.15.1 Guideline

The **Padding length** field in the packet parsing section of an ASF packet shall accurately reflect the amount of padding at the end of the packet. However, as noted in section 5.2.2, ASF reader implementations should be prepared to handle cases when this value is zero and to infer the correct value from the size of the payloads.

8.2.15.2 ASF objects affected

This guideline pertains to the packets in the **Data Object**.

8.2.15.3 Readers affected

None of the implementations covered in this section are guaranteed to be able to handle inaccurate **Padding length** values for on-disk ASF content, and therefore ASF writer implementations should always set the **Padding length** field correctly.

Windows Media Services is known to create and send packets with the **Padding length** equal to zero, and therefore ASF reader implementations should be able to parse such packets.

8.2.16 Ordering of payloads and media objects in packets

8.2.16.1 Guideline

Presentation times in payloads of a single media object should be the same and presentation times of media objects should only increase. The associated object ids in payloads of a single media

object should be the same and object ids of media objects should be incremental without gaps unless there is a discontinuity and/or object id wrap around.

8.2.16.2 ASF objects affected

This guideline pertains to the packets in the **Data Object**.

8.2.16.3 Readers affected

None of the implementations covered in this section are guaranteed to support non-consecutive ordering of payloads and media objects.

9. Standard ASF media types

ASF files store a wide variety of digital media content. It is anticipated that implementations will produce unique media types of their own creation. However, a rich set of standard media types that are commonly supported are defined to enable compatibility between diverse implementations.

The purpose of this section is to define a set of standard ASF media types. The explicit intention of this section is that if an implementation supports a media type defined within this section, that type must be supported in the manner described within this section if the implementation is to be considered content-compliant with the ASF specification. This commonality will define a minimum subset of digital media within which multi-vendor interoperability is possible. No restrictions are placed on how implementations support nonstandard media types (in other words, types other than those covered in this section).

The following subsections will define the core media types for audio, video, and commands.

9.1 Audio media type

When the **Stream Type** of the **Stream Properties Object** has the value **ASF_Audio_Media**, the ASF audio media type that populates the **Type-Specific Data** field of the **Stream Properties Object** is represented using the following structure (the **WAVEFORMATEX** structure).

Field name	Field type	Size (bits)
Codec ID / Format Tag	WORD	16
Number of Channels	WORD	16
Samples Per Second	DWORD	32
Average Number of Bytes Per Second	DWORD	32
Block Alignment	WORD	16
Bits Per Sample	WORD	16
Codec Specific Data Size	WORD	16
Codec Specific Data	BYTE	varies

The fields are defined as follows:

Codec ID / Format Tag

Specifies the unique ID of the codec used to encode the audio data. There is a registration procedure for new codecs. Defined as the **wFormatTag** field of a **WAVEFORMATEX** structure.

Number of Channels

Specifies the number of audio channels. Monaural data uses one channel and stereo data uses two channels. 5.1 audio uses six channels. Defined as the **nChannels** field of a **WAVEFORMATEX** structure.

Samples Per Second

Specifies a value in Hertz (cycles per second) that represents the sampling rate of the audio stream. Defined as the **nSamplesPerSec** field of a **WAVEFORMATEX** structure.

Average Number of Bytes Per Second

Specifies the average number of bytes per second of the audio stream. Defined as the **nAvgBytesPerSec** field of a **WAVEFORMATEX** structure.

Block Alignment

Specifies the block alignment, or block size, in bytes of the audio codec. Defined as the **nBlockAlign** field of a **WAVEFORMATEX** structure.

Bits per Sample

Specifies the number of bits per sample of monaural data. Defined as the **wBitsPerSample** field of a **WAVEFORMATEX** structure.

Codec Specific Data Size

Specifies the size, in bytes, of the **Codec Specific Data** buffer. Defined as the **cbSize** field of a **WAVEFORMATEX** structure. This value should be 0 when Codec ID is 1 (WAVE_FORMAT_PCM).

Codec Specific Data

Specifies an array of codec-specific data bytes.

For more information about the **WAVEFORMATEX** structure, see the *MSDN Library* documentation.

9.1.1 Spread audio

One **Error Correction Type** is spread audio. This refers to an error correction approach that minimizes the impact of lost audio data by spreading audio over a span of packets. The compressed silence is used for silence injection if lost payload data cannot be recreated. This approach works well for fixed bit rate audio codecs that have no interframe dependencies.

The **Error Correction Data** field is represented using the following structure.

Field name	Field type	Size (bits)
Span	BYTE	8
Virtual Packet Length	WORD	16
Virtual Chunk Length	WORD	16
Silence Data Length	WORD	16
Silence Data	BYTE	varies

The fields are defined as follows:

Span

Specifies the number of packets over which audio will be spread. Typically, this value should be set to 1.

Virtual Packet Length

Specifies the virtual packet length. The value of this field should be set to the size of the largest audio payload found in the audio stream.

Virtual Chunk Length

Specifies the virtual chunk length. The value of this field should be set to the size of the largest audio payload found in the audio stream.

Silence Data Length

Specifies the number of bytes stored in the **Silence Data** field. This value should be set to 1. It is also valid for this value to equal the **Block Alignment** value (from the **Audio Media Type**).

Silence Data

Specifies an array of silence data bytes. This value should be set to 0 for the length of **Silence Data Length**.

9.1.2 Audio payload sizes

Audio payloads do not need to be of equal size. However, they need to be a multiple of the **Block Alignment** field of the **WAVEFORMATEX** structure defined at the beginning of this section.

9.2 Video media type

When the **Stream Type** of the **Stream Properties Object** has the value **ASF_Video_Media**, the ASF video media type that populates the **Type-Specific Data** field of the **Stream Properties Object** is represented using the following structure.

Field name	Field type	Size (bits)
Encoded Image Width	DWORD	32
Encoded Image Height	DWORD	32
Reserved Flags	BYTE	8
Format Data Size	WORD	16
Format Data	See below	varies

The fields are defined as follows:

Encoded Image Width

Specifies the width of the encoded image in pixels.

