SMPTE STANDARD

ACES Image Container File Layout

Table of Contents

Copyright © 2013 by THE SOCIETY OF
MOTION PICTURE AND TELEVISION ENGINEERS
3 Barker Avenue, White Plains, NY 10601
(914) 761-1100

Approved
May 10, 2013

List of Tables

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in Part XIII of its Operations Manual.

SMPTE ST 2065-4 was prepared by Technology Committee 31FS.

Intellectual Property

At the time of publication, no notice had been received by SMPTE claiming patent rights essential to the implementation of this standard. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

Introduction

The ACES image container is a digital data file format that is designed for storing the individual frames of a motion picture during production and for archival. During production, it is important to maintain the highest possible image quality, and to minimize generational loss as images are repeatedly extracted from files, processed, and stored in new files. Fast random access to individual frames is highly desirable. Storage space and data transmission bandwidth limitations tend to be of secondary concern.

The ACES image container is not meant for distribution of motion pictures to theaters and home viewers, where representation of an entire movie as a single file, compact storage and data security take precedence.

Each ACES image container file contains a single monoscopic or stereoscopic image. Moving images are represented as sequences of image container files, with a separate image container file for each frame. This allows each frame to be read or written individually, without accessing other frames or any kind of enclosing image sequence container.

The ACES image container file format is derived from the popular OpenEXR file format, and is readable by many OpenEXR readers.

1 Scope

This Standard specifies the layout of, and metadata in, files containing images that conform to SMPTE ST 2065-1, Academy Color Encoding Specification (ACES).

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "NOTE" or "EXAMPLE"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword "reserved" indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; followed by formal languages; then figures; and then any other language forms.

3 Normative References

Note: All references in this document to other SMPTE documents use the current numbering style (e.g. SMPTE ST 12-1:2008) although, during a transitional phase, the document as published (printed or PDF) may bear an older designation (such as SMPTE 12M-1-2008). Documents with the same root number (e.g. 12-1) and publication year (e.g. 2008) are functionally identical.

The following standards contain provisions that, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

IEEE Std 754-2008, Standard for Floating-Point Arithmetic

IETF RFC 1321:1992, The MD5 Message-Digest Algorithm (Proposed Standard)

IETF RFC 4122:2005, A Universally Unique Identifier (UUID) URN Namespace (Proposed Standard)

ISO 11664-1:2007 (CIE S 014-1/E:2006), Colorimetry — Part 1: CIE Standard Colorimetric Observers

SMPTE ST 12-1:2008, Television — Time and Control Code

SMPTE ST 254:2008, Motion-Picture Film (35-mm) — Manufacturer-Printed Latent Image Identification Information

SMPTE ST 268:2003, File Format for Digital Moving-Picture Exchange (DPX), Version 2.0

SMPTE ST 331:2011, Element and Metadata Definitions for the SDTI-CP

SMPTE ST 2065-1:2012, Academy Color Encoding Specification (ACES)

The Unicode Consortium. The Unicode Standard, Version 6.0.0, (Mountain View, CA: The Unicode Consortium, 2011. ISBN 978-1-936213-01-6)

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

4.1
ACES image container file
file compliant with this standard

4.2
alpha opacity
proportion by which a pixel obscures the pixel behind it (from ISO/IEC 9592-1:1997)

4.3
byte order
ordering of bytes for multi-byte data primitives within a file, such as 4-byte integers, or 8-byte floats

4.4
channel
collection of all values of the same name in an image or in a part of an image

4.5
compositing
combining two images into a single image (from ISO/IEC 15444-2:2004)

4.6
editor
device that is both a compliant reader and a compliant writer and that updates files, or reads files and writes new files with derived contents

4.7
exposure time
time from the start of light exposure integration at a point on the sensor to the end of light exposure integration at that same point on the sensor

4.8
little-endian byte order
byte order having increasing numerical significance with increasing byte address (from the least significant byte to the most significant byte)

4.9
monoscopic image
image containing a single view of the depicted scene, intended to be presented to the viewer on a 2D display device

4.10
"over" compositing operation
compositing, where a foreground RGB color F with an associated alpha value A over a background RGB color B, will produce a combined color C, where C = F + (1-A) * B (from Porter-Duff, 1984)

4.11
pixel aspect ratio
pixel width divided by pixel height when the image is displayed with the intended aspect ratio, where the pixel width is the distance between the centers of two horizontally adjacent pixels on the display, and the pixel height is the distance between the centers of two vertically adjacent pixels on the display

4.12
pixel space
two-dimensional Cartesian coordinate system having signed integer coordinates with x coordinates increasing from left to right and y coordinates increasing from top to bottom

4.13
reader
device consuming files compliant with this standard

4.14
recorder
device that stores image data and image metadata received from a digital capture system

4.15
RGB
color model with red, green, blue components

4.16
scan line
row of pixels, with all pixels having the same y coordinate

4.17
stereoscopic image
image containing two separate views of the depicted scene, intended to be presented to the viewer on a 3D display device such that the left-eye view and right-eye view are visible only to the viewer's left and right eye respectively

4.18
window
axis-parallel rectangular region in pixel space, specified by the coordinates of two opposing corners, (x_{min}, y_{min}) and (x_{max}, y_{max}), where $x_{min} < x_{max}$ and $y_{min} < y_{max}$, and including all pixels with coordinates (x, y) where $x_{min} \le x \le x_{max}$, and $y_{min} \le y \le y_{max}$

4.19
writer
device producing files compliant with this standard

5 General

5.1 Image

The file shall contain exactly one image.

The image shall be an RGB image, monoscopic or stereoscopic, with or without alpha channels. Section 5.2 specifies, for each type of image, the channels that are present in the file.

The red, green, and blue components of pixels shall be as defined in SMPTE ST-2065-1 (Academy Color Encoding Specification).

The alpha value of a pixel shall be as defined in Section 5.3.

The image shall contain one value for each channel of each pixel in the data window.

The red, green, blue, and alpha values are of type `half` (16-bit floating-point).

Section 7.7 describes how pixel data and channels are organized into scan lines.

5.2 Image Channels

The type of image determines the channels present in the file:

- A monoscopic image without alpha shall contain channels “B”, “G”, and “R”.
- A monoscopic image with alpha shall contain channels “A”, “B”, “G”, and “R”.
- A stereoscopic image without alpha shall contain channels “B”, “G”, “R”, “left.B”, “left.G”, and “left.R”.
- A stereoscopic image with alpha shall contain channels “A”, “B”, “G”, “R”, “left.A”, “left.B”, “left.G” and “left.R”.

Note: OpenEXR allows an arbitrary number of image channels. The ACES image container restricts the set of image channels that can appear in a file to the four possibilities listed above.

Channels “B”, “G”, and “R” shall represent the blue, green, and red components of the pixel of the monoscopic image or in the right-eye view.

EXAMPLE 1: The collection of all R values of all pixels in a scan line is the “R channel” of that scan line.

Channels “left.B”, “left.G” and “left.R” shall represent the blue, green, and red components of the pixel in the left-eye view.

Channel “A” shall represent alpha of the pixel of the monoscopic image or in the right-eye view.

Channel “left.A” shall represent alpha of the pixel in the left-eye view.

The channel order in scan lines and in the channels attribute shall be “A”, “B”, “G”, “R”, “left.A”, “left.B”, “left.G”, “left.R”.

5.3 Alpha Channel

When an Alpha channel is not present, the pixels shall be considered fully opaque.

When an Alpha channel is present, the alpha value in the pixel's alpha channel shall specify the proportion by which the pixel shall obscure the background image's pixel if used in an "over" compositing operation. This is known as the opacity of the pixel.

