Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Internal reference number of this OGC® document: 21-061

Version: n.n

Category: OGC® <Implementation Standard / Abstract Specification / Best Practice>

Editor: Joan Maso

OGC BigTIFF community standard

Copyright notice

Warning

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OGC® <Standard/Abstract Specification/Best Practice>
Document subtype:	if applicable
Document stage:	Draft
Document language:	English

License Agreement

Table of Contents

i. Abstract

This document is OGC Standard that describes an extended internal structure for a image file that requires more than 4GBytes. The general structure is similar to the TIFF specification version 6 but it replaces two essential data structures called Image File Header and Image File Directory for new ones that support the necessary longer integer numbers that acts as internal offsets of the file. It also adds 3 new data types for very long integers. It is expected that other OGC standards, such as GeoTIFF, can support BigTIFF in the near future.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, TIFF, image format

iii. Preface

When the TIFF format was defined back in 1986 the storage capacity of the computers made difficult to image that a single file could ever be bigger than 4GBytes long. But today capacities makes it possible and if the file can be random accessed in small pieces, could be even useful to have a big remote sensing product in a single file.

The basic BigTIFF design was first proposed in 2004 and refined in (this discussion on the (Aware Systems mailing list. Contributors to the design discussion included Joris Van Damme, Lynn Quam, Frank Warmerdam, Chris Cox, Rob Tillaart, Dan Smith, Bob Freisenhahn, Andrey Kiselev, Phillip Crews, and Gerben Vos. The BigTIFF format was proposed in https://www.awaresystems.be/imaging/tiff/bigtiff.html maintained by Joris Van Damme and has been stable in recent years.

GeoTIFF 1.1 was brought to the OGC as a community standard but, in the process, was made dependent form the classical TIFF specification. This document brings BigTIFF to the OGC as another community standard. It is expected that Next version of GeoTIFF supports both Classical TIFF and BigTIFF.

This document is an adaptation of the https://www.awaresystems.be/imaging/tiff/bigtiff.html content to an OGC language to bring it to the OGC standards process as a community standard. It was done with the intention of creating a transcription with no practical modifications, so old BigTIFF implementations would be compatible with this document.

NOTE Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Security Considerations

No security considerations have been made for this Standard.

v. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- UAB-CREAF
- Geomatys
- IGN-France

vi. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name	Affiliation
Joan Maso	UAB-CREAF
Martin Desruisseaux	Geomatys
Gilles Cebelieu	IGN-F
Michel Leedahl	Michel Leedahl

Chapter 1. Scope

This document is OGC Standard that describes an extended internal structure for a image file that requires more than 4GBytes. The general structure is similar to the TIFF specification version 6 but two essential data structures called Image File Header and Image File Directory are replaced by Big File Header and Big Image File Directory to support the necessary longer integer numbers that act as internal offsets of the file. It also adds 3 new extra data types: LONG8 (uint64) SLONG8 (int64) or IFD8 (uint64).

Chapter 2. Conformance

This Standard defines a single conformance class

BigTIFF requirements (http://www.opengis.net/spec/BigTIFF/1.0/req/req-class-core)

This class defines the requirements of the BigTIFF format.

Conformance with this Standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

In order to conform to this OGC Standard, a software implementation shall choose to implement: * Any one of the conformance levels specified in Annex A (normative).

All requirements-classes and conformance-classes described in this document are owned by the Standard(s) identified.

Chapter 3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

Adobe Developers Association, TIFF Revision 6.0 Final, June 3, 1992, Adobe Systems Incorporated, https://www.adobe.io/open/standards/TIFF.html

NOTE This Standard includes requirements that overwrite some basic data structures specified in Section 2 of the TIFF Revision 6.0. The rest of the TIFF Revision 6.0 still applies to this Standard.