Encoded Image Height

Specifies the height of the encoded image in pixels.

Reserved Flags

Specifies reserved flags, and shall be set to 2.

Format Data Size

Specifies the size of the **Format Data** field in bytes.

Format Data

Specifies the details of the format of the image data. This format is structured as follows (the **BITMAPINFOHEADER** structure):

Field name	Field type	Size (bits)
Format Data Size	DWORD	32
Image Width	LONG	32

Image Height	LONG	32
Reserved	WORD	16
Bits Per Pixel Count	WORD	16
Compression ID	DWORD	32
Image Size	DWORD	32
Horizontal Pixels Per Meter	LONG	32
Vertical Pixels Per Meter	LONG	32
Colors Used Count	DWORD	32
Important Colors Count	DWORD	32
Codec Specific Data	BYTE	varies

The fields are defined as follows:

Format Data Size

Specifies the number of bytes stored in the **Format Data** field. Defined as the **biSize** field of a **BITMAPINFOHEADER** structure.

Image Width

Specifies the width of the encoded image in pixels. Defined as the **biWidth** field of a **BITMAPINFOHEADER** structure. This should be equal to the **Encoded Image Width** field defined previously.

Image Height

Specifies the height of the encoded image in pixels. Defined as the **biHeight** field of a **BITMAPINFOHEADER** structure. This should be equal to the **Encoded Image Height** field defined previously.

Reserved

Reserved. Shall be set to 1. Defined as the **biPlanes** field of a **BITMAPINFOHEADER** structure.

Bits Per Pixel Count

Specifies the number of bits per pixel. Defined as the **biBitCount** field of a **BITMAPINFOHEADER** structure.

Compression ID

Specifies the type of the compression, using a four-character code. For ISO MPEG-4 video, this contains MP4S, mp4s, M4S2, or m4s2. In the Compression ID, the first character of the four-character code appears as the least-significant byte; for instance MP4S uses the Compression ID 0x5334504D. Defined as the **biCompression** field of a **BITMAPINFOHEADER** structure.

Image Size

Specifies the size of the image in bytes. Defined as the **biSizeImage** field of a **BITMAPINFOHEADER** structure.

Horizontal Pixels Per Meter

Specifies the horizontal resolution of the target device for the bitmap in pixels per meter. Defined as the **biXPelsPerMeter** field of a **BITMAPINFOHEADER** structure.

Vertical Pixels Per Meter

Specifies the vertical resolution of the target device for the bitmap in pixels per meter. Defined as the **biYPelsPerMeter** field of a **BITMAPINFOHEADER** structure.

Colors Used Count

Specifies the number of color indexes in the color table that are actually used by the bitmap. Defined as the **biClrUsed** field of a **BITMAPINFOHEADER** structure.

Important Colors Count

Specifies the number of color indexes that are required for displaying the bitmap. If this value is zero, all colors are required. Defined as the **biClrImportant** field of a **BITMAPINFOHEADER** structure.

Codec Specific Data

Specifies an array of codec specific data bytes. The size of this array is equal to the **Format Data Size** field minus the size of the **Format Data** fields listed previously.

For more information about the **BITMAPINFOHEADER** structure, see the *MSDN Library* documentation.

9.3 Command media type

When the **Stream Type** of the **Stream Properties Object** has the value **ASF_Command_Media**, the ASF command media type that populates the **Type-Specific Data** field of the **Stream Properties Object** shall be null and the value of **Type-Specific Data Length** shall be 0.

Whereas the name-value pairs associated with the commands can be any value, system-defined command types include URL, Filename, and Text. The URL command type indicates that the URL is to be opened by a client in an HTML window or frame. The Filename command type indicates that the digital media file indicated is to be played immediately. The Text command type indicates that the data strings should be interpreted as captioned text to go along with the presentation.

For commands that are not stored in the **Script Command Object** (see section 3.6), each digital media sample is composed of a nul-terminated command type string followed by a nul-terminated command string. (Note that previous versions of this specification indicated that there should be an extra nul WCHAR between the two strings. This was not correct; there should be exactly one nul WCHAR between the strings, and that should be the nul-terminating character for the first string.)

9.4 Image media type

ASF supports both a JFIF image type and a degradable JPEG image type. The former is a media type indicating that the stream is in the JFIF format; the latter is a media type indicating that it is a loss-tolerant stream of JPEG images. To encode to or decode from the degradable JPEG type, you must use the Windows Media Format SDK.

9.4.1 JFIF/JPEG media type

When the **Stream Type** of the **Stream Properties Object** has a value equal to **ASF_JFIF_Media**, the ASF media type that populates the **Type-Specific Data** field of the **Stream Properties Object** is represented using the following structure.

Field name	Field type	Size (bits)
Image width	DWORD	32
Image height	DWORD	32
Reserved	DWORD	32

The fields are defined as follows:

Image width

Specifies the width of the encoded image in pixels.

Image height

Specifies the height of the encoded image in pixels.

Reserved

Reserved, must be 0.

9.4.2 Degradable JPEG media type

When the **Stream Type** of the **Stream Properties Object** has a value equal to **ASF_Degradable_JPEG_Media**, the ASF media type that populates the **Type-Specific Data** field of the **Stream Properties Object** is represnted using the following structure.

Field name	Field type	Size (bits)
Image width	DWORD	32
Image height	DWORD	32
Reserved	WORD	16
Reserved	WORD	16
Reserved	WORD	16
Interchange data length	WORD	16
Interchange data	BYTE	varies

The fields are defined as follows:

Image width

Specifies the width of the encoded image in pixels.

Image height

Specifies the height of the encoded image in pixels.

Reserved

These three fields must be, respectively, 0, 2, and 4.

Interchange data length

Specifies the number of bytes in the **Interchange data** field. If this value is 0, then **Interchange data** field shall consist of the single byte value 0x00.

Interchange data

Specifies the interchange data for this stream. If **Interchange data length** is set to 0, then this field shall still be present and shall consist of the single byte 0x00.