The alpha values shall be in the range [0.0, 1.0]. A value of 0.0 shall indicate that the pixel is fully transparent, and 1.0 shall indicate that the pixel is fully opaque.

The pixel's RGB values shall specify the amount of color that the pixel shall contribute if used in an "over" compositing operation.

The RGB values may exceed the alpha value, so the pixel can contribute color to the composition also when the alpha value is zero.

Note 1: While the above definition looks like "premultiplied alpha", this term is not used here, as some definitions of "premultiplied alpha" or "associated alpha" require that all RGB values are premultiplied with alpha or disallow RGB > alpha. For compatibility with OpenEXR, this file format does not impose such requirements on RGB. Other documents can contain workflow-specific requirements on preparing RGB values prior to storing them in the file. The above definition is not related to "straight alpha" or "unassociated alpha".

Note 2: RGB > alpha is useful for semi-transparent sources of light, such as fire with semi-transparent smoke, head-up displays, glow, lens flare, or window reflections.

Note 3: For color corrections and other operations that work on "straight alpha" instead of "premultiplied alpha", any premultiplied alpha baked into the file's RGB values can be removed by dividing the RGB values with alpha. As the RGB values are in floating point, this can be done without loss of precision over a wide range of values, except for alpha = 0. This removal is not appropriate when alpha is not baked into the file's RGB values, such as for fire, in which case other adjustments (if any) can be more appropriate.

5.4 Display Window, Data Window

Two windows specify the boundaries of the image:

The *display window* shall define the window that is to be displayed.

The *data window* shall define the window for which pixel data are available in the file.

No specific spatial relationship between the data window and the display window is required. The two windows may be coincident, they may overlap, or they may be completely disjoint.

EXAMPLES: Assume that a motion picture is produced with a resolution of 2048 by 1080 pixels. The upper left and lower right corners of the display window for all frames of the movie are at (0, 0) and (2047, 1079). For most images, in particular finished frames that will be part of the final product, the data window is the same as the display window, but for some images that are used in producing the finished frames, the data window differs from the display window.

For a background plate that will be heavily post-processed, extra pixels beyond the edge of the display window are recorded. The upper left and lower right corners of the data window are at (-100, -100) and (2147, 1179). The extra pixels are not normally displayed. Their existence allows operations such as large-kernel blurs or simulated camera shake to avoid edge artifacts.

Alternatively, while working on a computer-generated element, an artist can repeatedly render the same frame. To save time, the artist can render only a small region of interest, and the resulting image files contain no pixel data outside this region of interest. When the image is displayed, the area that is inside the display window but outside of the data window can be filled with an arbitrary color.

5.5 Scan Lines

The pixels in the data window shall be stored as *scan lines.* A data window with opposing corners (x_{min}, y_{min}) and (x_{max}, y_{max}) shall contain

$$y_{max} - y_{min} + 1$$

scan lines. Each scan line shall contain

$$x_{max} - x_{min} + 1$$

pixels.

5.6 Screen Window

The *screen window* is the boundary (left-right and top-bottom) of the camera's view of the scene, and it establishes a relationship between the camera's view of the scene and the file's display window. The screen window is a rectangle in a screen window coordinate system.

The screen window coordinate system shall be a left-handed, Cartesian coordinate system.

Its x-axis shall point in the same direction (to the right) as the x-axis in the display window (which is in pixel space).

Its y-axis shall point in the opposite direction (up) of the y-axis in the display window.

Its z-axis shall point in the viewing direction, into the scene.

The camera shall be at location {0, 0, 0} and shall look along the positive z axis.

The screen window shall be an axis-aligned rectangle on the z = +1 plane.

The screen window's rectangle shall map to the display window's rectangle (the latter in pixel space).

The camera's view of the scene as shown in the display window is then the scene cropped by the screen window boundary at the z = +1 plane.

The screen window's rectangle is specified by its width and center position.

The screen window's width W shall be the width of the rectangle in the screen window coordinate system.

The screen window's center C shall be the x and y coordinates of the rectangle's center in the screen window coordinate system.

Note 1: The screen window height H can be derived from the screen window width W, display window width dW and height dH and the pixel aspect ratio pR as follows:

$$H = W\ dH / dW / pR$$

The coordinates of the corners of the screen window / display window can be computed from C, W and H:

$$\{ x, y \} = \{ C.x \pm W / 2, C.y \pm H / 2 \}$$

Note 2: The screen window coordinate system is different from the coordinate systems used in Section 9.13 (cameraViewingDirection) and Section 9.10 (cameraPosition). The screen window coordinate system is locked with the pixel space and is dimensionless, so the z = +1 plane can be thought of as being placed at any distance from the camera including behind the scene.

EXAMPLE: If C = { 0, 0 } and W = 1, then the horizontal viewing angle is 53 degrees, and the camera is aligned with the center of the image.

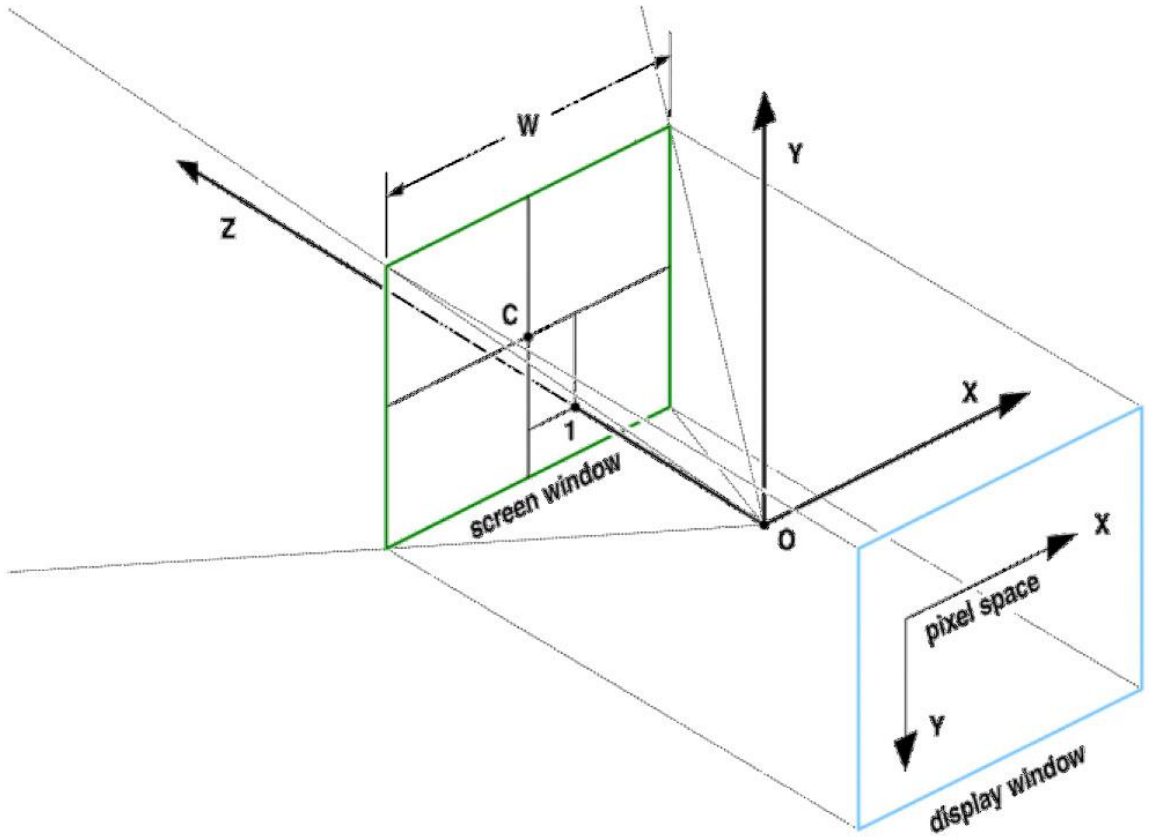

Figure 1 – Window Coordinate Systems (showing the screen window at z = +1)

6 Basic Data Types

6.1 Grouping

An ACES image container file shall be a sequence of 8-bit bytes.