Chapter 4. Terms and Definitions

This document used the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the 'ModSpec'. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

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

big image file directory

a modified data structure containing the TIFF fields (identified by a tag) of a BigTIFF file.

big image file header

a modified data structure at the beginning of a BigTIFF file.

classic TIFF

a TIFF file that follows the TIFF specification version 6.0.

field

each entry on a image file directory or a big image file directory. A field is characterized by a tag numerical value and contains or points to the data (source: TIFF specification version 6.0)

image file directory

a data structure containing a list of TIFF fields (identified by a tag) of a TIFF file. (source: TIFF specification version 6.0)

image file header

a data structure at the beginning of a TIFF file. (source: TIFF specification version 6.0)

tag

the identifying number of a field type in the image file directory or the big image file directory. (source: TIFF specification version 6.0)

NOTE In previous versions of the TIFF specification, the term “tag” referred both to the identifying number of a TIFF field and to the entire field.

TIFF

a tag-based file format for storing and interchanging raster images (source: TIFF specification version 6.0)

Chapter 5. Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Abbreviations

BIFD	BigTIFF Image File Directory
IFD	Image File Directory
TIFF	Tag Image File Format

5.2. Identifiers

The normative provisions in this Standard are denoted by the URI

http://www.opengis.net/spec/bigtiff/1.0

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

Chapter 6. Overview

6.1. What is BigTIFF

The TIFF file format was defined in mid-1980s as an extensible format. At that time 32bit offsets seemed sufficient, giving a limit to a four GByte file size. This has been quite sufficient for many years. Today however, there is a need for a good multi-purpose open image file format that can handle huge images, or very large collections of images, that together exceed the four GByte boundary.

BigTIFF is very similar to the original TIFF and only changes some internal headers to support 64 bits offsets and data types. The benefits of BigTIFF closely resembling TIFF are huge. For instance, existing TIFF libraries can quite easily extend their support for TIFF to also include this variant. This results in a very short list of requirements. All the much-appreciated properties of the TIFF format that have been around and have been extended for decades are inherited. All properly known tags are being reused, all supported bit depths and datatypes remain valid. The arbitrary number of 'extra channels', the tiling and striping schemes, the multitude of compression schemes, and the private tag scheme, that made TIFF very useful in pre-press as well as for storing scientific data, and many other applications, all remain intact. Yet, by changing the offset bit depth, BigTIFF files are no longer restrained by the 4 GByte limitation from which classic TIFF suffers.

Since the BigTIFF format is based on the TIFF format, the reader is encouraged to familiarize with the current version of the TIFF specification before reading this.

6.2. Origin and compatibility

This Standard collects the attempt to launch a new variant of TIFF, called BigTIFF, that closely resembles TIFF, but uses 64bit offsets instead. The results of the BigTIFF discussion resulted in an original specification exposed in the Aware Systems website: https://web.archive.org/web/20240706160214/https://www.awaresystems.be/imaging/tiff/bigtiff.html. Aware Systems was managed by Joris Van Damme but was discontinued after July 6th, 2025 and can only be accessible in backup copies done by the WebAchive iniciative. In order to preserve and consolidate the BigTIFF iniciative, the process of creating this Standard started with Joris Van Damme initial work and was translated it into an OGC standard. Another source of information on the BigTIFF format is the BigTIFF design page at the LibTiff site http://bigtiff.org/.

There are at least two independent implementations of the original BigTIFF specification. (AWare Systems AsTiff, and LibTiff since version 4.0). This OGC Standard does not contradicts the original BigTIFF specification so the implementations of BigTIFF are immediatelly compatible with this Standard.

6.3. Changes to the original version

In the elaboration of this Standard a new naming for the data types have been adopted for clarity.

Table 1. Changes in data types.

Original	Adopted	Description
Word	uint16	unsigned short integer (2 bytes)
Unsigned long	uint32	unsigned long integer (4 bytes)
Signed 8Byte	int64	signed long long integer (8 bytes)
Unsigned 8Byte	uint64	unsigned long long integer (8 bytes)

Chapter 7. BigTIFF requirements

The BigTIFF standard provides a solution for a TIFF file that requires a file size larger than 4 Gigabytes. In principle, the BigTIFF standard will also support smaller files. However, files that do not require file size bigger than 4 Gigabytes or not using the new data types, should use TIFF version 6 specification instead. This is in order to ensure backwards compatibility with old TIFF implementations not supporting BigTIFF.