9.5 File transfer and binary media types

When the **Stream Type** of the **Stream Properties Object** has a value equal to either **ASF_File_Transfer_Media** or **ASF_Binary_Media**, the ASF media type that populates the **Type-Specific Data** field of the **Stream Properties Object** is represented using the following structure.

Field name	Field type	Size (bits)
Major media type	GUID	128
Media subtype	GUID	128
Fixed-size samples	DWORD	32
Temporal compression	DWORD	32
Sample size	DWORD	32

Format type	GUID	128
Format data size	DWORD	32
Format data	See below	varies

The fields are defined as follows:

Major media type

This value must be equal to the **Stream Type** value in the **Stream Properties Object**.

Media subtype

Indicates the media subtype. Can be set to 0 if not relevant. **ASF_Web_Stream_Media_Subtype** is a possible value for file transfer streams that are Web streams.

Fixed-size samples

Valid values are 0 and 1. This value shall be set to 1 if this stream has fixed-size samples.

Temporal compression

Valid values are 0 and 1. This value shall be set to 1 if compression of media object N might depend on media object N-1. In general, this value should be set to 0.

Sample size

If the **Fixed-size samples** field has a value of 1, then this value is the fixed sample size. Otherwise, the value is ignored and should be 0.

Format type

If there is no additional media type information, this field, along with the value in the **Format data size** field, should be set to 0. For a Web stream, this can be set to **ASF_Web_Stream_Format**. Custom non-standard format types can also be defined, but they will not necessarily be understood across implementations.

Format data size

This is the number of bytes in **Format data**. If there is no format data, this field, along with the value in the **Format type** field, should be set to 0.

Format data

This is the additional format data for this media type. This shall be present only if **Format data size** is greater than 0. Custom format types can define how these bytes are formatted. If the **Format type** is equal to **ASF_Web_Stream_Format**, then there is a standard format for these bytes, detailed in section 9.5.1.

9.5.1 Web streams

Web streams are a subtype of file transfer streams. The media type should be expressed as previously mentioned. The **Media subtype** shall be set to **ASF_Web_Stream_Media_Subtype**. The **Format type** shall be set to **ASF_Web_Stream_Format**. The **Format data size** shall be set to 8. The **Format data** should use the values in the following table.

Field name	Field type	Size (bits)
Web stream format data size	WORD	16
Fixed sample header size	WORD	16
Version number	WORD	16
Reserved	WORD	16

The fields are defined as follows:

Web stream format data size

This shall be set to 8.

Fixed sample header size

This shall be set to 10. See below for a description of the Web stream header.

Version number

This shall be set to 1.

Reserved

Reserved, must be 0.

In addition, all media objects for a Web stream must begin with 10 bytes formatted as follows:

Field name	Field type	Size (bits)
Total header length	WORD	16
Part number	WORD	16
Total part count	WORD	16
Sample type	WORD	16
URL string	WCHAR	Varies

The fields are defined as follows:

Total header length

This is the total size of the media object header. This value should be set to 10 plus the length (not including the nul-terminating character) of the **URL string** field.

Part number

Current part of the file (0-based). Valid values are from 0 to Total number of parts – 1.

Total part count

Number of parts in the file.

Sample type

Valid values for this field are 1, which indicates the sample type is “file”, and 2, which indicates the sample type is “render” (which is essentially a command to render the data).

URL string

This is a nul-terminated string containing the URL for the file being transferred.

10. ASF GUIDs

GUIDs are used to uniquely identify all objects and entities within ASF files. This provides a foundation for extensibility and flexibility.

A GUID uniquely identifies each ASF media object type. New media types, codec types, error correction approaches, and other innovations can be created, identified by their own GUIDs, and inserted into ASF data streams.

New ASF object types may be defined. This extensibility enables ASF to support new innovations as they arise. Note, however, that each new ASF object type needs its own unique GUID.

The following sections describe standard GUIDs that have been defined for all ASF objects and related fields within this specification. Implementations may supplement this list with additional GUIDs when necessary to identify entities, elements, or ideas that have not yet been enumerated by this section.

10.1 Top-level ASF object GUIDS

The following table contains the names and values of top-level ASF object GUIDs.

Name	GUID
ASF_Header_Object	75B22630-668E-11CF-A6D9-00AA0062CE6C
ASF_Data_Object	75B22636-668E-11CF-A6D9-00AA0062CE6C
ASF_Simple_Index_Object	33000890-E5B1-11CF-89F4-00A0C90349CB
ASF_Index_Object	D6E229D3-35DA-11D1-9034-00A0C90349BE
ASF_Media_Object_Index_Object	FEB103F8-12AD-4C64-840F-2A1D2F7AD48C
ASF_Timecode_Index_Object	3CB73FD0-0C4A-4803-953D-EDF7B6228F0C

10.2 Header Object GUIDs

The following table contains the names and values of standard ASF **Header Object** GUIDs.

Name	GUID
ASF_File_Properties_Object	8CABDCA1-A947-11CF-8EE4-00C00C205365
ASF_Stream_Properties_Object	B7DC0791-A9B7-11CF-8EE6-00C00C205365
ASF_Header_Extension_Object	5FBF03B5-A92E-11CF-8EE3-00C00C205365
ASF_Codec_List_Object	86D15240-311D-11D0-A3A4-00A0C90348F6
ASF_Script_Command_Object	1EFB1A30-0B62-11D0-A39B-00A0C90348F6
ASF_Marker_Object	F487CD01-A951-11CF-8EE6-00C00C205365
ASF_Bitrate_Mutual_Exclusion_Object	D6E229DC-35DA-11D1-9034-00A0C90349BE
ASF_Error_Correction_Object	75B22635-668E-11CF-A6D9-00AA0062CE6C
ASF_Content_Description_Object	75B22633-668E-11CF-A6D9-00AA0062CE6C
ASF_Extended_Content_Description_Object	D2D0A440-E307-11D2-97F0-00A0C95EA850
ASF_Content_Branding_Object	2211B3FA-BD23-11D2-B4B7-00A0C955FC6E
ASF_Stream_Bitrate_Properties_Object	7BF875CE-468D-11D1-8D82-006097C9A2B2
ASF_Content_Encryption_Object	2211B3FB-BD23-11D2-B4B7-00A0C955FC6E
ASF_Extended_Content_Encryption_Object	298AE614-2622-4C17-B935-DAE07EE9289C
ASF_Digital_Signature_Object	2211B3FC-BD23-11D2-B4B7-00A0C955FC6E
ASF_Padding_Object	1806D474-CADF-4509-A4BA-9AABCB96AAE8

10.3 Header Extension Object GUIDs

The following table contains the names and values of the GUIDs for the standard objects found inside the ASF **Header Extension Object**.