Groups of bytes represent basic objects such as integral numbers, floating-point numbers, and strings. Those objects are grouped together to form compound objects such as attributes, or scan lines.

6.2 Packing

All multi-byte basic data types shall be stored in little-endian byte order.

Data in the file shall be densely packed. The file shall contain no "padding" bytes whose only purpose is to ensure alignment of other objects at file locations divisible by some power of two.

Note: Consider the following C struct:

```
struct SI
{
    short s;
    int i;
};
```

On most computers, the in-memory representation of the `SI` object occupies eight bytes: 2 bytes for `s`, 2 padding bytes to ensure four-byte alignment of `i`, and 4 bytes for `i`. In an ACES image container file an equivalent object consumes only six bytes: 2 bytes for `s` and 4 bytes for `i`. The two padding bytes are not included in the file.

6.3 Integer Numbers

Integer numbers can be signed or unsigned. Signed numbers shall be represented using two's complement. Integer numbers shall be as defined in Table 1.

Table 1 – Integer Data Types

Name	*Signed*	*Size in bytes*
`unsigned char`	no	1
`short`	yes	2
`unsigned short`	no	2
`int`	yes	4
`unsigned int`	no	4
`unsigned long`	no	8

6.4 Floating-Point Numbers

The representation of floating-point numbers shall conform to the IEEE 754 standard. Floating-point numbers shall be as defined in Table 2.

Table 2 – Floating Point Data Types

Name	*Size in bytes*	*IEEE 754-2008 format*
`half`	2	binary16
`float`	4	binary32
`double`	8	binary64

6.5 Character Encoding, Sequences, and Strings

6.5.1 Character encoding and sequences

Characters that represent text shall be encoded using Unicode UTF-8 encoding in NFC normal form, as defined in the Unicode standard.

Character sequences shall be stored as strings as specified in Section 6.5.2.

Note: Unicode extends the ASCII character set, and a UTF-8 encoded character can use several bytes, thus the number of characters in a string can be lower than the number of bytes in the same string.

6.5.2 Strings

Unless prescribed otherwise for a specific attribute or use case, strings shall contain only characters encoded as specified in Section 6.5.1.

A `null-terminated string` shall consist of a sequence of bytes, followed by a byte with the value null (0x00). The byte sequence shall not contain bytes with the value null.

A `length-counted string` shall consist of an `int`, followed by a sequence of bytes. The value of the `int` shall be the number of bytes in the sequence. The byte sequence may contain bytes with the value null.

Note: Most strings are stored using the `string` attribute type. See Section 8.14, and Section 9. Null-terminated `strings` are used in attribute names, attribute types, and `channels` names. `Length-counted strings` are used in the `stringVector` attribute type.

7 File Layout

7.1 File Extension

The file name's extension shall be "exr".

Note: This enables legacy OpenEXR readers to read the file.

7.2 High-Level File Layout

The file shall contain the components listed inTable 3, concatenated in the order shown in the table.

Table 3 – File Structure

magic number
version field
header
line offset table
scan line storage
end-of-file filler

7.3 Magic Number

The magic number shall be first in the file and shall be an `int` with the value 20 000 630 (decimal).

Note 1: The magic number allows file readers to distinguish ACES and OpenEXR image container files from other files, since the first four bytes of an ACES or OpenEXR image container file are always 0x76, 0x2f, 0x31, and 0x01.

Note 2: In order to maintain compatibility, ACES image container files use the same magic number as OpenEXR files.

7.4 Version Field

The version field shall be an `int` with the value 1026 (decimal) or the value 2 (decimal).

A value of 1026 shall indicate that the file can contain attribute names or attribute type names with a length exceeding 31 bytes.

A value of 2 shall indicate that all attribute names and attribute type names have lengths shorter than 32 bytes.

Note: The above numbers were chosen for compatibility with OpenEXR, where the version field is treated as two separate bit fields. The 8 least significant bits (bits 0 through 7) contain a file format version number. The 24 most significant bits (8 through 31) are treated as a set of boolean flags.

The current OpenEXR version number is 2, where “current” refers to the version that is compatible with the ACES image container file format.

In OpenEXR bit number 9 of the version field (bit mask 0x200) indicates if the file contains scan lines or tiles. Since the ACES image container format does not support tiles, bit number 9 is always zero.

In OpenEXR, when bit number 10 of the version field (bit mask 0x400) is zero, the length of attribute names, attribute type names, and channel names are limited to 31 bytes. Attribute names and attribute type names in the ACES image container can be longer than 31 bytes, in which case bit 10 is 1. Some older OpenEXR (v1.6) readers do not support value 1.

The remaining 22 flags in the version field are reserved and are zero.

7.5 Header

7.5.1 Header Attributes

The header shall be a sequence of attributes, followed by a null byte (0x00).

The size of the header shall not exceed 1 048 576 bytes.

The attributes may occur in any order. No sorting by name or type is required.

7.5.2 Attribute Structure

Each attribute shall contain the fields listed in Table 4.

Table 4 – Attribute Structure

attribute name
attribute type name
attribute size
attribute value

The attribute name and the attribute type name shall be `null-terminated strings`. Excluding the terminating null byte, the lengths of the attribute name and attribute type name shall be at least 1 byte and at most 255 bytes.

The attribute name is unique within the file. The file shall not include two or more attributes with equal attribute name strings. Several attributes and their names are defined in Section 9 (Predefined Attributes) of this document.

The attribute type name shall specify the type of data stored in the attribute value. Several attribute types and their type names are defined in Section 8 (Predefined Attribute Types) of this document.

The attribute size shall be an `int` specifying the size, in bytes, of the attribute value. The size shall be non-negative.

7.5.3 Required Attributes

The header shall contain the attributes listed in Table 5.

The `multiView` attribute shall be present, when, and only when, the file contains a stereoscopic image. The header may contain other attributes.

Table 5 – Required Attributes

`acesImageContainerFlag`
`channels`
`chromaticities`
`compression`
`dataWindow`
`displayWindow`
`lineOrder`
`pixelAspectRatio`
`screenWindowCenter`
`screenWindowWidth`

7.5.4 Attribute Requirements

Other standards documents can specify additional ACES container attributes and types, or impose additional constraints on attributes specified herein. For example, a playback specification can mandate that all images in an image sequence have the same values for dataWindow, pixelAspectRatio, and framesPerSecond; an archiving specification can mandate that the uuid attribute be present in every file.

A reader shall be capable of reading a file with an arbitrary set of attributes, including attribute names and types not specified in this standard.

Whenever possible, a writer shall use the attributes defined in section 9, the attribute types defined in section 8, or ACES container attributes or types specified in other standards documents, for metadata. When these specifications define no appropriate attribute name or type, a writer may define its own attribute names or types. The names of vendor-specific attributes and types shall comprise a vendor identifier, a period, and a descriptive name, for example, "wetDog.dryingTime".

An editor shall preserve existing attributes and attribute values, except those attributes and attribute values that are obsolete or invalid after the editing operation. The editor shall ensure that all attribute values remain valid, and shall remove or recalculate attributes as needed.

