Video File Format Specification
Version 10

Adobe Systems Incorporated

Published November 2008

Contents

Introduction 1
The FLV file format 1
The F4V file format 1
What's new in Video File Format 10. 2
Speex 2
XMP Metadata 2

Chapter 1: The FLV File Format 3
The FLV header 4
The FLV file body 4
FLV tags. 5
Audio tags 6
AUDIODATA 6
AACAUDIODATA 8
Video tags 8
VIDEODATA. 9
AVCVIDEOPACKET. 10
Data tags 10
onMetaData. 14

Chapter 2: The F4V File Format 17
The F4V box 17
ftyp box. 18
moov box 19
mvhd box 19
trak box. 21
udta box 21
meta box. 21
mdia box. 22
minf box 22
stbl box 22
tkhd box 23
mdhd box 24
stsd box 25

stsc box 26
stts box. 27
ctts box 28
stco and co64 boxes 28
stss box 29
stsz box 30
chpl box 30
pdin box 31
mdat box 32
Required structure. 32
Supported media types 33
Unsupported boxes. 34

Chapter 3: F4V Metadata 35

Tag box 36
ilst box 36
Image metadata 37
Text metadata 38
  styl box 38
  hlit box 39
  hclr box 39
  krok box. 40
  dlay box 40
  drpo box 41
  drpt box 41
  href box 41
  tbox box. 42
  blnk box 43
  twrp box 43
XMP Metadata 43
  uuid box 44

Introduction

This document provides technical format information for the video file formats supported by Adobe® Flash® Player software—FLV and F4V.

Adobe seriously considers all feedback to the video file format specification. E-mail any unclear or potentially erroneous information within the specification to Adobe at flashformat@adobe.com. All such email submissions shall be subject to the Submitted Materials guidelines in the Terms of Use at www.adobe.com/misc/copyright.html.

The FLV file format

Starting with SWF files published for Flash Player 6, Flash Player can exchange audio, video, and data over RTMP connections with the Adobe Flash Media Server™. One way to feed data to Flash Media Server (and thus on to Flash Player clients) is from files in the FLV file format. Starting with SWF files published for Flash Player 7, Flash Player can also play FLV files directly with MIME type video/x-flv.

An FLV file encodes synchronized audio and video streams. The audio and video data within FLV files are encoded in the same way as audio and video within SWF files.

This document describes FLV version 1. For more information on the FLV format, see Chapter 1, “The FLV File Format,” on page 3.

The F4V file format

Starting with SWF files published for Flash Player 9 Update 3 (9,0,115,0), Flash Player can play F4V files. The F4V format is based on the format specified by ISO/IEC 14496-12: ISO base media file format. For more information on the F4V format, see Chapter 2, “The F4V File Format,” on page 17.

A large part of the F4V format involves metadata. For more information on F4V metadata, see Chapter 3, “F4V Metadata,” on page 35.

What's new in Video File Format 10

The following features are new in the Flash video file format specifications (both FLV and F4V) corresponding to Flash Player 10.

Speex

Flash Player 10 supports the open source Speex audio codec. Speex data can serve as the audio stream in an FLV file. See "AUDIODATA" on page 6.

XMP Metadata

The F4V file format supports a box type called 'uuid' which can store data formatted in Adobe's Extensible Metadata Platform (XMP™) and present it to a SWF file via ActionScript®. See "XMP Metadata" on page 43.

CHAPTER 1

The FLV File Format

1

Each tag type in an FLV file constitutes a single stream. There can be no more than one audio and one video stream, synchronized together, in an FLV file. An FLV file cannot define multiple independent streams of a single type.

Unlike SWF files, FLV files store multibyte integers in big-endian byte order. For example, as a UI16 in SWF file format, the byte sequence that represents the number 300 (0x12C) is 0x2C 0x01; as a UI16 in FLV file format, the byte sequence that represents the number 300 is 0x01 0x2C. Also, FLV files use a 3-byte integer type that is not used in SWF files: a UI24 represents an unsigned 24-bit integer.

The FLV header

All FLV files begin with the following header:

FLV header		
Field	**Type**	**Comment**
Signature	UI8	Signature byte always 'F' (0x46)
Signature	UI8	Signature byte always 'L' (0x4C)
Signature	UI8	Signature byte always 'V' (0x56)
Version	UI8	File version (for example, 0x01 for FLV version 1)
TypeFlagsReserved	UB[5]	Must be 0
TypeFlagsAudio	UB[1]	Audio tags are present
TypeFlagsReserved	UB[1]	Must be 0
TypeFlagsVideo	UB[1]	Video tags are present
DataOffset	UI32	Offset in bytes from start of file to start of body (that is, size of header)

The DataOffset field usually has a value of 9 for FLV version 1. This field is present to accommodate larger headers in future versions.

The FLV file body

After the FLV header, the remainder of an FLV file consists of alternating back-pointers and tags. They interleave as shown in the following table:

FLV file body		
Field	**Type**	**Comment**
PreviousTagSize0	UI32	Always 0
Tag1	FLVTAG	First tag
PreviousTagSize1	UI32	Size of previous tag, including its header. For FLV version 1, this value is 11 plus the DataSize of the previous tag.
Tag2	FLVTAG	Second tag
...		

FLV file body		
Field	**Type**	**Comment**
PreviousTagSizeN-1	UI32	Size of second-to-last tag
TagN	FLVTAG	Last tag
PreviousTagSizeN	UI32	Size of last tag

FLV tags

FLV tags have the following format:

FLVTAG		
Field	**Type**	**Comment**
TagType	UI8	Type of this tag. Values are: 8: audio 9: video 18: script data all others: reserved
DataSize	UI24	Length of the data in the Data field
Timestamp	UI24	Time in milliseconds at which the data in this tag applies. This value is relative to the first tag in the FLV file, which always has a timestamp of 0.
TimestampExtended	UI8	Extension of the Timestamp field to form a SI32 value. This field represents the upper 8 bits, while the previous Timestamp field represents the lower 24 bits of the time in milliseconds.
StreamID	UI24	Always 0
Data	If TagType == 8 AUDIODATA If TagType == 9 VIDEODATA If TagType == 18 SCRIPTDATAOBJECT	Body of the tag

In playback, the time sequencing of FLV tags depends on the FLV timestamps only. Any timing mechanisms built into the payload data format are ignored.

