OGC BigTIFF standard

Abstract

This document is OGC Standard that describes an extended internal structure for an image file that requires more than 4 Gibibytes (2^{32}). 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, Will depend on this OGC Standard in the near future.

Preface

When the TIFF format was defined back in 1986 the storage capacity of the computers made difficult (or even impossible) for an single file image to be bigger than Gibibytes (2^{32}) long. But today capacities makes it viable and if the file can be random accessed in small pieces, could be useful for example in the case of a big remote sensing dataset that is delivered in a single TIFF file.

The basic BigTIFF design was first proposed in 2004 and refined in (this discussion (now only accessible in the web.archive.org) on the Aware Systems (Aware Systems mailing list (now only available in the web.archive.org). Contributors to the design discussion included Lynn Quam, Frank Warmerdam, Chris Cox, Rob Tillaart, Dan Smith, Bob Freisenhahn, Andrey Kiselev, Phillip Crews, and Gerben Vos. The authors thank all those who came before them for creating libtiff and designing the BigTIFF enhancements. 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 (now only accessible in the web.archive.org).

Changes were made by Ole Eichhorn (ole.eichhorn@gmail.com) while at (www.aperio.com)[Aperio] (now only accessible in the web.archive.org) and the work was donated as public domain, in gratitude to Sam Leffler, Silicon Graphics, Joris Van Damme, Aware Systems, Frank Warmerdam, Andrey Kisley, Mike Welles, and all who have worked on libtiff over the years to provide support to files bigger than previous limitations. These changes were published on an "as is" basis and neither Ole Eichhorn nor Aperio made any warranty as to their fitness for any intended use.

The intention of this Standard is to elevate the work done by all cited previous people to the category of International Standard recognizing and respecting their work. This document is an adaptation of the https://www.awaresystems.be/imaging/tiff/bigtiff.html specification to an OGC Standard format following the OGC Modular Specification. It was done with the intention of creating a transcription with no practical modifications, so old BigTIFF implementations would be immediately compatible with this Standard. In the process, the Standard was made partially dependent from the classical TIFF v6 specification. It is expected that next versions of GeoTIFF will support both classical TIFF v6 and BigTIFF overcoming file size limitations.

NOTE	Recipients of this Standard 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.

Security considerations

While this Standard upgrades TIFF by accommodating capacity for massive datasets (such as satellite imagery, gigapixel medical pathology slides, and high-fidelity 3D maps) it introduces specific, security vulnerabilities. The following list is not exhaustive but highlights the most critical concerns:

Memory Allocation Bombs (Severe DoS): With BigTIFF's 64-bit fields, a malicious actor can craft an Image File Directory (IFD) tag indicating that an array requires exabytes or petabytes of memory. If a parser software naively executes memory allocation (malloc) based on that 64-bit header value before validating the actual file payload size, it will cause an immediate application crash or severe system-wide memory exhaustion.

64-bit to 32-bit Truncation Vulnerabilities: Legacy applications handle file pathways using 32-bit code. When a modern BigTIFF library passes a 64-bit offset or byte-count pointer to an older code, the upper bits can be truncated. This type mismatch creates scenarios leading to out-of-bounds memory reads or heap corruptions.

Unchecked Malware Staging & Steganography: Because BigTIFF files easily scale into tens of gigabytes, they are perfect vehicles for data exfiltration or malware storage. Most enterprise antivirus software, firewalls, and email scanners bypass deep content scanning for files exceeding 50 MB to 100 MB to maintain performance. Attackers can embed entire encrypted payloads into obscure 64-bit tag blocks without altering the viewable image, completely avoiding detection.

Infinite Tiles: BigTIFF heavily relies on tiled structures to read giant images efficiently. A corrupted or malicious file can define millions of deeply nested that may lock up server CPU threads almost indefinitely.

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

Scope

This document is OGC Standard that describes an extended internal structure for a image file that requires more than 4 Gibibytes (2^{32}). 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).

The intention of this Standard is to elevate the work done by the original creators of BigTIFF to the category of International Standard recognizing and respecting their work. Introducing changes in the original work is out of scope of this Standard version.