Note 1: Attributes whose values can get invalidated indirectly by an editing operation include, but are not limited to, headerDigestMD5, imageDigestMD5, originalImageFlag, and uuid.

EXAMPLE: To store the exposure time for an image, the pre-defined `expTime` attribute is used, but storing an unusual item for which no pre-defined attribute exists, such as a company-specific billing number, is only possible by defining a new, application-specific attribute.

Note 2: The layout of an attribute allows file readers to either skip unknown attributes or to preserve unknown attributes as opaque byte strings.

7.6 Line Offset Table

The line offset table shall be a sequence of *line offsets* with one line offset for each scan line of the data window.

Each line offset shall be an `unsigned long` containing the offset, in bytes, from the start of the file to the start of the corresponding scan line storage. The table entries shall be in ascending order of the scan lines' Y coordinate.

Note 1: If the opposing corners of the data window are at (x_{min}, y_{min}) and (x_{max}, y_{max}), then the line offset table contains y_{max} - y_{min} + 1 line offsets; the first line offset corresponds to the scan line with y coordinate y_{min} and the last line offset corresponds to scan line with y coordinate y_{max}.

Note 2: The line offset table in the file is redundant; the position of the scan lines in the file can be computed from the data window and the line order attribute. The scan line table is included for compatibility with OpenEXR, where the table is necessary to support random access to scan lines when data compression is used.

7.7 Scan Line Storage

7.7.1 Scan Line Objects

The Scan Line Storage shall be a sequence of scan line objects with one scan line object for each scan line of the data window.

The layout of a scan line object shall be as described in Table 6.

Table 6 – Scan Line Object

y coordinate
pixel data size
pixel data for first channel
….
pixel data for last channel

The y coordinate shall be an `int` specifying the Y coordinate of the scan line in the data window.

The pixel data size shall be an `int` specifying the number of bytes in the pixel data.

The pixel data shall be a sequence of values of type `half` (16-bit floating-point).

The sequence of values shall be grouped in channels as specified in Section 5.2, and in the channel order specified in the `channels` attribute (Section 9.16).

The values of the first channel shall be stored contiguously in ascending order of the pixels' X coordinate, followed by the values for the next channel in ascending order of the pixels' X coordinate, and so on for all channels.

EXAMPLE: 17, 24, B0, B1, B2, B3, G0, G1, G2, G3, R0, R1, R2, R3 represents scan line 17 of a monoscopic RGB image with a data window width of 4.

7.7.2 Scan Line Order

The scan lines are stored according to their Y coordinates. The scan lines may be stored:

– in *increasing Y* order where scan lines with lower Y coordinates (than other scan lines) shall be stored closer to the beginning of the file, or

– in *decreasing Y* order where scan lines with higher Y coordinates shall be stored closer to the beginning of the file.

The scan lines should be stored in increasing order. The scan line order is specified in the lineOrder attribute.

Note: Pixel Y coordinates always increase from top to bottom, regardless of scan line storage order.

7.8 End-of-File Filler

The end-of-file filler shall be 0 to 1 048 576 bytes long. It shall contain no meaningful information. The value should be 0. Its sole purpose is to provide readable bytes beyond the last scan line.

Note: Reading the file can be faster when the end-of-file filler pads the file to a sector boundary. The length of the filler can be determined from the file size.

8 Predefined Attribute Types

8.1 General

This section specifies the data layout for the attribute values of predefined attribute types. The file may include additional attribute types not defined here.

Note: These type specifications define only the syntax, not the semantics, of the attribute values. The range and semantics of an attribute value is specified in the attribute definition. See Section 9 (Predefined Attributes) for these definitions of predefined attributes.

8.2 box2i

Attribute values of type `box2i` shall have the layout specified in Table 7.

Table 7 – box2i Structure

Field	*Data type*
`xMin`	`int`
`yMin`	`int`
`xMax`	`int`
`yMax`	`int`

8.3 chlist

Attribute values of type `chlist` shall contain a sequence of `channel` objects, followed by a null byte (0x00). The layout of a `channel` object shall be as specified in Table 8.

Table 8 – chlist Structure

Field	*Data type*
`name`	`null-terminated string` with a length of at least 1 and at most 255 bytes
`pixelType`	`int`
`pLinear`	`unsigned int`
`xSampling`	`int`
`ySampling`	`int`

8.4 chromaticities

Attribute values of type `chromaticities` shall have the layout specified in Table 9.

Table 9 – chromaticities Structure

Field	*Data type*
`red.x`	`float`
`red.y`	`float`
`green.x`	`float`
`green.y`	`float`
`blue.x`	`float`
`blue.y`	`float`
`white.x`	`float`
`white.y`	`float`

8.5 compression

Attribute values of type `compression` shall contain a single `unsigned char`.

8.6 double

Attribute values of type `double` shall contain a single `double` object.

8.7 float

Attribute values of type `float` shall contain a single `float` object.

8.8 half

Attribute values of type `half` shall contain a single `half` object.

8.9 int

Attribute values of type `int` shall contain a single `int` object.

8.10 lineOrder

Attribute values of type `lineOrder` shall contain a single `unsigned char`.

8.11 keycode

Attribute values of type `keycode` shall have the layout and values specified in Table 10, and shall specify the key code information equivalent to key codes as defined in SMPTE ST 254 and SMPTE ST 268.

Note: SMPTE ST 268 specifies key codes as ASCII values, while this attribute in this specification uses binary-encoded values.

Table 10 – keyCode Structure and Values

Field	*Data type*	*Interpretation*	*Value range*
`filmMfcCode`	`int`	film manufacturer code	0 – 99
`filmType`	`int`	film type code	0 – 99
`prefix`	`int`	prefix to identify film roll	0 – 999999
`count`	`int`	count, increments once every `perfsPerCount` perforations	0 – 9999
`perfOffset`	`int`	offset of frame, in perforations, from the zero-frame reference mark	1 - 119
`perfsPerFrame`	`int`	number of perforations per frame	1 – 15 Note: typical values include 1 for 16mm film, 3, 4 or 8 for 35 mm film, 5, 8 or 15 for 65mm film.
`perfsPerCount`	`int`	number of perforations per count	20 – 120 Note: typical values include 20 for 16mm film, 64 for 35mm film, 80 or 120 for 65mm film.

8.12 rational

Attribute values of type `rational` shall have the layout specified in Table 11.

Table 11 – rational Structure

Field	*Data type*
`n`	`int`
`d`	`unsigned int`

8.13 short

Attribute values of type `short` shall contain a single `short` object.

8.14 string

The attribute value of type `string` shall contain a sequence of bytes.

Unless prescribed otherwise in a specific attribute or use case, the byte sequence shall contain only characters encoded as specified in Section 6.5.1.

Note: As specified in Section 7.5.2 (Attribute Structure), the length of the string is given in the attribute size. Storing an explicit length count as part of the attribute value would be redundant.

8.15 stringVector

Attribute values of type `stringVector` shall contain a sequence of zero or more `length-counted strings`.

Note 1: The length counts of the individual strings are stored as part of the attribute value since they are not redundant as in attributes of type string.

Note: 2 The number of strings in the attribute can be inferred from the attribute size and the length counts of the individual strings. Explicitly storing the number of strings would be redundant.

8.16 timecode

Attribute values of type `timecode` shall have the layout specified in Table 12. The `timeAndFlags` and `userData` fields shall represent time code information as defined in SMPTE ST 12–1, and shall be stored as an 8-byte structure as specified for type value 81h (SMPTE 12 time code metadata) in SMPTE ST 331.