Audio tags

Audio tags are similar to the DefineSound tag in the SWF file format. Their payload data is identical except for the additional Nellymoser 8-kHz format, which is not permitted in SWF. (For information on the SWF file format, see the SWF File Format Specification at www.adobe.com/go/swf_file_format.)

AUDIODATA

AUDIODATA

Field	Type	Comment
SoundFormat (see notes following table)	UB[4] 0 = Linear PCM, platform endian 1 = ADPCM 2 = MP3 3 = Linear PCM, little endian 4 = Nellymoser 16-kHz mono 5 = Nellymoser 8-kHz mono 6 = Nellymoser 7 = G.711 A-law logarithmic PCM 8 = G.711 mu-law logarithmic PCM 9 = reserved 10 = AAC 11 = Speex 14 = MP3 8-Khz 15 = Device-specific sound	Format of SoundData Formats 7, 8, 14, and 15 are reserved for internal use AAC is supported in Flash Player 9,0,115,0 and higher. Speex is supported in Flash Player 10 and higher.
SoundRate	UB[2] 0 = 5.5-kHz 1 = 11-kHz 2 = 22-kHz 3 = 44-kHz	Sampling rate For AAC: always 3
SoundSize	UB[1] 0 = snd8Bit 1 = snd16Bit	Size of each sample. This parameter only pertains to uncompressed formats. Compressed formats always decode to 16 bits internally. 0 = snd8Bit 1 = snd16Bit

AUDIODATA

Field	Type	Comment
SoundType	UB[1] 0 = sndMono 1 = sndStereo	Mono or stereo sound For Nellymoser: always 0 For AAC: always 1
SoundData	UI8[size of sound data]	if SoundFormat == 10 AACAUDIODATA else Sound data–varies by format

Format 3, linear PCM, stores raw PCM samples. If the data is 8-bit, the samples are unsigned bytes. If the data is 16-bit, the samples are stored as little endian, signed numbers. If the data is stereo, left and right samples are stored interleaved: left - right - left - right - and so on.

Format 0 PCM is the same as format 3 PCM, except that format 0 stores 16-bit PCM samples in the endian order of the platform on which the file was created. For this reason, format 0 is not recommended for use.

Nellymoser 8-kHz and 16-kHz are special cases— 8- and 16-kHz sampling rates are not supported in other formats, and the SoundRate bits can't represent this value. When Nellymoser 8-kHz or Nellymoser 16-kHz is specified in SoundFormat, the SoundRate and SoundType fields are ignored. For other Nellymoser sampling rates, specify the normal Nellymoser SoundFormat and use the SoundRate and SoundType fields as usual.

If the SoundFormat indicates AAC, the SoundType should be set to 1 (stereo) and the SoundRate should be set to 3 (44 kHz). However, this does not mean that AAC audio in FLV is always stereo, 44 kHz data. Instead, the Flash Player ignores these values and extracts the channel and sample rate data is encoded in the AAC bitstream.

For information regarding Speex capabilities and limitations when stored in a SWF file, see the SWF File Format Specification.

AACAUDIODATA

The AAC format is supported in Flash Player 9,0,115,0 and higher.

AACAUDIODATA		
Field	**Type**	**Comment**
AACPacketType	UI8	0: AAC sequence header 1: AAC raw
Data	UI8[n]	if AACPacketType == 0 AudioSpecificConfig else if AACPacketType == 1 Raw AAC frame data

The AudioSpecificConfig is explained in ISO 14496-3. Note that it is not the same as the contents of the esds box from an MP4/F4V file. This structure is more deeply embedded.

Video tags

Video tags are similar to the VideoFrame tag in the SWF file format, and their payload data is identical. (For information on the SWF file format, see the SWF File Format Specification at www.adobe.com/go/swf_file_format.)

VIDEODATA

VIDEODATA		
Field	**Type**	**Comment**
FrameType	UB[4]	1: keyframe (for AVC, a seekable frame) 2: inter frame (for AVC, a non-seekable frame) 3: disposable inter frame (H.263 only) 4: generated keyframe (reserved for server use only) 5: video info/command frame
CodecID	UB[4]	1: JPEG (currently unused) 2: Sorenson H.263 3: Screen video 4: On2 VP6 5: On2 VP6 with alpha channel 6: Screen video version 2 7: AVC
VideoData	If CodecID == 2 H263VIDEOPACKET If CodecID == 3 SCREENVIDEOPACKET If CodecID == 4 VP6FLVVIDEOPACKET If CodecID == 5 VP6FLVALPHAVIDEOPACKET If CodecID == 6 SCREENV2VIDEOPACKET if CodecID == 7 AVCVIDEOPACKET	Video frame payload or UI8 (see note following table)

If FrameType = 5, instead of a video payload, the message stream contains a UI8 with the following meaning:

- 0 = Start of client-side seeking video frame sequence
- 1 = End of client-side seeking video frame sequence

AVCVIDEOPACKET

An AVCVIDEOPACKET carries a payload of AVC video data.

AVCVIDEOPACKET		
Field	**Type**	**Comment**
AVCPacketType	UI8	0: AVC sequence header 1: AVC NALU 2: AVC end of sequence (lower level NALU sequence ender is not required or supported)
CompositionTime	SI24	if AVCPacketType == 1 Composition time offset else 0
Data	UI8[n]	if AVCPacketType == 0 AVCDecoderConfigurationRecord else if AVCPacketType == 1 One or more NALUs (can be individual slices per FLV packets; that is, full frames are not strictly required) else if AVCPacketType == 2 Empty

See ISO 14496-12, 8.15.3 for an explanation of composition times. The offset in an FLV file is always in milliseconds.