A BigTIFF file (and a TIFF) is defined as a sequence of 8-bit bytes, where the bytes are numbered from 0 to N. In a BigTIFF file, the largest possible TIFF file is 2^{64} bytes in length.

7.1. Requirement Class Core

Requirements Class Core	
http://www.opengis.net/spec/BigTIFF/1.0/req/req-class-core	
Target type	BigTIFF Encoder
Dependency	https://www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf (except for the Section 2 that is replaced by the requirements of this Standard)

7.1.1. Basic format

Permission 1	**/per/core/big-offsets**
A	A BigTIFF is allowed to ignore the introduction and the first 2 subsections in Section 2 of the TIFF version 6 specification describing the *Image File Header* and the *Image File directory* and use a *Big Image File Header* and a *Big Image File directory* instead. When some part of the the *Image File directory* subsection are still relevant for this standard, they are explicitly referenced

NOTE Commonly "permissions" are statements that emphasizes the usefulness of some possibilities that may be not obvious but do not contractic requirements. In this case we are forced to use a permission to overwrite requirements in the original TIFF specification and allow for 64 bit offsets.

Requirement 1	/req/req-class-core/big-offsets
A	A BigTIFF SHALL begin with a 16 byte *big image file header* (that replaces the TIFF v6 8 byte *image file header*)
B	A *big image file header* SHALL point to the first *big image file directory* (BIFD). A *big image file directory* contains information about the image, as well as pointers to the actual image data in a similar way as a *image file directory* does in TIFF version 6

Requirement 2	/req/req-class-core/bifh
A	A *big image file header* SHALL follow the structure presented in Table 2

Table 2. BigTIFF file header (normative)

Offset	Datatype	Value
0	**uint16**	**Byte order indication (use the values stated in the TIFF version 6 Image File Header Bytes 0-1)**
2	**uint16**	**Version number (always 43)**
4	**uint16**	**Size of offset in bytes Always 8 in BigTIFF (it could be 16 in a future version for superbig TIFFs)**
6	**uint16**	**Reserved. Populate with 0**
8	**uint64**	**Offset (in bytes) to first BIFD**

(The Classic TIFF v6 basic data structures and complete list of BigTIFF data types (Informative) contains the TIFF version 6 classic structures for comparison purposes).

While the classic TIFF *image file header* points to the first *Image File Directory* (IFD), the BigTIFF *big image file header* points to the first *Big Image File Directory* (BIFD). In that sense IFD and BIFD are equivalent. Both IFD and BIFD can be located anywhere in the file. Every 'page' in a multi-page TIFF, either classic TIFF or BigTIFF, is represented by exactly one IFD and one BIFD respectively. Please note that BIFD and IFD can not be used together in the same file.

Requirement 3	/req/req-class-core/bifd
A	There SHALL be at least ONE *big image file directory* in a TIFF file and each BIFD must have at least one entry.
B	A *big image file directory* SHALL follow the structure of *Big Directory Entries* presented in Table 3
C	The *big directory entries* in a BIFD SHALL be sorted by code (in the same way than classic TIFF does)
D	Each *big directory entry* SHALL follow the structure presented in Table 4
E	The tag *value* SHALL be written inside the tag structure, in the *big directory entry*, if its size is smaller than or equal to 8 bytes. Otherwise, it’s written elsewhere, and pointed to by the offset written as the value
F	In case of multibyte values (e.g a SHORT, LONG, etc), the byte order (II or MM) in the TIFF header SHALL govern the order of the bytes

G	The reader SHALL check the type to verify that it contains an expected value. TIFF allows for more than one valid type for some fields. For example, ImageWidth and ImageLength are usually specified as having type SHORT, but images with more than 64K rows or columns should use the LONG field type. TIFF readers SHALL accept BYTE, SHORT, LONG or LONG8 values for any unsigned integer field. This allows a single procedure to retrieve any integer value, and makes reading more robust, and saves disk space in some situations

A *Big Image File Directory* (BIFD) consists of an 8-byte count of the number of directory entries (i.e., the number of fields), followed by a sequence of 20-byte field entries, followed by an 8-byte offset of the next BIFD (or 0 if no other BIFD is required).