Table 12 – timecode Structure

Field	*Data type*
`timeAndFlags`	`unsigned int`
`userData`	`unsigned int`

The `timeAndFlags` field, as defined in SMPTE ST 12-1 and SMPTE ST 331, contains a time code packed for 24-frame film, 60-field television or 50-field television, as shown in Table 13.

Table 13 – timeAndFlags Structure

Bits	*Contents for 24-frame film*	*Contents for 60-field television*	*Contents for 50-field television*
0 – 3	frame units	frame units	frame units
4 – 5	frame tens	frame tens	frame tens
6	unused, 0	drop frame flag	unused, 0
7	unused, 0	color frame flag	color frame flag
8 – 11	seconds units	seconds units	seconds units
12 – 14	seconds tens	seconds tens	seconds tens
15	phase flag	field/phase flag	bgf0
16 – 19	minutes units	minutes units	minutes units
20 – 22	minutes tens	minutes tens	minutes tens
23	bgf0	bgf0	bgf2
24 – 27	hours units	hours units	hours units
28 – 29	hours tens	hours tens	hours tens
30	bgf1	bgf1	bgf1
31	bgf2	bgf2	field/phase flag

The `userData` field contains user-defined data and control codes, packed as shown in Table 14.

Table 14 – userData Structure

Bits	*Contents*
0 – 3	Binary group 1
4 – 7	Binary group 2
8 – 11	Binary group 3
12 – 15	Binary group 4
16 – 19	Binary group 5
20 – 23	Binary group 6
24 – 27	Binary group 7
28 – 31	Binary group 8

In the tables above, the bits are numbered from 0 to 31, with 0 referring to the least significant bit and 31 to the most significant bit.

8.17 unsignedChar

Attribute values of type `unsignedChar` shall contain a single `unsigned char` object.

8.18 unsignedInt

Attribute values of type `unsignedInt` shall contain a single `unsigned int` object.

8.19 unsignedLong

Attribute values of type `unsignedLong` shall contain a single `unsigned long` object.

8.20 unsignedShort

Attribute values of type `unsignedShort` shall contain a single `unsigned short` object.

8.21 v2f

Attribute values of type `v2f` shall have the layout specified in Table 15.

Table 15 – v2f Structure

Field	*Data type*
x	float
y	float

8.22 v3f

Attribute values of type `v3f` shall have the layout specified in Table 16.

Table 16 – v3f Structure

Field	*Data type*
x	float
y	float
z	float

9 Predefined Attributes

9.1 General

This section specifies predefined attributes and the type and semantics of their attribute values. Files may include additional attributes not defined here.

9.2 acesImageContainerFlag

The `acesImageContainerFlag` attribute shall be of type `int`, and shall contain the value 1. Values other than 1 are reserved. This attribute and value shall indicate that the file and all attribute values are compliant with this specification. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

Note 1: A writer cannot indicate compliance by merely setting the flag, without providing compliant attribute values.

Note 2: This flag is absent in OpenEXR files, which by coincidence can have attribute values compliant with the specification.

9.3 altitude

The `altitude` attribute shall be of type `float`, and shall specify the altitude of the location where the image was created or captured. Altitude shall be specified in meters, with positive values indicating locations above sea level.

9.4 aperture

The `aperture` attribute shall be of type `float`, and shall specify the T-stop setting of the lens. The T-stop is the calibrated F-number corrected by a manufacturer for actual light transmission of all of the lens elements, rather than the nominal aperture opening size. The nominal F-number may be used if the T-stop is not known.

Note: In general, the F-number is the focal length of the lens divided by the diameter of the entrance pupil.

9.5 cameraFirmwareVersion

The `cameraFirmwareVersion` attribute shall be of type `string`, and shall specify the firmware version of the camera. The string may contain any printable characters. The string shall contain at least one byte.

9.6 cameraIdentifier

The `cameraIdentifier` attribute shall be of type `string`, and shall identify this camera uniquely among all cameras from all vendors. The string shall contain only printable characters.

Note: This can be the camera's MAC address, or a concatenation of fully defined cameraMake, cameraModel, cameraSerialNumber, etc.

9.7 cameraLabel

The `cameraLabel` attribute shall be of type `string`, and shall contain a text label for how the camera is used or assigned in this clip, for example "Camera 1 Left", "B Camera", or "POV". The string shall contain only printable characters. The string shall contain at least one byte.

9.8 cameraMake

The `cameraMake` attribute shall be of type `string`, and shall specify the manufacturer or vendor of the camera. The string shall contain only printable characters. The string shall contain at least one byte.

9.9 cameraModel

The `cameraModel` attribute shall be of type `string`, and shall specify the model name or model number of the camera. The string shall contain only printable characters. The string shall contain at least one byte.

9.10 cameraPosition

The `cameraPosition` attribute shall be of type `v3f`, and shall specify the x, y, and z position of the camera, in meters. This attribute can be used to record the position of a stationary camera or to track a moving camera. The attribute may record the center of the camera's sensor, or the center of the entrance pupil of the lens.

The coordinate system for the camera's position shall be fixed in relation to the set and shall be a right-handed, Cartesian coordinate system with the z-axis pointing upwards, and the y-axis pointing 90 degrees to the left of the x-axis direction.

Note 1: Recording the position of the center of the entrance pupil provides the most usable value for example when panning. However, this position changes with lens and focus. For cases when this precision is not needed, recording the center of the sensor is provided as an alternative.

Note 2: For highest precision, the chosen coordinate system is usually NOT the GPS system, but an on-set coordinate system. Any coordinate system can be chosen for recording the camera's position, as long as its axes are orthogonal, and z points up. The coordinate system can be moving with the set, for example, when on a ship. Its reference point (usually assigned position 0,0,0) can be within or outside the set.

9.11 cameraSerialNumber

The `cameraSerialNumber` attribute shall be of type `string`, and shall specify the serial number of the camera. The string may contain any printable characters, not only digits. The string shall contain at least one byte.

9.12 cameraUpDirection

The `cameraUpDirection` attribute shall be of type `v3f`, and shall specify the up direction of the camera.

When the cameraUpDirection attribute is present, the cameraViewingDirection attribute shall also be present.

The v3f structure shall specify the x, y, z values for the camera's "up" direction. The up direction shall be perpendicular to the direction specified in the cameraViewingDirection attribute. The coordinate system for the camera's up direction shall be the coordinate system defined for the cameraViewingDirection attribute

9.13 cameraViewingDirection

The `cameraViewingDirection` attribute shall be of type `v3f`, and shall specify the viewing direction of the camera.

The v3f structure shall specify the x, y, z values for the viewing direction of the camera (the direction of the optical axis of the lens).

The coordinate system for the camera's direction shall be a right-handed, Cartesian coordinate system with the z-axis pointing upwards, and the y-axis pointing 90 degrees to the left of the x-axis direction. No units are specified, nor needed, for this coordinate system.

When the cameraPosition attribute is present, the axes of the cameraViewingDirection coordinate system shall be parallel to, and point in the same direction as, those of the cameraPosition coordinate system.
When the cameraPosition attribute is not present, the directions of the axes in the coordinate system shall be fixed in relation to the set.

When the cameraViewingDirection attribute is present, the cameraUpDirection attribute shall also be present. When the cameraViewingDirection attribute is not present, the cameras's viewing and up directions are undefined.

Note: The methods for converting measured intrinsic rotational angles (heading, elevation, bank or yaw, pitch, roll or roll, pitch, yaw) to the cameraViewingDirection are out of scope for this specification as they are specific to the order in which the rotations are applied to the camera mount.