See ISO 14496-15, 5.2.4.1 for the description of AVCDecoderConfigurationRecord. This contains the same information that would be stored in an avcC box in an MP4/FLV file.

Data tags

Data tags encapsulate single-method invocation, which is usually called on a NetStream object in Flash Player. Data tags are formed from a method name and a set of arguments.

SCRIPTDATA		
Field	**Type**	**Comment**
Objects	SCRIPTDATAOBJECT[]	Arbitrary number of SCRIPTDATAOBJECT structures
End	UI24	Always 9, also known as a SCRIPTDATAOBJECTEND

SCRIPTDATAOBJECT and SCRIPTDATAOBJECTEND

A SCRIPTDATAOBJECT record defines object data in ActionScript. Lists of SCRIPTDATAOBJECT records are terminated by using the SCRIPTDATAOBJECTEND tag.

SCRIPTDATAOBJECT

Field	Type	Comment
ObjectName	SCRIPTDATASTRING	Name of the object
ObjectData	SCRIPTDATAVALUE	Data of the object

SCRIPTDATAOBJECTEND

Field	Type	Comment
ObjectEndMarker2	UI24	Always 9

SCRIPTDATASTRING and SCRIPTDATALONGSTRING

The SCRIPTDATASTRING and SCRIPTDATALONGSTRING records are used to define strings for data tags.

The SCRIPTDATALONGSTRING record can be used to specify strings larger than 65535 characters.

SCRIPTDATASTRING

Field	Type	Comment
StringLength	UI16	String length in bytes
StringData	STRING	String data

SCRIPTDATALONGSTRING

Field	Type	Comment
StringLength	UI32	String length in bytes
StringData	STRING	String data

SCRIPTDATAVALUE

A SCRIPTDATAVALUE record represents an abstract definition of an ActionScript value or object. It can contain a list of values, objects, variables, or arrays.

SCRIPTDATAVALUE

Field	Type	Comment
Type (see notes following table)	UI8	Type of the variable: 0 = Number type 1 = Boolean type 2 = String type 3 = Object type 4 = MovieClip type 5 = Null type 6 = Undefined type 7 = Reference type 8 = ECMA array type 10 = Strict array type 11 = Date type 12 = Long string type
ECMAArrayLength	If Type = 8, UI32	Approximate number of fields of ECMA array

SCRIPTDATAVALUE

Field	Type	Comment
ScriptDataValue	If Type == 0 DOUBLE If Type == 1 UI8 If Type == 2 SCRIPTDATASTRING If Type == 3 SCRIPTDATAOBJECT[n] If Type == 4 SCRIPTDATASTRING defining the MovieClip path If Type == 7 UI16 If Type == 8 SCRIPTDATAVARIABLE[ECMAArrayLength] If Type == 10 SCRIPTDATAVARIABLE[n] If Type == 11 SCRIPTDATADATE If Type == 12 SCRIPTDATALONGSTRING	Script data values
ScriptDataValueTerminator	If Type == 3 SCRIPTDATAOBJECTEND If Type == 8 SCRIPTDATAVARIABLEEND	Terminators for Object and Strict array lists

If Type = 8 (ECMA array type), the ECMAArrayLength provides a hint to the software about how many items might be in the array. The array continues until SCRIPTDATAVARIABLEEND appears.

If Type = 10 (strict array type), the array begins with a UI32 type and contains that exact number of items. The array does not terminate with a SCRIPTDATAVARIABLEEND tag.

SCRIPTDATAVARIABLE and SCRIPTDATAVARIABLEEND

A SCRIPTDATAVARIABLE record defines variable data in ActionScript. Lists of SCRIPTDATAVARIABLE records are terminated by using the SCRIPTDATAVARIABLEEND tag.

SCRIPTDATAVARIABLE		
Field	**Type**	**Comment**
VariableName	SCRIPTDATASTRING	Name of the variable
VariableData	SCRIPTDATAVALUE	Data of the variable

SCRIPTDATAVARIABLEEND		
Field	**Type**	**Comment**
VariableEndMarker1	UI24	Always 9

SCRIPTDATADATE

A SCRIPTDATADATE record defines a particular date and time.

SCRIPTDATADATE		
Field	**Type**	**Comment**
DateTime	DOUBLE	Number of milliseconds since Jan 1, 1970 UTC.
LocalDateTimeOffset	SI16	Local time offset in minutes from UTC. For time zones located west of Greenwich, UK, this value is a negative number. Time zones east of Greenwich, UK, are positive.

onMetaData

An FLV file can contain metadata with an "onMetaData" marker. Various stream properties are available to a running ActionScript program via the `NetStream.onMetaData` property. The available properties differ depending on the software used. Common properties include:

- `duration`: a DOUBLE indicating the total duration of the file in seconds
- `width`: a DOUBLE indicating the width of the video in pixels
- `height`: a DOUBLE indicating the height of the video in pixels
- `videodatarate`: a DOUBLE indicating the video bit rate in kilobits per second

- `framerate`: a DOUBLE indicating the number of frames per second
- `videocodecid`: a DOUBLE indicating the video codec ID used in the file (see "Video tags" on page 8 for available CodecID values)
- `audiosamplerate`: a DOUBLE indicating the frequency at which the audio stream is replayed
- `audiosamplesize`: a DOUBLE indicating the resolution of a single audio sample
- `stereo`: a BOOL indicating whether the data is stereo
- `audiocodecid`: a DOUBLE indicating the audio codec ID used in the file (see "Audio tags" on page 6 for available SoundFormat values)
- `filesize`: a DOUBLE indicating the total size of the file in bytes

CHAPTER 2

The F4V File Format

2

Flash Player Update 3 (9,0,115,0) and higher can play F4V files. The F4V format is based on the format specified by ISO/IEC 14496-12: ISO base media file format.

Unlike SWF files, F4V files store multibyte integers in big-endian byte order. For example, as a UI16 in SWF file format, the byte sequence that represents the number 300 (0x12C) is 0x2C 0x01; as a UI16 in F4V file format, the byte sequence that represents the number 300 is 0x01 0x2C.