Every *Big Directory Entry* takes up exactly 20 bytes in BigTIFF, and the complete BIFD looks like this:

Table 3. BigTIFF Image File Directory (BIFD)

Offset	Datatype	Value
0	uint64	Number of big directory entries in BIFD (B)
8+0*20	Big directory entry structure (see Table 4)	First tag data
8+1*20	Big directory entry structure (see Table 4)	Second tag data
…		
8+(B-1)*20	Big directory entry structure (see Table 4)	Last tag data
8+B*20	uint64	Offset to next BIFD, if there is a next BIFD or 0 otherwise

A *TIFF field* is a logical entity consisting of TIFF tag and its value. This logical concept is implemented as a *Directory Entry* in the TIFF version 6 and it is implemented in here as a *Big Directory Entry*, plus the actual value if it does not fit into the value/offset part, the last 8 bytes of the BIFD Entry.

Table 4. BigTIFF Tag structure

Offset	Datatype	Value
0	uint16	Tag identifying code [1]
2	uint16	Datatype of tag data. See Table 5
4	uint64	Number of values (it is not the total number of bytes but the number of values)
12	x * Tag data or uint64 offset	Tag data or offset to tag data

Offset	Datatype	Value
[1] **See a list of codes in the TIFF version 6 specification. The GeoTIFF standard adds some extra TIFF tag identifiers.**		

Each TIFF field has an associated Count. This means that all fields are actually one-dimensional arrays, even though most fields contain only a single value.

The same rule for value 'inlining' the tag data applies to both classic TIFF and BigTIFF, only the threshold size differs. Above the threshold values are pointed by an offset and written elsewhere.

The following table contains the Tag datatypes that are defined in BigTIFF, in addition to the ones defined in the TIFF specifications (see the last part of Table 5)

Table 5. BigTIFF Datatype Tag values

Value	Name	Description	Source
16	LONG8 (uint64)	64-bit (8-byte) unsigned integer	BigTIFF
17	SLONG8 (int64)	64-bit (8-byte) signed integer	BigTIFF
18	IFD8	A uint64 IFD offset	BigTIFF

Permission 2	/per/core/long8
A	StripOffsets, StripByteCounts, TileOffsets, and TileByteCounts tags are allowed to have the datatype LONG8 in BigTIFF when necessary. LONG, and SHORT are still valid
B	Tags that point to other IFDs, such as the SubIFDs tag (see TIFF PageMaker version 6), are allowed to have the datatype IFD8. The datatypes IFD (and the not recommendable LONG) are still valid

Chapter 8. Media Types for any data encoding(s)

IANA defines the image/tif media type for TIFF (https://www.iana.org/assignments/media-types/image/tiff). Since the software is able to programaticaly differenciate between a TIFF and a BigTIFF, we suggest the use of the same media time for the BigTIFF.

We also suggest the use of the same file extension ('tif' or 'tiff') despite other suggestions such as 'tf8' and 'btf' (see discussion in the LibTiff mailing list).

Annex A: Conformance Class Abstract Test Suite (Normative)

This is the Abstract test of this BigTIFF Standard. However to implement a complete test on a BigTIFF file the dependency on TIFF version 6 should be taken into account. Unfortunately the TIFF version 6 file does not contain an abstract test to guide this developement

A.1. Conformance Class Core

A.1.1. Requirement 1

Test id:	/conf/conf-class-core/bif-offsets
Requirement:	/req/req-class-core/bifd-offsets
Test purpose:	Verify that the section *big image file header* uses 8 bit offsets.
Test method:	Inspect that aBigTIFF begins with a 16 byte *big image file header* (that replaces the TIFF v6 8 byte *image file header*) Inspect that the *big image file header* points to the first *big image file directory* (BIFD)

A.1.2. Requirement 2

Test id:	/conf/conf-class-core/bifh
Requirement:	/req/req-class-core/bifh
Test purpose:	Verify that the section *big image file header* exists and is correctly encoded.
Test method:	Inspect that a *big image file header* follows the structure presented in Table 2

A.1.3. Requirement 3

Test id:	/conf/conf-class-core/bifd
Requirement:	/req/req-class-core/bifd
Test purpose:	Verify that the section *big image file directory* exists and is correctly encoded.