NOTE	In this Standard Gibibytes (2^{32}) is used instead of the most common Gigabytes (10^{9}) because the limit in the file format size is related to internal offsets expressed using 32 bit long integer numbers.

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.

It is expected that applications implementing the previous proposed BigTIFF specifications will be compatible with this OGC Standard. However, it is recommended that those implementations are also tested for conformance using the tests specified in Annex A (normative) of this OGC Standard in order to be able to declare OGC Compliance.

In order to conform to this OGC® interface standard, a software implementation shall choose to implement:

- The conformance levels specified in Annex A (normative).

The single requirements-class and conformance-class described in this Standard are owned by the standard(s) identified.

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.

- [TIFF Revision 6.0 Final], Adobe Developers Association, TIFF Revision 6.0 Final, June 3, 1992, Adobe Systems Incorporated, https://www.itu.int/itudoc/itu-t/com16/tiff-fx/docs/tiff6.pdf

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 is still a normative reference to this Standard.

Terms and Definitions

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

big image file directory

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

big image file header

a modified *image file header* data structure at the beginning of a BigTIFF file.

big TIFF or BigTIFF

a TIFF file that follows this Standard.

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)

[TIFF Revision 6.0 Final]

gibibyte

2^{30} bytes (1 073 741 824 bytes)

NOTE	this number is similar but not equal to a gigabyte that is 10^9* bytes (1 000 000 000 or a billion bytes).

ISO/IEC 80000-13

image file directory

a data structure containing a list of TIFF fields (identified by a tag) of a *classic TIFF* file.

[TIFF Revision 6.0 Final]

image file header

a data structure at the beginning of a *classic TIFF* file.

[TIFF Revision 6.0 Final]

offset

location in bytes with respect to the beginning of the TIFF file. The first byte of the file has an offset of 0.

[TIFF Revision 6.0 Final]

tag

the identifying number of a field type in the *image file directory* or the *big image file directory*.

[TIFF Revision 6.0 Final]

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

[TIFF Revision 6.0 Final]

Conventions

These sections provide details and examples for any conventions used in this Standard. Examples of conventions are symbols, abbreviations, or special notes regarding how to read the Standard.

Abbreviations

BIFD	BigTIFF Image File Directory
BIFH	BigTIFF Image File Header
IFD	Image File Directory
IFH	Image File Header
TIFF	Tag Image File Format

Identifiers

The normative provisions in this Standard are denoted by the URI

`https://www.opengis.net/spec/bigtiff/1.0`

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

Overview

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 4 Gibibytes (2^{32}) 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 4 Gibibytes (2^{32}) 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 Gibibytes (2^{32}) 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.

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.

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	twos-complement signed long long integer (8 bytes)
Unsigned 8Byte	uint64	unsigned long long integer (8 bytes)

Implications for EXIF

EXIF is a metadata standard that uses a TIFF-compatible structure such as *image file header* and *image file directory* as a container. It adopts many of the accepted [TIFF Revision 6.0 Final] tags and provides an extra list of EXIF tags and a pointer to a dedicated EXIF sub-IFD.

EXIF extra tags define a photography-focused subset and extends it with standardized metadata fields for digital cameras. It captures information such as exposure, date/time, GPS, and thumbnail-related data. For TIFF files, EXIF information is integrated by adding new TIFF tags. For the JPEG format, EXIF is included in the Application Market Segment (APP1). APP1 is recorded immediately after the SOI marker indicating the beginning of the file.

EXIF specification references Classic TIFF as a data structure and current cameras are not generating images with size bigger that 4Gib. However, nothing prevents a BigTIFF file to use EXIF following the same pattern specified in this Standard (by replacing IFH and IFDs with BIFH and BIFDs respectively). Implementations of EXIF readers should consider the possibility to find BigTIFF structures when reading EXIF, and the need to transform them to Classic TIFF structures when exporting data and metadata to JPEG or other formats.

NOTE In the future, the structure BigTIFF format could support both IFDs and BIFDs and by saving this EXIF part in the first part of the document this issue could be avoided. However introducing changes in the original BigTIFF specification is out of scope of this Standard.

BigTIFF requirements

The BigTIFF standard provides a solution for a TIFF file that requires a file size larger than 4 Gibibytes (2^{32}). In a BigTIFF file, the largest possible file is 2^{64} bytes in length. In this situation the classic TIFF internal 32 bit offsets are not long enough to point to the a byte beyond the 4 Gibibytes

limit.