9.14 capDate

The `capDate` attribute shall be of type `string`, and shall specify the time when the image was created or captured. The time shall be given in local time. The string shall be 19 bytes long and shall be formatted as

YYYY:MM:DD hh:mm:ss

where YYYY is the year (4 digits, e.g. 2011), MM is the month (2 digits, 01, 02, … 12), DD is the day of the month (2 digits, 01, 02, … 31), hh is the hour (2 digits, 00, 01, … 23), mm is the minute, and ss is the second (2 digits each, 00, 01, … 59). DD and hh shall be separated by one space character.

9.15 captureRate

The `captureRate` attribute shall be of type `rational`, and shall specify the capture rate of the image sequence to which the image belongs. This can be used for variable frame rates and time-lapse photography, and for any clock rate. The capture rate shall be specified in frames per second. The capture rate shall be specified as a rational number, n/d, where both n and d shall be greater than zero.

The value of the capture rate shall be calculated as follows:

$$r = \frac{1}{t_n - t_{n-1}},$$

where:

- r shall be the capture rate, in frames per second,
- t_n shall be the time of this frame,
- t_{n-1} shall be the time of the previous frame, and
- the time of a frame shall be the time of the center of the frame's exposure interval, in seconds.

If there is no previous frame, the rate may be calculated as if such a frame existed at a desired time.

Note: `captureRate` is stored as rational number in order to allow exact representation of NTSC frame and field rates.

9.16 channels

The `channels` attribute shall be of type `chlist`, and shall contain three to eight `channel` objects, depending on the type of image, as defined in section 5.1. Section 7.5.3 Required Attributes specifies that this attribute is required.

For a monoscopic RGB image without alpha, the `name` fields in the `channel` objects shall be “B”, “G”, and “R”, respectively.

For a monoscopic RGB image with alpha, the `name` fields in the `channel` objects shall be “A”, “B”, “G,” and “R”, respectively.

For a stereoscopic RGB image without alpha, the `name` fields in the `channel` objects shall be “B”, “G”, “R”, “left.B”, “left.G”, and “left.R”, respectively.

For a stereoscopic RGB image with alpha, the `name` fields in the `channel` objects shall be “A”, “B”, “G”, “R”, “left.A”, “left.B”, “left.G”, and “left.R”, respectively.

In all `chlist` objects, the `pixelType`, `xSampling` and `ySampling` values shall be 1, and the `pLinear` value shall be 0.

Note: The fields `pixelType, pLinear, xSampling, ySampling` have fixed values in ACES-compliant channel objects. The `pixelType` value 1 indicates that pixel values are stored in half-float format. The `xSampling` and `ySampling` values 1 indicate no subsampling. The `pLinear` value 0 indicates linearly encoded pixel values.

9.17 chromaticities

The `chromaticities` attribute shall be of type `chromaticities`, and shall contain the chromaticity values of the ACES RGB primaries and the ACES RGB white point as defined in SMPTE ST 2065-1 (Academy Color Encoding Specification (ACES)). Section 7.5.3 (Required Attributes) specifies that this attribute is required.

These values are shown in Table 17.

Table 17 – chromaticities Values

`red.x` = 0.73470	`red.y` = 0.26530
`green.x` = 0.00000	`green.y` = 1.00000
`blue.x` = 0.00010	`blue.y` = -0.07700
`white.x` = 0.32168	`white.y` = 0.33767

9.18 comments

The `comments` attribute shall be of type `string`, and may contain image information in human readable form, for example, a verbal description of the image, or notes related to the production or intended use of the image. The attribute shall not be used in place of any other predefined attribute. The format of the string is undefined. The string shall contain only printable characters. The string shall contain at least one byte.

9.19 compression

The `compression` attribute shall be of type `compression`, and shall contain the value 0, indicating no compression. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

9.20 convergenceDistance

The `convergenceDistance` attribute shall be of type `float`, and shall specify the distance in meters from the baseline of the two lens nodal points to the point where the lens axes would cross each other. A positive value shall indicate a point in front of the camera. For orthogonal stereo images, this value should be a positive-infinity floating-point value.

9.21 creator

The `creator` attribute shall be of type `string`, and shall specify the creator of the image. The string shall contain only printable characters. The string shall contain at least one byte.

9.22 dataWindow

The `dataWindow` attribute shall be of type `box2i`, and shall specify the data window, with `xMin` and `yMin` defining the upper left corner and `xMax` and `yMax` defining the lower right corner. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

9.23 displayWindow

The `displayWindow` attribute shall be of type `box2i`, and shall specify the display window, with `xMin` and `yMin` defining the upper left corner and `xMax` and `yMax` defining the lower right corner. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

9.24 expTime

The `expTime` attribute shall be of type `float`, and shall specify the exposure time for the image, in seconds.

9.25 focalLength

The `focalLength` attribute shall be of type `float`, and shall specify the actual focal length of the lens, in millimeters, when the image was captured.

9.26 focus

The `focus` attribute shall be of type `float`, and shall specify the focus distance, in meters, when the image was captured. Focus distance shall be from the camera's film or sensor plane. If the lens was focused at its hyper-focal setting, the value may be the hyper-focal distance or a positive-infinity floating-point value.

9.27 framesPerSecond

The `framesPerSecond` attribute shall be of type `rational`, and shall specify the nominal playback rate of the image sequence to which the image belongs. The playback rate shall be specified in frames per second. The playback rate shall be a rational number; n/d. n and d shall both be greater than zero.

Note 1: `framesPerSecond` is stored as rational number in order to allow exact representation of NTSC frame and field rates, as well as "film-like" frame rates that synchronize with NTSC frame rates. Some commonly used frame rates include 24000/1001, 24/1, 25/1, 30000/1001, 30/1, 48000/1001, 48/1, 50/1, 60000/1001 and 60/1.

Note 2: The attribute name was chosen for compatibility with OpenEXR.

9.28 free

The `free` attribute shall be of type `string`. It can be used for storage alignment, and reserving space for additional attributes. The string shall contain no meaningful information. Each byte in the string should be equal to 0x00.

Note: Reading the file can be faster when the `free` attribute is used to align the scan line storage in a set of files to start at the same offset or on a sector boundary.

9.29 imageDigestMD5

The `imageDigestMD5` attribute shall be of type `string`, and shall be an MD5 message digest of the scan line storage area, calculated as specified in IETF RFC 1321. The string is 128 bits long.

Note: An edit of pixel data will result in a different digest value.

9.30 imageCounter

The `imageCounter` attribute shall be of type `int`, and shall specify an image number. For a sequence of images, the image number shall be increasing when the images are accessed in the intended play order.

Note: imageCounter can be used to uniquely identify frames of high-speed photography that would have identical time codes.

9.31 imageRotation

The `imageRotation` attribute shall be of type `float`, and shall specify a rotation of the image in degrees. The value shall be in the range [-45, +45].The center of rotation shall be the center of the display window. A positive value shall specify a rotation from x-axis towards y-axis. When the attribute is not present, then the image shall not be rotated.

9.32 interocularDistance

The `interocularDistance` attribute shall be of type `float`, and shall specify the distance in meters along the baseline between the nodal points of the two lenses. The center of the lens axis as measured at the front surface of each lens may be used. A negative value can reflect a 'flip' of the eye positions for each camera.

9.33 isoSpeed

The `isoSpeed` attribute shall be of type `float`, and shall specify the exposure index (EI) rating of the camera when the image was captured.

Note: The ISO standard ISO 12232:2006 provides techniques for determining the exposure index rating.

EXAMPLE: isoSpeed 600

9.34 keyCode

The `keyCode` attribute shall be of type `keycode`, and shall specify the key code of the image.