Test method:	Inspect that there is at least ONE *big image file directory* in a TIFF file and that each BIFD have at least one entry. Inspect that a *big image file directory* follows the structure of *Big Directory Entries* presented in Table 3 Inspect that *big directory entries* in a BIFD are sorted by code (in the same way than classic TIFF does) Inspect that each *big directory entry* follows the structure presented in Table 4 Inspect that the tag *value* is written inside the tag structure, in the *big directory entry*, if its size is smaller than or equal to 8 bytes. Otherwise, inspect that it's written elsewhere, and pointed to by the offset written as the value In case of multibyte values (e.g a SHORT, LONG, etc), Inspect that the byte order (II or MM) in the TIFF header governs the order of the bytes In case of a client/reader implementation, inspect it checks the type to verify that it contains an expected value. TIFF allows for more than one valid type for some fields. Inspect that the reader accept BYTE, SHORT, LONG or LONG8 values for any unsigned integer field.

Annex B: Classic TIFF v6 basic data structures and complete list of BigTIFF data types (Informative)

B.1. Image File Header

For comparison purposes, we reproduce the original TIFF version 6 TIFF header here.

Table 6. Classic TIFF file header (informative)

Offset	Datatype	Value
0	uint16	Byte order indication
2	uint16	Version number (always 42)
4	uint32	Offset to first IFD

B.2. Image File Directory

For comparison purposes, we reproduce the original TIFF version 6 Image File Directory here.

Table 7. Classic IFD (informative)

Offset	Datatype	Value
0	uint16	Number of tags in IFD (B)
2+0*12	Tag structure	First tag data
2+1*12	Tag structure	Second tag data …
2+(B-1)*12	Tag structure	Last tag data
2+B*12	uint32	Offset to next IFD, if there is a next IFD or 0 otherwise

Every tag takes up exactly 12 bytes in classic TIFF, and looks like this:

Table 8. Classic Tag structure (informative)

Offset	Datatype	Value
0	uint16	Tag identifying code
2	uint16	Datatype of tag data
4	uint32	Number of values
8	x * Tag data datatype or uint32 offset	Tag data or offset to tag data

In classic TIFF, the tag data was written inside the tag structure, in the IFD, if its size was smaller than or equal to 4 bytes. Otherwise, it’s written elsewhere, and pointed to.