Name	GUID
ASF_Extended_Stream_Properties_Object	14E6A5CB-C672-4332-8399-A96952065B5A
ASF_Advanced_Mutual_Exclusion_Object	A08649CF-4775-4670-8A16-6E35357566CD
ASF_Group_Mutual_Exclusion_Object	D1465A40-5A79-4338-B71B-E36B8FD6C249
ASF_Stream_Prioritization_Object	D4FED15B-88D3-454F-81F0-ED5C45999E24
ASF_Bandwidth_Sharing_Object	A69609E6-517B-11D2-B6AF-00C04FD908E9
ASF_Language_List_Object	7C4346A9-EFE0-4BFC-B229-393EDE415C85
ASF_Metadata_Object	C5F8CBEA-5BAF-4877-8467-AA8C44FA4CCA

ASF_Metadata_Library_Object	44231C94-9498-49D1-A141-1D134E457054
ASF_Index_Parameters_Object	D6E229DF-35DA-11D1-9034-00A0C90349BE
ASF_Media_Object_Index_Parameters_Object	6B203BAD-3F11-48E4-ACA8-D7613DE2CFA7
ASF_Timecode_Index_Parameters_Object	F55E496D-9797-4B5D-8C8B-604DFE9BFB24
ASF_Compatibility_Object	26F18B5D-4584-47EC-9F5F-0E651F0452C9
ASF_Advanced_Content_Encryption_Object	43058533-6981-49E6-9B74-AD12CB86D58C

10.4 Stream Properties Object Stream Type GUIDs

The following table contains the names and values of standard GUIDs for the **Stream Type** field of the **Stream Properties Object**.

Name	GUID
ASF_Audio_Media	F8699E40-5B4D-11CF-A8FD-00805F5C442B
ASF_Video_Media	BC19EFC0-5B4D-11CF-A8FD-00805F5C442B
ASF_Command_Media	59DACFC0-59E6-11D0-A3AC-00A0C90348F6
ASF_JFIF_Media	B61BE100-5B4E-11CF-A8FD-00805F5C442B
ASF_Degradable_JPEG_Media	35907DE0-E415-11CF-A917-00805F5C442B
ASF_File_Transfer_Media	91BD222C-F21C-497A-8B6D-5AA86BFC0185
ASF_Binary_Media	3AFB65E2-47EF-40F2-AC2C-70A90D71D343

10.4.1 Web stream Type-Specific Data GUIDs

The following table contains the names and values of the GUIDs used in the **Type-Specific Data** field of the **Stream Properties Object**.

Name	GUID
ASF_Web_Stream_Media_Subtype	776257D4-C627-41CB-8F81-7AC7FF1C40CC
ASF_Web_Stream_Format	DA1E6B13-8359-4050-B398-388E965BF00C

10.5 Stream Properties Object Error Correction Type GUIDs

The following table contains the names and values of the GUIDs for the **Error Correction Type** field of the **Stream Properties Object** as well as in the **Error Correction Object**.

Name	GUID
ASF_No_Error_Correction	20FB5700-5B55-11CF-A8FD-00805F5C442B
ASF_Audio_Spread	BFC3CD50-618F-11CF-8BB2-00AA00B4E220

10.6 Header Extension Object GUIDs

The following table contains the name and value of the GUID for the **Reserved 1** field of the **Header Extension Object**.

Name	GUID
ASF_Reserved_1	ABD3D211-A9BA-11cf-8EE6-00C00C205365

10.7 Advanced Content Encryption Object System ID GUIDs

The following table contains the names and values of standard GUIDs for the **System ID** field of the **Advanced Content Encryption Object**.

Name	GUID
ASF_Content_Encryption_System_Windows_Media_DRM_Network_Devices	7A079BB6-DAA4-4e12-A5CA-91D38DC11A8D

10.8 Codec List Object GUIDs

The following table contains the name and value of the GUIDs for the **Reserved 2** field of the **Codec List Object**.

Name	GUID
ASF_Reserved_2	86D15241-311D-11D0-A3A4-00A0C90348F6

10.9 Script Command Object GUIDs

The following table contains the name and value of the GUIDs for the **Reserved 3** field of the **Script Command Object**.

Name	GUID
ASF_Reserved_3	4B1ACBE3-100B-11D0-A39B-00A0C90348F6

10.10 Marker Object GUIDs

The following table contains the name and value of the GUIDs for the **Reserved 4** field of the **Marker Object**.

Name	GUID
ASF_Reserved_4	4CFEDB20-75F6-11CF-9C0F-00A0C90349CB

10.11 Mutual Exclusion Object Exclusion Type GUIDs

The following table contains the names and values of the GUIDs for the **Exclusion Type** field of the **Mutual Exclusion Object**:

Name	GUID
ASF_Mutex_Language	D6E22A00-35DA-11D1-9034-00A0C90349BE
ASF_Mutex_Bitrate	D6E22A01-35DA-11D1-9034-00A0C90349BE
ASF_Mutex_Unknown	D6E22A02-35DA-11D1-9034-00A0C90349BE

10.12 Bandwidth Sharing Object GUIDs

The following table contains the names and values of the GUIDs for the **Sharing Type** field of the **Bandwidth Sharing Object**.