9.35 latitude

The `latitude` attribute shall be of type `float`, and shall specify the geographic latitude of the location where the image was created or captured. Latitude shall be specified in decimal degrees, with positive values indicating locations north of the equator and negative values indicating locations south of the equator. The value shall be in the range [-90.0, +90.0].

9.36 lensAttributes

The `lensAttributes` attribute shall be of type `string`. The string shall contain at least one byte. The string may contain lens metadata not specified in other predefined attributes (such as aperture, focus, focalLength, lensMake, lensModel, lensSerialNumber). The attribute shall not be used in place of any other predefined attribute. The format of the string is undefined.

The file may include additional lens attributes. The attribute name shall be a vendor-specific prefix followed by "`.lensAttributes`". The attribute shall meet the requirements of the `lensAttributes` attribute.

9.37 lensMake

The `lensMake` attribute shall be of type `string`, and shall specify the manufacturer or vendor of the lens. The string shall contain only printable characters. The string shall contain at least one byte.

9.38 lensModel

The `lensModel` attribute shall be of type `string`, and shall specify the model name or model number of the lens. The string shall contain only printable characters. The string shall contain at least one byte.

9.39 lensSerialNumber

The `lensSerialNumber` attribute shall be of type `string`, and shall specify the serial number of the lens. The string may contain any printable characters, not only digits. The string shall contain at least one byte.

9.40 lineOrder

The `lineOrder` attribute shall be of type `lineOrder`, and shall specify the order in which the scan lines are located in the file. The attribute value shall be either 0 indicating increasing Y line order or 1 indicating decreasing Y line order. The value should be 0. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

Note: Pixel Y coordinates always increase from top to bottom. The direction of the Y-axis is not affected by the lineOrder attribute. This attribute indicates only the storage order of rows in the scan line area.

9.41 longitude

The `longitude` attribute shall be of type `float`, and shall specify the geographic longitude of the location where the image was created or captured. Longitude shall be specified in decimal degrees, with positive values indicating locations east of Greenwich and negative values indicating locations west of Greenwich. The value shall be in the range [-180.0, +180.0].

9.42 metadataDigestMD5

The `metadataDigestMD5` attribute shall be of type `string`, and shall be an MD5 message digest, calculated as specified in IETF RFC 1321. The string is 128 bits long. The input to the digest calculation shall start with the first byte of the file, and end with the last byte in the line offset table, in file byte order. The input to the digest calculation shall have a `metadataDigestMD5` attribute value with all bits equal to zero.

Note: A reorder or edit of the attributes will result in a different digest value.

9.43 multiView

The `multiView` attribute shall be of type `stringVector`, and shall contain the two strings “right” and “left”, in that order. Section 7.5.3 (Required Attributes) specifies that this attribute is required for stereoscopic images.

9.44 originalImageFlag

The `originalImageFlag` attribute shall be of type `int`. The attribute value shall be 1 if the pixel data is an unaltered original from a source such as an electronic camera, a film scanner or a computer-graphics rendering engine. The attribute value shall be 0 if the pixel data has been altered in any way.

Attribute values other than 0 or 1 are reserved.

Note: “Unaltered original” does not usually mean “raw image sensor data.” Some amount of processing will be necessary to convert raw data from the sensor in an electronic camera or a film scanner into an ACES image container file, but this type of processing is not considered altering the image. The resulting file is still an original.

9.45 owner

The `owner` attribute shall be of type `string`, and shall specify the owner of the image. The string shall contain only printable characters. The string shall contain at least one byte.

9.46 pixelAspectRatio

The `pixelAspectRatio` attribute shall be of type `float`, and shall specify the intended pixel aspect ratio, defined in Section 4, of the displayed image. The value shall be greater than zero. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

9.47 recorderFirmwareVersion

The `recorderFirmwareVersion` attribute shall be of type `string`, and shall specify the firmware version of the recorder. The string may contain any printable characters. The string shall contain at least one byte.

9.48 recorderMake

The recorderMake attribute shall be of type `string`, and shall specify the manufacturer or vendor of the recorder. The string shall contain only printable characters. The string shall contain at least one byte.

9.49 recorderModel

The `recorderModel` attribute shall be of type `string`, and shall specify the model name or model number of the recorder. The string shall contain only printable characters. The string shall contain at least one byte.

9.50 recorderSerialNumber

The `recorderSerialNumber` attribute shall be of type `string`, and shall specify the serial number of the recorder. The string may contain any printable characters, not only digits. The string shall contain at least one byte.

9.51 reelName

The `reelName` attribute shall be of type `string`, and shall specify a name for a sequence of unique images. The string shall contain only printable characters. The string shall contain at least one byte.

Note: The value is often a reel name, tape name, clip name, or reel number, and can be used as a source identifier in an EDL (Edit Decision List). Some EDL systems require that the value is restricted to 1-8 alphanumeric characters (A..Z, a..z, 0..9) or underscore (_).

9.52 screenWindowCenter

The `screenWindowCenter` attribute shall be of type `v2f`, and shall specify the x and y coordinates of the center of the screen window. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

9.53 screenWindowWidth

The `screenWindowWidth` attribute shall be of type `float`, and shall specify the width of the screen window. The value shall be greater than zero. Section 7.5.3 (Required Attributes) specifies that this attribute is required.

9.54 storageMediaSerialNumber

The `storageMediaSerialNumber` attribute shall be of type `string`, and shall specify the serial number of the physical medium on which the camera output is stored. The string may contain any printable characters, not only digits. The string shall contain at least one byte.

9.55 timeCode

The `timeCode` attribute shall be of type `timecode`, and shall specify the time code for the image.

9.56 timecodeRate

The `timecodeRate` attribute shall be of type `int`, and shall specify the timecode rate of the image sequence to which the image belongs. The rate shall be specified in timecodes per second.

9.57 utcOffset

The `utcOffset` attribute shall be of type `float`, and shall specify the offset of the local time from Universal Coordinated Time (UTC) when the image was created or captured. When the attribute is not present, the UTC offset shall be 0. The offset shall be specified in seconds, such that local time = UTC + UTC offset.

9.58 uuid

The `uuid` attribute shall be of type `string`, and shall contain a universally unique identifier (UUID) formatted as specified in IETF RFC 4122. The string is 128 bits long. The identifier assigned to the image shall be such that with a very high probability no other image will ever be assigned the same identifier.

10 Attributes For Stereoscopic Images

The attributes listed in Table 18 may have different attribute values for right and left image.

The name for the right image attribute shall be the name listed in the table.

The name for the left image attribute shall be the prefix "`left.`" followed by the name listed in the table. This prefix corresponds to the naming of image channels.

A left image attribute shall be present only if the attribute values for the right and left images are different.

Table 18 – Names Of Stereoscopic Attributes

`altitude`	`aperture`	`cameraFirmwareVersion`
`cameraIdentifier`	`cameraLabel`	`cameraMake`
`cameraModel`	`cameraPosition`	`cameraSerialNumber`
`cameraUpDirection`	`cameraViewingDirection`	`comments`
`displayWindow`	`expTime`	`focalLength`
`focus`	`imageCounter`	`isoSpeed`
`keyCode`	`latitude`	`lensAttributes`
`lensMake`	`lensModel`	`lensSerialNumber`
`longitude`	`originalImageFlag`	`pixelAspectRatio`
`recorderFirmwareVersion`	`recorderMake`	`recorderModel`
`recorderSerialNumber`	`reelName`	`screenWindowCenter`
`screenWindowWidth`	`storageMediaSerialNumber`	`timeCode`
`uuid`		

11 Reader and Writer Requirements

A reader should be able to process files having a data window of up to 4096 x 3112 pixels, and having up to 8 channels. The size of such a file is less than 200 MBytes. Readers and writers should support larger files.