This chapter discusses all aspects of the F4V format except metadata. For information on metadata, see Chapter 3, "F4V Metadata," on page 35.

The F4V box

The fundamental building block of an F4V file is a box which has the following BOX format:

F4V box

Field	Type	Comment
Header	BOXHEADER	A consistent header that all boxes have
Payload	UI8[]	A number of bytes, the length of which is defined by the BOXHEADER

Each box structure begins with a BOXHEADER structure:

BOXHEADER

Field	Type	Comment
TotalSize	UI32	The total size of the box in bytes, including this header
BoxType	UI32	The type of atom
ExtendedSize	If TotalSize equals 1 UI64	The total 64-bit length of the box in bytes, including this header

Many boxes are well under 4 gigabytes in length and can store their size in the TotalSize field. The format also supports very large boxes by setting the 32-bit TotalSize field to 1 and storing a 64-bit size in ExtendedSize.

Each box is identified with a 32-bit type. For most boxes, this 32-bit type doubles as a human-readable four-character ASCII code or FourCC, such as 'moov' (0x6D6F6F76) and 'mdat' (0x6D646174).

The box payload immediately follows the box header. The size of the payload in bytes is equal to the total size of the box minus either 8 bytes or 16 bytes, depending on the size of the header.

For more information, see section 4.2 of ISO/IEC 14496-12.

ftyp box

The F4V format is based on the ISO MP4 format, which in turn is based on the Apple QuickTime container format. The subsets of the format support different features. The ftyp box helps identify the features that a program must support to play a particular file.

Flash Player does not enforce any restrictions with respect to ftyp boxes. The program tries to play any file it is given, within the restrictions of the codec types it supports.

ftyp box

Field	Type	Comments
Header	BOXHEADER	BoxType = 'ftyp' (0x66747970)
MajorBrand	UI32	The primary brand identifier
MinorVersion	UI32	The secondary brand identifier
CompatibleBrands	UI32[]	Arbitrary number of compatible brands, until the end of the box

For more information, see section 4.3 of ISO/IEC 14496-12.

moov box

An F4V file must contain one and only one moov box. The moov box is effectively the "header" of an F4V file. The moov box itself contains one or more other boxes, which in turn contain other boxes which define the structure of the F4V data.

moov box

Field	Type	Comment
Header	BOXHEADER	BoxType = 'moov' (0x6D6F6F76)
Boxes	BOX[]	Many other boxes which define the structure

For more information, see section 8.1 of ISO/IEC 14496-12.

mvhd box

An F4V file must contain one and only one mvhd box. The mvhd box is contained within a moov box and defines playback information that applies to the entire F4V file.

mvhd box

Field	Type	Comment
Header	BOXHEADER	BoxType = 'mvhd' (0x6D766864)
Version	UI8	Either 0 or 1
Flags	UI24	Reserved, set to 0
CreationTime	if Version == 0 SI32 if Version == 1 SI64	The creation time of the F4V file, expressed as seconds elapsed since midnight, January 1, 1904 (UTC)
ModificationTime	if Version == 0 SI32 if Version == 1 SI64	The last modification time of the F4V file, expressed as seconds elapsed since midnight, January 1, 1904 (UTC)
TimeScale	SI32	Specifies the time coordinate system for the entire F4V file; for example, 100 indicates the time units are 1/100 second each

mvhd box		
Field	**Type**	**Comment**
Duration	if Version == 0 SI32 if Version == 1 SI64	The total length of the F4V file presentation with respect to the TimeScale; this value is also the duration of the longest track in the file
Rate	SI32	The preferred rate of playback, expressed as a fixed point 16.16 number (commonly 0x00010000 = 1.0, or normal playback rate)
Volume	SI16	The master volume of the file, expressed as a fixed point 8.8 number (commonly 0x0100 = 1.0, or full volume)
Reserved	UI16	Reserved, set to 0
Reserved	UI32[2]	Reserved, set to 0
Matrix	UI32[9]	Transformation matrix for the F4V file; F4V restricts these values to {0x00010000, 0, 0, 0, 0x00010000, 0, 0, 0, 0x40000000}
Reserved	UI32[6]	Reserved, set to 0
NextTrackID	UI32	The ID of the next track to be added to the presentation; this value may not be 0 but might be all 1's to indicate an undefined state

For more information, see section 8.3 of ISO/IEC 14496-12.

trak box

An F4V file must contain one or more trak boxes. Each trak box is contained within a moov box. Each trak box corresponds to an individual media track within the F4V file and contains other boxes which further define the properties of the media track.

trak box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'trak' (0x7472616B)
Boxes	BOX[]	Arbitrary number of boxes that define the media track

For more information, see section 8.4 of ISO/IEC 14496-12.

udta box

The optional udta box defines free-form user data. Flash Player does not care what is contained in this box. An F4V file can contain at most one udta box.

udta box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'udta' (0x75647461)
UserData	UI8[]	Free-form user data

For more information, see section 8.27 of ISO/IEC 14496-12.

meta box

The optional meta box can contain a variety of other boxes that carry metadata. An F4V file can contain at most one meta box.

meta box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'meta' (0x6D657461)
Boxes	BOX[]	Arbitrary number of boxes that define the file's metadata

For more information, see section 8.44.1 of ISO/IEC 14496-12.

mdia box

Each trak box must contain one and only one mdia box. The mdia box contains boxes that define media track properties.

mdia box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'mdia' (0x6D646961)
Boxes	BOX[]	Arbitrary number of boxes that define media track properties

For more information, see section 8.7 of ISO/IEC 14496-12.

minf box

Each mdia box must contain one and only one minf box. The minf box contains boxes that define the track's media information.

minf box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'minf' (0x6D696E66)
Boxes	BOX[]	Arbitrary number of boxes that define the track's media information

For more information, see section 8.10 of ISO/IEC 14496-12.

stbl box

Each minf box must contain one and only one stbl box. The stbl box contains boxes that define properties about the samples that make up a track.

stbl box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'stbl' (0x7374626C)
Boxes	BOX[]	Arbitrary number of boxes that define properties about the track's constituent samples