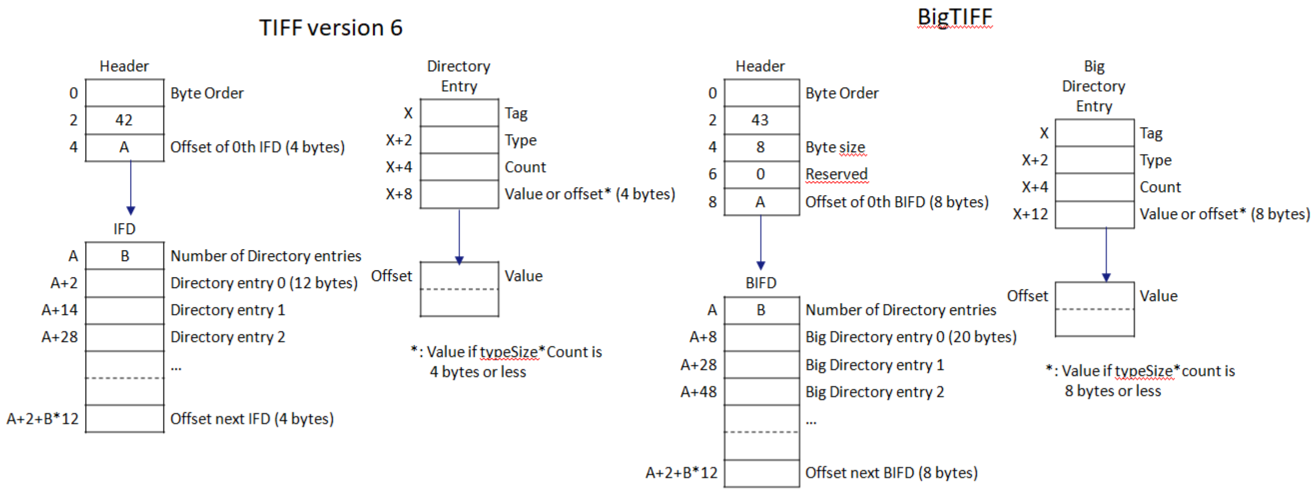
TIFF version 6
Header
0
2
42
4
A
Byte Order
Offset of 0th IFD (4 bytes)
IFD
A
B
A+2
A+14
A+28
A+2+B*12
Number of Directory entries
Directory entry 0 (12 bytes)
Directory entry 1
Directory entry 2
...
Offset next IFD (4 bytes)
Directory Entry
X
X+2
X+4
X+8
Tag
Type
Count
Value or offset* (4 bytes)
Offset
Value
*: Value if typeSize*Count is 4 bytes or less
BigTIFF
Header
0
2
43
4
8
6
0
8
A
Byte Order
Byte size
Reserved
Offset of 0th BIFD (8 bytes)
BIFD
A
B
A+8
A+28
A+48
A+2+B*12
Number of Directory entries
Big Directory entry 0 (20 bytes)
Big Directory entry 1
Big Directory entry 2
...
Offset next BIFD (8 bytes)
Big Directory Entry
X
X+2
X+4
X+12
Tag
Type
Count
Value or offset* (8 bytes)
Offset
Value
*: Value if typeSize*count is 8 bytes or less

Figure 1. TIFF and BigTIFF data structures side by side.

B.3. Datatype Tag values

The following table summarizes the Tag datatypes aplicable for BigTIFF. This list includes "Types" described in the TIFF version 6 specification, as well as some extra types that have been defined by the BigTIFF (this Standard).

Table 9. BigTIFF Datatype Tag values

Value	Name	Description	Source
1	BYTE (uint8)	8-bit unsigned integer	TIFF specification
2	ASCII	8-bit byte that contains a 7-bit ASCII code; the last byte SHALL be a NUL character (binary zero)	TIFF specification
3	SHORT (uint16)	16-bit (2-byte) unsigned integer	TIFF specification
4	LONG (uint32)	32-bit (4-byte) unsigned integer	TIFF specification
5	RATIONAL	Two LONGs: the first represents the numerator of a fraction; the second, the denominator	TIFF specification
6	SBYTE	An 8-bit signed (twos-complement) integer	TIFF version 6
7	UNDEFINED	An 8-bit byte that may contain anything, depending on the definition of the field [a]	TIFF version 6
8	SSHORT (int16)	A 16-bit (2-byte) signed (twos-complement) integer	TIFF version 6
9	SLONG (int32)	A 32-bit (4-byte) signed (twos-complement) integer.	TIFF version 6

Value	Name	Description	Source
10	SRATIONAL	Two SLONGs: the first represents the numerator of a fraction, the second the denominator	TIFF version 6
11	FLOAT	Single precision (4-byte) IEEE format	TIFF version 6
12	DOUBLE	Double precision (8-byte) IEEE format	TIFF version 6
13	IFD	A uint32 IFD offset	TIFF PageMaker version 6
16[b]	LONG8 (uint64)	64-bit (8-byte) unsigned integer	BigTIFF
17	SLONG8 (int64)	64-bit (8-byte) signed integer	BigTIFF
18	IFD8	A uint64 IFD offset	BigTIFF

[a] It can be used to store a complicated data structure in a single private field, The Count will be the number of bytes required to hold the data structure.

[b] numbers 14 and 15 have no meaning.

Annex C: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-28	0.1	G. Editor	all	initial version

Annex D: Bibliography

[1] Joris Van Damme. The BigTIFF File Format. In Internet: https://web.archive.org/web/20240706160214/https://www.awaresystems.be/imaging/tiff/bigtiff.html

[2] Welcome to the BigTIFF version of the libtiff library. In Internet: http://bigtiff.org/