Name	GUID
ASF_Bandwidth_Sharing_Exclusive	AF6060AA-5197-11D2-B6AF-00C04FD908E9
ASF_Bandwidth_Sharing_Partial	AF6060AB-5197-11D2-B6AF-00C04FD908E9

10.13 Standard Payload Extension System GUIDs

The following table contains the names and values of the GUIDs for the standard **Payload Extension Systems**.

Name	GUID
ASF_Payload_Extension_System_Timecode	399595EC-8667-4E2D-8FDB-98814CE76C1E
ASF_Payload_Extension_System_File_Name	E165EC0E-19ED-45D7-B4A7-25CBD1E28E9B
ASF_Payload_Extension_System_Content_Type	D590DC20-07BC-436C-9CF7-F3BBFBF1A4DC
ASF_Payload_Extension_System_Pixel_Aspect_Ratio	1B1EE554-F9EA-4BC8-821A-376B74E4C4B8
ASF_Payload_Extension_System_Sample_Duration	C6BD9450-867F-4907-83A3-C77921B733AD
ASF_Payload_Extension_System_Encryption_Sample_ID	6698B84E-0AFA-4330-AEB2-1C0A98D7A44D
ASF_Payload_Extension_System_Degradable_JPEG	00E1AF06-7BEC-11D1-A582-00C04FC29CFB

11. Codec information

This section describes how specific audio and video information is stored in the **Stream Properties Object**. For more information, see sections 9.1 and 9.2.

11.1 Audio codec type-specific data in ASF

This section outlines how audio information is stored in the **Stream Properties Object** for the following codecs.

Codec name	Format tag	Notes
Windows Media Audio	0x161	Versions 7, 8, and 9 Series
Windows Media Audio 9 Professional	0x162	9 Series
Windows Media Audio 9 Lossless	0x163	9 Series
GSM-AMR	0x7A21	Fixed bit rate, no SID
GSM-AMR	0x7A22	Variable bit rate, including SID

11.1.1 Windows Media Audio

The type-specific information for the Windows Media Audio codec is stored in the following format as part of the **Type-Specific Data** of the **Stream Properties Object** as outlined in section 9.1.

```
struct WMA_TYPE_SPECIFIC_DATA
{
    WAVEFORMATEX wfx;
    DWORD        dwSamplesPerBlock;
    WORD         wEncodeOptions;
    DWORD        dwSuperBlockAlign;
};
```

11.1.2 GSM-AMR

The type-specific information for GSM-AMR is stored in the following format as part of the **Type-Specific Data** of the **Stream Properties Object** as outlined in section 9.1.

```
struct GSMAMR_TYPE_SPECIFIC_DATA
{
    WAVEFORMATEX wfx;
    // bit[0] = SID is used (must be zero in case wFormat == 0x7A21)
    // bit[1] = varying bitrate is used (must be zero in case wFormat == 0x7A21)
    DWORD dwFlags;
};
```

11.2 MPEG-4 Video type-specific data in ASF

This section explains how a video decoder should decode the stream header information (decoder configuration) contained in ASF. This recommendation ensures that the decoder will be compatible with current and future encoded content that makes use of the MPEG video FourCC codes MP4S and M4S2. To achieve this full compatibility, only a very simple test is needed in the decoder software to decode the video headers correctly.

11.2.1 Background

There are two raw MPEG-4 video stream types, with FourCCs MP4S and M4S2. At the frame level, all MPEG-4 compatible decoders can decode both stream types, but there is a difference at the level of the initial header information. MP4S is an earlier format that was based on MPEG-4 version 1 of August 1999 and was compatible with the MPEG ISO reference software of that time. Consequently, some of the new version 2 header information is not included in this stream type. M4S2 is the new format and is fully compatible with MPEG-4 version 2 bit stream as it is now defined in ISO MPEG-4 document ISO/IEC-14496.

To achieve maximum compatibility, video decoders should be able to decode both stream types. This is very simple to achieve with only a tiny modification to the decoding process.

11.2.2 Decoding process

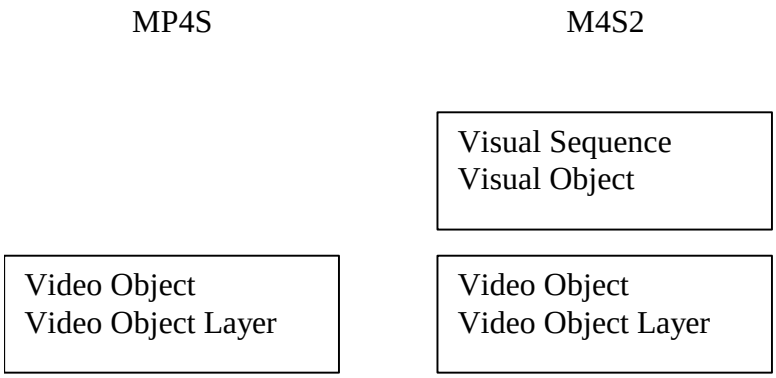
The video stream is made up of two parts: the video bit stream header, contained in the extended **BITMAPINFOHEADER** (or "EBIH"), and the video bit stream itself, consisting of compressed frames. To handle both FourCC types, it is necessary to make a software test to correctly decode the EBIH. There are no changes to the method of decoding the compressed frames.

The EBIH contains decoder configuration information:

- In the case of the MP4S stream type, it has the format described in the section "Decoding MP4S header information". The first start code is **video_object_start_code**. All remaining syntax elements are fully compatible with MPEG-4 with visual_object_verid = 1.

- In the case of M4S2, the header has the format described in ISO/IEC-14496, and the first start code is **visual_object_sequence_start_code**, and all syntax elements are fully compatible with MPEG-4. However, to allow flexibility for those people generating short header streams using M4S2, we currently allow the whole MPEG-4 header (from Visual Sequence to Video Object Layer, inclusive) to be replaced by a single 22-bit **short_video_start_marker**. This is not compatible with MPEG-4, but is allowed for convenience.