For more information, see section 8.14 of ISO/IEC 14496-12.

tkhd box

Each trak box must contain one and only one tkhd box. The tkhd box describes the main properties of a track.

tkhd box

Field	Type	Comment
Header	BOXHEADER	BoxType = 'tkhd' (0x746B6864)
Version	UI8	Versions 0 and 1 are defined
Flags	UI24	Bit 0: this bit is set if the track is enabled Bit 1 = this bit is set if the track is part of the presentation Bit 2 = this bit is set if the track should be considered when previewing the F4V file
CreationTime	if version == 0 UI32 if version == 1 UI64	The creation time of the track, expressed as seconds elapsed since midnight, January 1, 1904 (UTC)
ModificationTime	if version == 0 UI32 if version == 1 UI64	The last modification time of the track, expressed as seconds elapsed since midnight, January 1, 1904 (UTC)
TrackID	UI32	The track's unique identifier
Reserved	UI32	Reserved, set to 0
Duration	if version == 0 UI32 if version == 1 UI64	The duration of the track, expressed in the TimeScale defined in the mvhd box for this track
Reserved	UI32[2]	Reserved, set to 0
Layer	SI16	The position if the front to back ordering of tracks; this value is expected to be 0 for F4V files
AlternateGroup	SI16	0

tkhd box		
Field	**Type**	**Comment**
Volume	SI16	If the track is audio, this value is set to 0x0100 (fixed point 8.8 number representing 1.0), otherwise, it is 0
Reserved	SI16	Reserved, set to 0
TransformMatrix	SI32[9]	A matrix of fixed point 16x16 values defining a perspective transform; this matrix is restricted to the following values: {0x00010000, 0, 0, 0, 0x00010000, 0, 0, 0, 0x40000000}
Width	SI32	Applicable to a video track
Height	SI32	Applicable to a video track

For more information, see section 8.5 of ISO/IEC 14496-12.

mdhd box

A mdia box must contain one and only one mdhd box. The mdhd box describes properties about a media track.

mdhd box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'mdhd' (0x6D646864)
Version	UI8	Version 0 and 1 are supported
Flags	UI24	Reserved, set to 0
CreationTime	if version == 0 UI32 if version == 1 UI64	The creation time of the box, expressed as seconds elapsed since midnight, January 1, 1904 (UTC)
ModificationTime	if version == 0 UI32 if version == 1 UI64	The last modification time of the box, expressed as seconds elapsed since midnight, January 1, 1904 (UTC)
TimeScale	UI32	The base clock tick frequency that this track uses for timing its media

mdhd box

Field	Type	Comment
Duration	if version == 0 UI32 if version == 1 UI64	The total duration of this track, measured in reference to the TimeScale
Pad	UB[1]	Padding, set to 0
Language	UB[5][3]	3-character code specifying language (see ISO 639-2/T); each character is interpreted as 0x60 + (5 bit) code to yield an ASCII character
Reserved	SI16	Reserved, set to 0

For more information, see section 8.8 of ISO/IEC 14496-12.

stsd box

The stsd box defines the sample description for a sample table. Each stbl box must contain one and only one stsd box.

An stsd box can contain multiple descriptions for a track, one for each media type contained in the track.

stsd box

Field	Type	Comment
Header	BOXHEADER	BoxType = 'stsd' (0x73747364)
Version	UI8	Expected to be 0
Flags	UI24	None defined, set to 0
Count	UI32	The number of entries
Descriptions	DESCRIPTIONRECORD[Count]	An array of records whose types vary depending on whether the track contains audio or video data

A SAMPLEDESCRIPTION record has the following layout:

SAMPLEDESCRIPTION		
Field	**Type**	**Comment**
DescriptionLength	UI32	The length of the description record
Type	UI32	The type of the description; this value is often 4 human-readable ASCII characters that also double as the track's codec identifier
Description	UI8[DescriptionLength-8]	Per-codec configuration data

For more information, see section 8.16 of ISO/IEC 14496-12.

stsc box

The stsc box defines the sample-to-chunk mapping in the sample table of a media track. An stbl box must contain one and only one stsc box.

stsc box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'stsc' (0x73747363)
Version	UI8	Expected to be 0
Flags	UI24	Reserved, set to 0
Count	UI32	The number of STSCRECORD entries
Entries	STSCRECORD[Count]	An array of STSCRECORD structures

Each STSCRECORD has the following format:

STSCRECORD		
Field	**Type**	**Comment**
FirstChunk	UI32	The first chunk that this record applies to
SamplesPerChunk	UI32	The number of consecutive samples that this record applies to
SampleDescIndex	UI32	The sample description that describes this sequence of chunks

For more information, see section 8.18 of ISO/IEC 14496-12.

stts box

The stts box defines the time-to-sample mapping for a sample table. Each stbl box must contain one and only one stts box.

stts box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'stts' (0x73747473)
Version	UI8	Expected to be 0
Flags	UI24	None defined, set to 0
Count	UI32	The number of STTSRECORD entries
Entries	STTSRECORD[Count]	An array of STTSRECORD structures

Each STTSRECORD has the following format:

STTSRECORD		
Field	**Type**	**Comment**
SampleCount	UI32	The number of consecutive samples that this STTSRECORD applies to
SampleDelta	UI32	Sample duration

For more information, see section 8.15.2 of ISO/IEC 14496-12.

ctts box

The optional ctts box defines the composition time to sample mapping for a sample table. An stbl box can contain at most one ctts box.

ctts box

Field	Type	Comment
Header	BOXHEADER	BoxType = 'ctts' (0x63747473)
Count	UI32	The number of CTTSRECORD entries
Entries	CTTSRECORD[Count]	An array of CTTSRECORD structures

Each CTTSRECORD has the following structure:

CTTSRECORD

Field	Type	Comment
SampleCount	UI32	The number of consecutive samples that this CTTSRECORD applies to
SampleOffset	UI32	For each sample specified by the SampleCount field, this field contains a positive integer that specifies the composition offset from the decoding time

Samples are not always composed (presented to the user) at the time of decoding. The ctts box contains offsets from the decoding time when samples are meant to be presented to the user.