Annex A Bibliography (Informative)

Technical Introduction to OpenEXR, 2009, http://www.openexr.com/TechnicalIntroduction.pdf

OpenEXR File Layout, 2007, http://www.openexr.com/openexrfilelayout.pdf

ISO 12232:2006, Determination of Exposure Index, ISO Speed Ratings, Standard Output Densitivity, and Recommended Exposure Index

Annex B Sample File (Informative)

The following sample file contains 17 attributes — the 10 required, 5 optional, and custom attributes.

Table B.1 – Sample Attribute List

Attribute name	*Type*	*Size*	*Contents*
acesImageContainerFlag	int	4	1
channels	chlist	55	R 1 0 1 1 G 1 0 1 1 B 1 0 1 1
chromaticities	chromaticities	32	0.7347 0.2653 0.0000 1.0000 0.0001 -0.0770 0.32168 0.33767
compression	compression	1	0
dataWindow	box2i	16	0 0 2047 1091
displayWindow	box2i	16	0 0 2047 1091
lineOrder	lineOrder	1	0
pixelAspectRatio	float	4	1.0
screenWindowCenter	v2f	8	0.0 0.0
screenWindowWidth	float	4	1.0
uuid	string	16	0x0fef0886f5f8bb9f4bba7c9e6f24c954
cameraModel	string	15	Ikonoskop A-CAM
framesPerSecond	rational	8	25/1
originalImageFlag	int	4	1
timeCode	timecode	8	84279300 0
software	string	27	Sample Software Version 0.1

This hex dump covers the beginning of the same file, including the start of the line offset table.

```
00000000: 76 2F 31 01 02 04 00 00 61 63 65 73 49 6D 61 67  v/1.....acesImag
00000010: 65 43 6F 6E 74 61 69 6E 65 72 46 6C 61 67 00 69  eContainerFlag.i
00000020: 6E 74 00 04 00 00 00 01 00 00 00 63 68 61 6E 6E  nt.........chann
00000030: 65 6C 73 00 63 68 6C 69 73 74 00 37 00 00 00 52  els.chlist.7...R
00000040: 00 01 00 00 00 00 00 00 00 01 00 00 00 01 00 00  ................
00000050: 00 47 00 01 00 00 00 00 00 00 00 01 00 00 00 01  .G..............
00000060: 00 00 00 42 00 01 00 00 00 00 00 00 00 01 00 00  ...B............
00000070: 00 01 00 00 00 00 63 68 72 6F 6D 61 74 69 63 69  ......chromatici
00000080: 74 69 65 73 00 63 68 72 6F 6D 61 74 69 63 69 74  ties.chromaticit
00000090: 69 65 73 00 20 00 00 00 4D 15 3C 3F 67 D5 87 3E  ies. ...M.<?g..>
000000A0: 00 00 00 00 00 00 80 3F 17 B7 D1 38 2D B2 9D BD  .......?...8-...
000000B0: 3E B3 A4 3E 15 E3 AC 3E 63 6F 6D 70 72 65 73 73  >..>...>compress
000000C0: 69 6F 6E 00 63 6F 6D 70 72 65 73 73 69 6F 6E 00  ion.compression.
000000D0: 01 00 00 00 00 64 61 74 61 57 69 6E 64 6F 77 00  .....dataWindow.
000000E0: 62 6F 78 32 69 00 10 00 00 00 00 00 00 00 00 00  box2i...........
000000F0: 00 00 FF 07 00 00 43 04 00 00 64 69 73 70 6C 61  ......C...displa
00000100: 79 57 69 6E 64 6F 77 00 62 6F 78 32 69 00 10 00  yWindow.box2i...
00000110: 00 00 00 00 00 00 00 00 00 00 FF 07 00 00 43 04  ..............C.
00000120: 00 00 6C 69 6E 65 4F 72 64 65 72 00 6C 69 6E 65  ..lineOrder.line
00000130: 4F 72 64 65 72 00 01 00 00 00 00 70 69 78 65 6C  Order......pixel
00000140: 41 73 70 65 63 74 52 61 74 69 6F 00 66 6C 6F 61  AspectRatio.floa
00000150: 74 00 04 00 00 00 00 00 80 3F 73 63 72 65 65 6E  t........?screen
00000160: 57 69 6E 64 6F 77 43 65 6E 74 65 72 00 76 32 66  WindowCenter.v2f
00000170: 00 08 00 00 00 00 00 00 00 00 00 00 00 00 73 63 72  ..............scr
00000180: 65 65 6E 57 69 6E 64 6F 77 57 69 64 74 68 00 66  eenWindowWidth.f
00000190: 6C 6F 61 74 00 04 00 00 00 00 00 80 3F 75 75 69  loat........?uui
000001A0: 64 00 73 74 72 69 6E 67 00 20 00 00 00 30 66 65  d.string. ...0fe
000001B0: 66 30 38 38 36 66 35 66 38 62 62 39 66 34 62 62  f0886f5f8bb9f4bb
000001C0: 61 37 63 39 65 36 66 32 34 63 39 35 34 61 64 6F  a7c9e6f24c954ado
000001D0: 70 74 65 64 4E 65 75 74 72 61 6C 00 76 32 66 00  ptedNeutral.v2f.
000001E0: 08 00 00 00 3E B3 A4 3E 15 E3 AC 3E 63 61 6D 65  ....>..>...>came
000001F0: 72 61 4D 6F 64 65 6C 00 73 74 72 69 6E 67 00 0F  raModel.string..
00000200: 00 00 00 49 6B 6F 6E 6F 73 6B 6F 70 20 41 2D 43  ...Ikonoskop A-C
00000210: 41 4D 66 72 61 6D 65 73 50 65 72 53 65 63 6F 6E  AMframesPerSecon
00000220: 64 00 72 61 74 69 6F 6E 61 6C 00 08 00 00 00 19  d.rational......
00000230: 00 00 00 01 00 00 00 6F 72 69 67 69 6E 61 6C 49  .......originalI
00000240: 6D 61 67 65 46 6C 61 67 00 69 6E 74 00 04 00 00  mageFlag.int....
00000250: 00 01 00 00 00 74 69 6D 65 43 6F 64 65 00 74 69  .....timeCode.ti
00000260: 6D 65 63 6F 64 65 00 08 00 00 00 04 00 06 05 00  mecode..........
00000270: 01 00 00 73 6F 66 74 77 61 72 65 00 73 74 72 69  ...software.stri
00000280: 6E 67 00 1B 00 00 00 53 61 6D 70 6C 65 20 53 6F  ng.....Sample So
00000290: 66 74 77 61 72 65 20 56 65 72 73 69 6F 6E 20 30  ftware Version 0
000002A0: 2E 31 00 C3 24 00 00 00 00 00 00 CB 54 00 00 00  .1..$.......T...
000002B0: 00 00 00 D3 84 00 00 00 00 00 00 DB B4 00 00 00  ................
000002C0: 00 00 00 E3 E4 00 00 00 00 00 00 EB 14 01 00 00  ................
000002D0: 00 00 00 F3 44 01 00 00 00 00 00 FB 74 01 00 00  ....D.......t...
```

This hex dump shows the end of scan line 0 and beginning of scan line 1, starting at 0x54CB.

```
000054B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000054C0: 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00  ................
000054D0: 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0...............
000054E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000054F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00005500: 00 00 00 00 00 00 00 D7 24 71 28 CD 28 04 29 DA  ........$q(.(.).
00005510: 28 B3 28 CF 28 00 29 F1 28 E9 28 EC 28 E6 28 B0  (.(.(.).(.(.(.(.
```