The following diagram shows the difference between MP4S and M4S2 EBMI headers:



The following code should be used to handle both header types. It is based on examining the first few bits of the EBIH header.

```
if( peekbits(22) == 0x20 )  // short_header_start_marker
{
      // Ignore the rest of EBIH; decode as short header stream.
}
else if ( peekbits(32) == 0x1B0 )  // visual_object_sequence_start_code
{
      // M4S2
      DecodeVisualSequenceHeader(…);
      DecodeVisualObjectHeader(…);
}
else
```

```
{
      // MP4S
      // Assume default values for visual headers above.
      // (See section 11.2.3.2 of this document.)
      InitDefaultVisualSequenceAndObjectElements();

      // Set default value of visual object syntax element
      // to indicate version number of bit stream.
      visual_object_verid = 1;
}

DecodeVideoObjectHeader(…);
DecodeVideoObjectLayerHeader(…);
```

This pseudo code is very simple to implement and provides full compatibility with existing and future encoders. An alternative to testing the bit stream is to test the value of the FourCC to decide which branch to take.

11.2.3 Decoding MP4S header information

- Any stream containing the M4S2 FourCC includes the appropriate Visual Sequence and Visual Object headers.
- To decode MP4S content, decoders should be able to handle a bit stream without Visual Sequence and Visual Object headers, and they should assume default values for these headers. (The default values are shown in the section 11.2.3.2.)
- Decoding of the header should then start from the Video Object header.

11.2.3.1 Format of MP4S EBIH (Video Object and Video Object Layer headers)

Syntax element	Number of bits	Value from Windows Media Encoder
video_object_start_code	32	0x00000100
video_object_layer_start_code	32	0x00000120
random_accessible_vol	1	0
video_object_type_indication	8	0x01 (= simple object)
is_object_layer_identifier	1	0
aspect_ratio_info	4	1 (= square)
vol_control_parameters	1	0
video_object_layer_shape	2	0
marker_bit	1	1
vop_time_increment_resolution	16	resolution of the time line (see note 1)
marker_bit	1	1
fixed_vop_rate	1	0
marker_bit	1	1
video_object_layer_width	13	Frame Width
marker_bit	1	1

video_object_layer_height	13	Frame Height
marker_bit	1	1
Interlaced	1	0
obmc_disable	1	1
sprite_enable	1 (see note 2)	0
not_8_bit	1	0
quant_type	1	0
complexity_estimation_disable	1	1
resync_marker_disable	1	Can be 1 or 0.
data_partitioned	1	0
Scalability	1	0
Padding to next byte boundary	6	0b011111 (see note 3)

Note 1: This field indicates the number of evenly spaced subintervals, called ticks, within one modulo time. One modulo time represents the fixed interval of one second. For details, see the MPEG-4 Visual standard (ISO/IEC 14496-2).

Note 2: visual_object_verid is assumed to be equal to 1; therefore, sprite enable has 1 bit.

Note 3: A zero stuffing bit followed by a number of one stuffing bits shall be present until the current position is on a byte boundary. As described above, the numbers of bits in MP4S EBIH before this field are 138, so 0b011111 is stuffed.

11.2.3.2 Default Values for Visual Sequence and Visual Object Headers when decoding MP4S

Syntax element	Number of bits	Default value
visual_object_sequence_start_code	32	0x000001B0
profile_and_level_indication	8	0x01
visual_object_start_code	32	0x000001B5
is_visual_object_identifier	1	0
visual_object_type	4	0x01
video_signal_type	1	0
Padding to next byte boundary	2	0x01

Note that because visual_object_verid is not present here, its value is assumed to be = 1.

Appendix A: VC-1 Video streams in ASF

A.1 Background

VC-1 is a published SMPTE standard for video compression technology. There are three profiles of VC-1: Simple, Main, and Advanced. This document provides information on how VC-1 is carried in ASF files.

Note: Some of this information is not required for decoding VC-1 bitstreams, but can be useful for implementing a more efficient decoder. Please refer to [1] for VC-1 related definitions.

Video streams in ASF are made up of two parts: the video bitstream header, contained in CodecPrivateData (also referred to as extended BITMAPINFOHEADER), and the video bitstream itself, consisting of compressed frames, which may be progressive frames, interlaced frames, or two interlaced fields.

VC-1 that is carried over ASF has two different Four Character Code (FourCC) values. VC-1 Simple and Main profile bitstreams have the FourCC value corresponding to 'WMV3'. VC-1 Advanced profile bitstreams have the FourCC value corresponding to 'WVC1'.

The following sections explain the differences between how headers and compressed data are stored in ASF for WMV3 and for WVC1.

A.2 Overview of VC-1 streams in ASF

Simple and Main profile bitstreams carried over ASF use the same mechanism to store sequence headers. The sequence header of Simple or Main profile is specified in Annex J of the VC-1 specification[1].

Advanced profile, which supports a richer set of features, uses a different mechanism to store headers. The sequence header and entry point header of the Advanced profile are specified in section 6 of the VC-1 specification [1].

In Advanced profile, a bitstream data unit (BDU) is a unit of compressed data that can be parsed independently of other information at the same hierarchical level. The beginning of the BDU is signaled by an identifier called a start code. Start codes consist of a unique three-byte Start Code Prefix (SCP), and a one-byte Start Code Suffix (SCS). The SCP shall be a unique sequence of three bytes (0x000001). The SCS is used to identify the type of BDU that follows the start code. For example, the suffix of the start code before a sequence header BDU is different from the suffix of the start code before a frame BDU. Start codes shall always be byte-aligned.

An encapsulation is used to prevent emulation of start code prefixes in the bitstream. The compressed data before encapsulation is called a Raw Bitstream Decodable Unit (RBDU), while the term Encapsulated BDU (EBDU) refers to the data after encapsulation.

Note 1: There are no start codes or encapsulation for Simple or Main profile.

Note 2: When Advanced profile bitstreams are carried over ASF, start codes may be omitted for some BDUs as explained below.