For more information, see section 8.15.3 of ISO/IEC 14496-12.

stco and co64 boxes

The stco and co64 boxes define chunk offsets for each chunk in a sample table. Each sample table must contain one and only one box of either the stco or co64 type.

stco and co64 boxes

Field	Type	Comment
Header	BOXHEADER	BoxType = 'stco' (0x7374636F) or 'co64' (0x636F3634)
Version	UI8	Expected to be 0
Flags	UI24	No flags defined, set to 0

stco and co64 boxes

Field	Type	Comment
OffsetCount	UI32	The number of offsets in the Offsets table
Offsets	if BoxType == 'stco' UI32[OffsetCount] else if BoxType == 'co64' UI64[OffsetCount]	A table of absolute chunk offsets within the file

For more information, see section 8.19 of ISO/IEC 14496-12.

stss box

The optional stss box specifies which samples within a sample table are sync samples. Sync samples are defined as samples that are safe to seek to. If the track is a video track, sync samples are the keyframes/intraframes that do not rely on any data from any other frames.

An stbl box can contain at most one stss box. If the stbl box doesn't contain an stss box, all samples in the track are treated as sync samples.

stss box

Field	Type	Comment
Header	BOXHEADER	BoxType = 'stss' (0x73747373)
Version	UI8	Expected to be 0
Flags	UI24	No flags defined, set to 0
SyncCount	UI32	The number of entries in SyncTable
SyncTable	UI32[SyncCount]	A table of sample numbers that are also sync samples; the table is sorted in ascending order of sample numbers

For more information, see section 8.20 of ISO/IEC 14496-12.

stsz box

The stsz box specifies the size of each sample in a sample table. An stsz atom must contain one and only one stsz box.

stsz box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'stsz' (0x7374737A)
Version	UI8	Expected to be 0
Flags	UI24	No flags defined, set to 0
ConstantSize	UI32	If all samples have the same size, this field is set with that constant size; otherwise it is 0
SizeCount	UI32	The number of entries in SizeTable
SizeTable	if ConstantSize == 0 UI32[SizeCount]	A table of sample sizes; if ConstantSize is 0, this table is empty

For more information, see section 8.17.2 of ISO/IEC 14496-12.

chpl box

The optional chpl box allows an F4V file to specify individual chapters along the main timeline of an F4V file. The information in this box is provided to ActionScript. The chpl box occurs within a moov box.

chpl box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'chpl' (0x6368706C)
Version	UI8	Expected to be 0
Flags	UI24	Reserved, set to 0
Count	UI8	The number of entries in the Chapters array
Chapters	CHAPTERRECORD[Count]	An array of timestamps along the timeline; each indicates the beginning of a new chapter

Each CHAPTERRECORD has the following layout:

CHAPTERRECORD		
Field	**Type**	**Comment**
Timestamp	UI64	The absolute timestamp of the chapter, in reference to the master timescale and timeline of the F4V file
TitleSize	UI8	The length of the Title string
Title	UI8[TitleSize]	The chapter title

pdin box

The optional pdin box defines information about progressive download. A file can contain one pdin box at the top level of the box hierarchy. The information in this box is provided to ActionScript.

The payload of a pdin box provides hints to software about how much data to download before the software can safely begin playback.

pdin box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = 'pdin' (0x7064696E)
Version	UI8	Expected to be 0
Flags	UI24	Reserved, set to 0
RateDelay	RATEDELAYRECORD[]	Populated until the end of the box

Each RATEDELAYRECORD has the following layout:

RATEDELAYRECORD		
Field	**Type**	**Comment**
BitRate	UI32	The bit rate (in bytes/second) to be considered for this record
InitialDelay	UI32	The number of milliseconds to delay before beginning playback at this bit rate

For more information, see section 8.43 of ISO/IEC 14496-12.

mdat box

An mdat box contains the media data payload for the F4V file. An F4V file must contain one and only one mdat box. The mdat box occurs at the top level of an F4V file, along with the moov box.

The mdat box cannot be understood on its own, which is why a moov box must be present in the file as well.

mdat box

Field	Type	Comment
Header	BOXHEADER	BoxHeader = 'mdat' (0x6D646174)
Payload	UI8[n]	n bytes of media data, the structure of which is defined in the file's moov box

For more information, see section 8.2 of ISO/IEC 14496-12

Required structure

Flash Player expects a valid F4V file to begin with the one of the following top-level boxes:

- ftyp (see "ftyp box" on page 18)
- moov (see "moov box" on page 19)
- mdat (see "mdat box" on page 32)

Many tools that create these files place an mdat box at the front of the file. Before Flash Player can use the file, it is necessary to change the order of boxes in the file. Some tools store an ftyp box followed immediately by an mdat box. In these situations, it is still recommended to use a post-processing step to move the moov box to the front of the file (just after the ftyp box).

Supported media types

The following table describes the media types that Flash Player plays back when the media is encapsulated inside an F4V file.

Media type	Comments
GIF	A media type of gif (0x67696620) denotes a still frame of video data compressed using the CompuServe GIF format. The space character, hex 0x20, is included to make a complete four-character code.
PNG	A media type of png (0x706E6720) denotes a still frame of video data compressed using the standard PNG format. The space character, hex 0x20, is included to make a complete four-character code.
JPEG	A media type of jpeg (0x6A706567) denotes a still frame of video data compressed using the standard JPEG format.
Text	A media type of either text (0x74657874) or tx3g (0x74783367) indicates that the track contains textual data that is made available via ActionScript.
AMF0	A media type of amf0 (0x616D6630) indicates that the track contains data corresponding to the original version of the ActionScript Message Format (AMF).
AMF3	A media type of amf3 (0x616D6633) indicates that the track contains data corresponding to the ActionScript Message Format (AMF) version 3.
H.264	A media type of H264 (0x48323634), h264 (0x68323634), or avc1 (0x61766331) indicates that the track is encoded with H.264 video. Flash Player supports the following H.264 video profiles: • 0: supported for older media that neglects to set profile • 66: baseline • 77: extended • 88: main • 100: YUV 4:2:0, 8 bits/sample; a.k.a. “High” • 110: YUV 4:2:0, 10 bits/sample; a.k.a. “High 10” • 122: YUV 4:2:2, 10 bits/sample; a.k.a. “High 4:2:2” • 144: YUV 4:4:4, 12 bits/sample; a.k.a. “High 4:4:4”