The following table summarizes the differences between WMV3 and WVC1 ASF files.

VC-1 streams in ASF	Simple and Main Profile	Advanced Profile

FourCC in ASF	'WMV3' or 'wmv3'	'WVC1' or 'wvc1'
Bitstreams	Not emulated	Encapsulated
Stream headers	Header stored in the extended BITMAPINFOHEADER (or CodecPrivateData)	Header stored in the extended BITMAPINFOHEADER (or CodecPrivateData)
Data in ASF CodecPrivateData	Sequence Header	ASF binding byte + Sequence header start code + Sequence header + Entry point header start code + Entry point header
Start codes in compressed streams	No start codes	Start codes are necessary if there is more than one BDU in one ASF packet. **Note:** Start codes are optional in the cases specified later in section A.7.

The remaining sections describe these requirements in more detail.

A.3 FourCC codes for VC-1 streams in ASF

Both the Simple and Main Profiles of VC-1 have a FourCC code equal to 'WMV3' or 'wmv3'. (Either of these two FourCC codes is acceptable.) Advanced Profile has a FourCC code equal to 'WVC1' or 'wvc1'.

A.4 VC-1 stream emulation

If the FourCC is 'WVC1' or 'wvc1' (Advanced Profile VC-1), the bitstream is always in the encapsulated form (EBDU). Advanced Profile ASF bitstreams are never unencapsulated.

WMV3 bitstreams are never in encapsulated form.

A.5 VC-1 stream headers in CodecPrivateData

If the FourCC is 'WMV3' or 'wmv3', the CodecPrivateData contains nothing but the sequence header of Simple and Main profile. The syntax of the sequence header is specified in Table 263 of Annex L of the VC-1 specification. The semantics of the sequence header are specified by Annex J of the VC-1 specification.

If the FourCC is 'WVC1' or 'wvc1', the CodecPrivateData contains the following elements: One ASF binding byte, followed by the sequence header start code and sequence header, followed by an entry-point start code and entry-point header. The sequence header start code and the entry-point start code are specified in Annex F of the VC-1 specification[1]. The syntax and semantics of the sequence header are specified in section 6 of the VC-1 specification.

The syntax and semantics of the ASF binding byte are specified as follows.

A.6 ASF binding byte for Advanced Profile in ASF CodecPrivateData

When VC-1 Advanced Profile bitstreams are carried over ASF, the ASF binding byte is present before the sequence start code in CodePrivateData. The syntax and semantics of this byte are not specified in the VC-1 specification, and this byte is not required for decoding of the bitstream.

The decoder does not need to process the value of this byte to decode the bitstream. However, the presence of this byte can help the decoder perform some tasks more efficiently. For example, if this byte indicates that B frames are not present, the latency and buffering requirements of the decoder can be reduced. Thus, this byte is not really necessary but can be useful when present. The ASF binding byte is present for VC-1 Advanced Profile ('WVC1' or 'wvc1'), but will not be present for VC-1 Simple and Main profiles ('WMV3' or 'wmv3').

The syntax of this byte is as follows.

ASF_BINDING_BYTE	Number of Bits1	Descriptor
ASFBINDINGBYTE_RESERVED_BIT7	1	uimsbf
ASFBINDINGBYTE_RESERVED_BIT6	1	uimsbf
ASFBINDING_NO_INTERLACESOURCE	1	uimsbf
ASFBINDING_NO_MULTIPLESEQ	1	uimsbf
ASFBINDING_NO_MULTIPLEENTRY	1	uimsbf
ASFBINDING_NO_SLICECODE	1	uimsbf
ASFBINDING_NO_BFRAME	1	uimsbf
ASFBINDING_RESERVED_BIT0	1	uimsbf

The semantics of the ASF_Binding byte are as follows.

ASFBINDING_RESERVED_BIT0 = (BINDINGBYTE &1)

bit 0 – reserved; should be 1 always

ASFBINDING_NO_BFRAME = ((BINDINGBYTE>>1) &1)

bit 1 – when 0, b frame might be present

ASFBINDING_NO_SLICECODE = ((BINDINGBYTE>>2) &1)

bit 2 – when 0, slice might be present

ASFBINDING_NO_MULTIPLEENTRY = ((BINDINGBYTE>>3) &1)

bit 3 – when 0, multiple entry headers might be present

ASFBINDING_NO_MULTIPLESEQ = ((BINDINGBYTE>>4) &1)

bit 4 – when 0, multiple sequence headers might be present

ASFBINDING_NO_INTERLACESOURCE = ((BINDINGBYTE>>5) &1)

bit 5 – when 0, interlace sequence might be present

ASFBINDINGBYTE_RESERVED_BIT6 = ((BINDINGBYTE>>6) &1)

bit 6 – reserved

ASFBINDINGBYTE_RESERVED_BIT7 = ((BINDINGBYTE>>7) &1)

bit 7 – reserved

A.7 Start codes in ASF

Start codes are required for VC-1 Advanced Profile elementary streams. However, when they are carried within ASF, the frame start code in an ASF packet can be omitted under certain circumstances as explained below.

The rules for the presence of the start code are as follows.

1. If there is more than one BDU in a single ASF payload, start codes are present within the frame bitstream and they mark the start of each BDU. The presence of a start code at the beginning of the frame is what signals the decoder that multiple BDUs are present.
2. The exception to (1) occurs when the frame payload is coded as an interlace field picture. In this case, if the only other BDU in the frame is the second field, the start code may be omitted at the start of the frame, although it may also be present. The second field, however, must start with a byte-aligned start code. If there are any other BDUs in the ASF packet (such as slices or user data), the frame must start with a start code.

Another way of stating these rules is as follows:

1) If an ASF packet contains only a frame BDU, or if it contains a frame BDU followed by a field BDU, the frame start code is optional. If the frame start code is optional, there must not be any stuffing bytes in the packet.
2) In all other cases, the start code is mandatory. That is, when the ASF packet contains a sequence or entry-point header in addition to compressed frame information, or if there are slices, or if there is user data, all start codes are mandatory. Also, start codes are mandatory when there are stuffing bytes.