Media type	Comments
MP3	A media type of .mp3 (0x2E6D7033) indicates that the track contains MP3 audio data. The dot character, hex 0x2E, is included to make a complete four-character code.
AAC (Flash Player 9,0,115,0 and higher)	A media type of mp4a (0x6D703461) indicates that the track is encoded with AAC audio. Flash Player supports the following AAC profiles, denoted by their object types: • 1: main profile • 2: low complexity, a.k.a. LC • 5: high efficiency/scale band replication, a.k.a. HE/SBR

An avcC box occurs inside the stsd box of a sample table when the video codec is H.264, and contains initialization data that an H.264 decoder requires to decode the stream. Bytes 1 and 3 after the BOXHEADER contain the profile and level, respectively, for the AVC data. For more information about the remainder of the avcC box, see section 5.3.4.1 of ISO/IEC 14496-15.

An esds box occurs inside the stsd box of a sample table when the action codec is AAC, and contains initialization data that an AAC decoder requires to decode the stream. See ISO/IEC 14496-3 for more information about the structure of this box.

Unsupported boxes

Many box types are described in the formal ISO specification, as well as in the original Apple QuickTime specification, that Flash Player does not acknowledge. Flash Player might still play files with these box types, but gracefully disregards these boxes and their contents.

CHAPTER 3

F4V Metadata

3

When Flash Player loads an F4V file, various stream properties are made available to a running ActionScript program via the `NetStream.onMetaData` property. The available properties differ depending on the software used. These properties are:

- `duration`: a DOUBLE indicating the total length of the movie in seconds
- `moovposition`: a DOUBLE indicating the absolute offset of the moov box within the F4V file; this property is useful for determining if the file will load progressively
- `videocodecid`: a STRING with four characters that define the video codec used, if video is present and is encoded with a codec that Flash Player can decode
- `width`: a DOUBLE indicating the width of the video, if video is present and is encoded with a codec that Flash Player can decode
- `height`: a DOUBLE indicating the height of the video, if video is present and is encoded with a codec that Flash Player can decode
- `avcprofile`: a DOUBLE indicating the AVC profile that the video conforms to, if video is present and is encoded with AVC/H.264
- `avclevel`: a DOUBLE indicating the AVC level that the video conforms to, if video is present and is encoded with AVC/H.264
- `videoframerate`: a DOUBLE indicating the average video frame rate of the video, if video is present and is encoded with a codec that Flash Player can decode
- `audiocodecid`: a STRING with four characters that define the audio codec used, if audio is present and is encoded with a codec that Flash Player can decode

Tag box

The F4V file format supports an assortment of optional tag boxes that can occur within a moov box. An FLV file can contain up to 256 tags (including the tags in these boxes and the tags defined in an ilst box).

Tag box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxType = one of several
Version	UI8	Must be 0
Flags	UI24	Reserved, set to 0
LanguageCode	UI16	ISO-639-2/T two-letter codes
TagString	UI8[n]	n = remaining size of the box when the tag is reached; the maximum size that the Player honors for a TagString is 65535 bytes

These tags can occur one level beneath a moov box. Recognized tag types include 'auth' (0x61757468), 'titl' (0x7469746C), 'dscp' (0x64736370), and 'cprt' (0x63707274).

ilst box

An ilst box occurs inside a meta box and contains an arbitrary number of metadata tags that are available to ActionScript. An FLV file can contain up to 256 tags (including the tags in this box and in the auth, dscp, and cprt tag box types).

ilst box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'ilst' (0x696C7374)
TagCount	UI32	The number of tags enumerated in the ilst box
Tags	TAGRECORD[TagCount]	A number of TAGRECORD entries

Each TAGRECORD has the following layout:

TAGRECORD		
Field	**Type**	**Comment**
TagLength	UI32	The total length of the TAGRECORD, including this length field
TagName	UI8[4]	4 bytes indicating the name of the tag; these bytes usually come from the human-readable ASCII set, but not always
DataLength	UI32	The total length of the data portion of the TAGRECORD
DataTag	UI8[4]	The 4 bytes 'd', 'a', 't', and 'a' to indicate the data portion of the TAGRECORD
DataType	UI32	Specifies the type of data in the data payload of the TAGRECORD
Reserved	UI32	Reserved, set to 0
Payload	UI8[]	An arbitrary number of bytes occupying the remainder of the TAGRECORD; the precise payload format is dependent on the DataType

The supported values for the DataType are:

- 0: custom data; in the case of 'trkn' and 'disk' tag types, the data payload is interpreted as a single UI32
- 1: text data
- 13, 14: binary data
- 21: generic data

Image metadata

If an F4V sample is an image type (GIF, PNG, or JPEG), the data is made available to a running ActionScript program through the `onImageData` property. The following properties are present:

- `trackid`: a DOUBLE indicating the track that this sample belongs to
- `data`: a BYTEARRAY containing the compressed image data (that is, the original JPEG, PNG or GIF file data)

Text metadata

Text samples ('text' or 'tx3g') can contain a wide range of metadata boxes whose contents are exposed to a running ActionScript program through the `onTextData` property.

styl box

A styl box carries text style specifications. This information is exposed to ActionScript via the `style` property.

styl box

Field	Type	Comment
Header	BOXHEADER	BoxHeader = 'styl' (0x7374796C)
Count	UI16	The number of entries in the Styles array
Styles	STYLERECORD[Count]	An array of STYLERECORD structures; each is exposed as an ActionScript object

An individual STYLERECORD has the following layout:

STYLERECORD

Field	Type	Comment
StartChar	UI16	The first character to which this STYLERECORD applies; exposed to ActionScript via a DOUBLE property named `startchar`
EndChar	UI16	The last character to which this STYLERECORD applies; exposed to ActionScript via a DOUBLE property named `endchar`
FontID	UI16	The font ID to use for this style; exposed to ActionScript via a DOUBLE property named `fontid`
FaceStyleFlags	UI8	Exposed to ActionScript via a DOUBLE property named `facestyleflags`

STYLERECORD		
Field	**Type**	**Comment**
FontSize	UI8	The size to use for the font; exposed to ActionScript via the property `fontsize`
TextColor	UI32	The RGBA color for the text; exposed to ActionScript via the property `textcolor`

hlit box

An hlit box specifies a range of text to be highlighting. This information is exposed to ActionScript via the `highlight` property.

hlit box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'hlit' (0x686C6974)
StartChar	UI16	The first character to highlight; exposed to ActionScript via a DOUBLE property named `startchar`
EndChar	UI16	The final character to highlight; exposed to ActionScript via a DOUBLE property named `endchar`

hclr box

An hclr box specifies the highlight color for text. This information is exposed to ActionScript via the `highlightcolor` property.

hclr box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'hclr' (0x68636C72)
HighlightColor	UI16[3]	A three-element array that holds values for red, green, and blue components in that order; exposed to ActionScript via a DOUBLE property named `highlightcolor`

krok box

A krok box specifies karaoke metadata. This information is exposed to ActionScript via the `karaoke` property. Times are expressed in timescale units in relation to the track.

krok box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'krok' (0x6B726F6B)
StartTime	UI32	Exposed to ActionScript via a DOUBLE property named `starttime`
Count	UI16	The number of entries in the KaraokeRecords array
KaraokeRecords	KARAOKEREC[Count]	An array of KARAOKEREC structures; each is exposed to ActionScript as an object

An individual KARAOKEREC has the following structure:

KARAOKEREC		
Field	**Type**	**Comment**
EndTime	UI32	Exposed to ActionScript via a DOUBLE property named `endtime`
StartChar	UI16	Exposed to ActionScript via a DOUBLE property named `startchar`
EndChar	UI16	Exposed to ActionScript via a DOUBLE property named `endchar`

dlay box

A dlay box specifies a scroll delay. This information is exposed to ActionScript via the `scrolldelay` property, expressed in timescale units in relation to the track.

dlay box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'dlay' (0x646C6179)
ScrollDelay	UI32	Exposed to ActionScript via a DOUBLE property named `scrolldelay`

drpo box

A drpo box specifies drop shadow offset coordinates for text.

drpo box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'drpo' (0x6472706F)
DropShadowOffsetX	UI16	Exposed to ActionScript via a DOUBLE property named `dropshadowoffsetx`
DropShadowOffsetY	UI16	Exposed to ActionScript via a DOUBLE property named `dropshadowoffsety`

drpt box

A drpt box specifies drop shadow alpha value.

drpt box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'drpt' (0x64727074)
DropShadowAlpha	UI16	A 16-bit alpha value; exposed to ActionScript via a DOUBLE property named `dropshadowalpha`

href box

An href box specifies a hypertext link with ALT text over a text range. This information is exposed to ActionScript via the `hypertext` property.

href box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'href' (0x68726566)
StartChar	UI16	The beginning character of the text range; exposed to ActionScript via a DOUBLE property named `startchar`
EndChar	UI16	The last character of the text range; exposed to ActionScript via a DOUBLE property named `endchar`

href box		
Field	**Type**	**Comment**
URLSize	UI8	The length of the URL string
URL	UI8[URLSize]	The URL string; exposed to ActionScript via a STRING property named `url`
ALTSize	UI8	The length of the ALT string
ALT	UI8[ALTSize]	The ALT string which is displayed when the user's mouse hovers over the link; exposed to ActionScript via a STRING property named `alt`

tbox box

A tbox box defines the coordinates for a text box. This information is exposed to ActionScript via the `textbox` property.

tbox box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'tbox' (0x74626F78)
Top	UI16	The top pixel coordinate; exposed to ActionScript via a DOUBLE property named `top`
Left	UI16	The left pixel coordinate; exposed to ActionScript via a DOUBLE property named `left`
Bottom	UI16	The bottom pixel coordinate; exposed to ActionScript via a DOUBLE property named `bottom`
Right	UI16	The right pixel coordinate; exposed to ActionScript via a DOUBLE property named `right`

blnk box

A blnk box specifies a range of text to set blinking. This information is exposed to ActionScript via the `blink` property.

blnk box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'blnk' (0x626C6E6B)
StartChar	UI16	The first character in the blinking range; exposed to ActionScript via a DOUBLE property named `startchar`
EndChar	UI16	The ending character in the blinking range; exposed to ActionScript via a DOUBLE property named `endchar`

twrp box

A twrp box sets the wrap flag for text.

twrp box		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'twrp' (0x74777270)
WrapFlag	UI8	A boolean that is nonzero if the text should wrap; exposed to ActionScript via a DOUBLE property named `wrapflag`

XMP Metadata

Beginning in version 10, Flash Player can load XMP data embedded in an F4V file. XMP is Adobe's Extensible Metadata Platform. For more information, see www.adobe.com/go/xmp.

uuid box

A uuid box exists at the top level of an F4V file and is the method by which the file can communicate XMP metadata to a SWF movie via ActionScript.

uuid		
Field	**Type**	**Comment**
Header	BOXHEADER	BoxHeader = 'uuid' (0x75756964)
UUID	UI8[16]	A 16-byte (128-bit) universally unique identifier (UUID). The UUID that must appear here consists of these hexadecimal bytes: BE 7A CF CB 97 A9 42 A8 9C 71 99 94 91 E3 AF AC.
XMPMetadata	UI8[]	XMP metadata, formatted according to the XMP metadata standard; exposed to ActionScript via a STRING property named `data`

Note that the maximum allowed size of an XMP metadata box is 64 megabytes.