If the FourCC code of an ASF file is WMV3, there will be no start codes in compressed stream data.

Note: The rationale for this quirk was as follows.

Start codes are needed for demarcation between BDUs. In ASF, when there is only a frame BDU in the ASF payload, a start code is not required, because the ASF container can provide the separation. In this case, start codes are optional. It is easy to determine whether the start code is present by looking at the first 3 bytes, because the byte sequence 0x00 0x00 0x01 is only present for start codes in an encapsulated bitstream.

Startcodes are required in ASF only when there is a sequence header or entry-point header, or when a single ASF payload contains multiple slices or other user data. In these cases, the ASF container by itself cannot provide the necessary demarcation between individual BDUs.

Note: The WMV decoder for ASF shall handle both cases, with or without optional start codes.

A.8 References

[1] SMPTE s421M, "SMPTE STANDARD—VC-1 Compressed Video Bitstream Format and Decoding Process" (http://www.smpte.org).

Appendix B: Super-P frame video streams in ASF

B.1 Background

Super-P frame video format has cached frame (long-term) reference support in addition to all the features in the progressive format of VC-1 Advanced Profile. It is mainly used in real-time communication and video-streaming scenarios. Super-P frame video has a new error recovery architecture that uses periodical and synchronized frame caching in the encoder and decoder. Both encoder and decoder periodically cache reconstructed frames, in synchronization. When the encoder receives a packet loss event reported by the decoder, it will encode the next P frame by referring to the latest cached frame. This P frame is called a Super-P (SP) frame. The decoder then decodes this SP frame by referring to the same cached reference locally, and recovers from the packet loss. The recovery time at the decoder is therefore the interval between the packet loss event and the arrival of the SP frame, instead of the interval between the packet loss event and the arrival of the new I frame in VC-1 Advanced Profile. In Super-P frame video, there's only one cache frame, and therefore a new cache frame should simply replace the old cache frame. SP frame always uses the latest cache frame as reference. Compared with the existing error-resilient coding technologies, the key advantages of SP frame video are:

- Bandwidth is utilized more efficiently in packet loss recovery, because an SP frame can usually be coded with fewer bits than an I frame at similar quality.
- The signaling overhead required for slice-based coding is eliminated, which is important in low-bit-rate video communication.

B.2 Overview of Super-P frame videos in ASF

Super-P frame video format is a variant based on the progressive format of VC-1 Advanced Profile. The main differences between Super-P frame in ASF and VC-1 Advanced Profile in Advanced Systems Format (ASF) are:

1. Super-P frame supports progressive coding format only, and does not support interlace coding format.
2. The four-character code (FourCC) of Super-P frame videos in ASF is 'VC1S' or 'vc1s'.
3. All the BDUs must have a start code of 0x00 00 01, so there is never a missing start code for Super-P frame video in ASF.
4. Each ASF packet/payload may have a BDU in the type of 0x10 to indicate the frame type. The BDU has the layout of 0x00 00 01 10 *xx*, where *xx* indicates frame type as shown in Table B.1. If the BDU doesn't exist, frame type is inferred to be "regular frame" for any picture, I, P, or B frame, in the ASF packet/payload. If the BDU exists, it shall be the first BDU in the ASF packet/payload.

Frame type	Code word
Regular	0x00

Cache	0x01
SP	0x02
SP and Cache	0x03

Table B.1 Frame type used in Super-P frame video

All other aspects are the same as those for VC-1 Advanced Profile in ASF.

12. Revision history

This section describes the changes made in revisions of this document. Changes were not tracked prior to revision 01.20.02. The following table lists the changes by revision number.

Document revision	Changes
01.20.02	The following sections were added: • 4.12 Compatibility Object (optional, only 1) • 4.13 Advanced Content Encryption Object (optional, 0 or 1) • 7.3.2.6 ASF_Payload_Extension_System_Encryption_Sample_ID • 7.5 Pixel aspect ratio • 8.2.16 Ordering of payloads and media objects in packets • 10.7 Advanced Content Encryption Object System ID GUIDs (subsequent topics were renumbered) • 12 Revision history The following section was removed: • 3.16 Alternate Extended Content Encryption Object (optional, 0 to many) (subsequent topics were renumbered) The following sections include changes: • 3.3 - Provided additional information on valid stream numbering • 4.3 - Changed references to Advanced Mutual Exclusion Object to the correct name, Group Mutual Exclusion Object • 6.1 - Clarified that the Packet Number field references the number of the packet that contains the closest past key frame • 7.3.2.3 - Added the Interlaced repeat first field flag. Provided definitions for all fields in this section • 10.2 - Removed the GUID for the ASF_Alternate_Extended_Content_Encryption_Object • 10.3 - Corrected an error in the GUID value specified for the ASF_Media_Object_Index_Parameters_Object • 10.3 - Added the GUID for the ASF_Compatibility_Object • 10.3 and 10.13 Updated the names associated with GUIDs to maintain consistency with other names in this specification

01.20.03	End user license agreement changed to remove a restriction about using this specification to build solutions primarily designed to distribute content.
01.20.05	• 3.4. Clarified definition of Object Size field • Clarified correct usage of Replicated Data Length Type field and Offset Into Media Object Length field • 5.2.3.1. Clarified correct data size for Media Object Number, Offset Into Media Object, Replicated Data Length • 7.3.2.1. Corrected byte order for Timecode field • 7.3.2.4. Corrected layout of ASF_Payload_Extension_System_Pixel_Aspect_Ratio payload extension • 7.3.2.7. Added ASF_Payload_Extension_Degradable_JPEG payload extension • 8.2.12.1. Clarified use of NumberOfFrames attribute • 8.2.12.3. Clarified use of NumberOfFrames attribute • 10.3. Corrected GUID value for ASF_Compatibility_Object • 10.13. Added GUID value for ASF_Payload_Extension_System_Degradable_JPEG • Added Appendix A
01.20.06	• Added Appendix B