Requirement Class Core

Example 1. Requirements Class 'Core'

identifier

https://www.opengis.net/spec/BigTIFF/1.0/req/core

subject

BigTIFF file

inherit

[TIFF Revision 6.0 Final] (except for the Section 2 that is replaced by the requirements of this Standard)

requirement

/req/core/big-offsets

requirement

/req/core/bifh

requirement

/req/core/bifd

NOTE This requirement class has *BigTIFF file* as target of all requirements. Indirectly, the requirements class imposes requirements on a BigTIFF writter (that should deliver a file following these requirements). It also indirectly imposes requirements on how a BigTIFF should operate to understand a BigTIFF file.

identifier

/per/core/per-big-offsets

part

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 a part of the the *Image File directory* subsection are still relevant for this standard, it is explicitly reproduced in this Standard

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

identifier

/rec/core/rec-small-files

part

In principle, the BigTIFF standard could also support smaller than 4 Gibibytes (2^{32}) files. However, files that do not require file size bigger than 4 Gibibytes (2^{32}) or are not using the new data types defined in this Standard SHOULD use Classic TIFF ([TIFF Revision 6.0 Final]) instead of the format defined in this Standard (BigTIFF).

NOTE This is in order to ensure maximum backwards compatibility with old TIFF implementations not supporting BigTIFF.

Basic format

identifier

/req/core/big-offsets

part

A BigTIFF SHALL shall be a file with a size bigger than 4Gibibytes or a TIFF file using using the new data types defined in this Standard

part

A BigTIFF SHALL begin with a 16 byte *big image file header* (that replaces the [TIFF Revision 6.0 Final] 8 byte *image file header*)

part

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 Revision 6.0 Final]

NOTE There is a paradox in the actual 4 Gibibytes limit. In a case that all *Image File Directories* are at the beginning of the file, a 32 bit offset can be sufficient to point to an array of image strips offsets (Tag code 273) and 32 bit numbers may be sufficient for image strips offsets, but imagining that strips are in other and that the second to last strip is at the offset 4.1 10^9 and that the last strip is 250 MegaByte long, the total size of the file will cross the 4 GiB boundary. In a case like this, even if BigTIFF it is not technically necessary, a BigTIFF is mandated by this Standard. This provides a clear criteria to discriminate the need for a BigTIFF file that is independent from order and size of the internal parts of the file.

identifier

/req/core/bifh

part

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 TIFF file. Every 'page' in a multi-page TIFF, either classic TIFF or BigTIFF, is represented by exactly one IFD and one BIFD respectively.

identifier

/req/core/bifd

part

There SHALL be at least ONE *big image file directory* in a BigTIFF file and each BIFD must have at least one entry

part

A BigTIFF file SHALL NOT contain any classic TIFF *image file directory*. In other words, BIFD and IFD can not be used together in the same TIFF file.

part

A *big image file directory* SHALL follow the structure of *Big Directory Entries* presented in Table 3

part

The *big directory entries* in a BIFD SHALL be sorted by code (in the same way than classic TIFF requires)

part

Each *big directory entry* SHALL follow the structure presented in Table 4

part

The tag *value* SHALL be included inside the tag structure, in the *big directory entry*, if its size is smaller than or equal to 8 bytes. Otherwise, it SHALL be included elsewhere, and the *value* SHALL be an offset from the beginning of the file.

part

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 these bytes.

part

The value of *Offset to next BIFD* in a BIFD SHALL NOT point to a previous BIFD

part

The `type` in the BIFD SHALL take priority over the specified (in [TIFF Revision 6.0 Final] or elsewhere) `type` of the expected value. In particular, the [TIFF Revision 6.0 Final] 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 will use the LONG field `type`. The `type` of any specified unsigned integer SHALL be expressed as BYTE, SHORT, LONG or LONG8.

The last item in the requirement allows a single procedure to retrieve any integer value, and makes reading more robust, and saves disk space in some situations.

A 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 in a TIFF field
8+1*20	Big directory entry structure (see Table 4)	Second tag data in a TIFF field
…		
8+(B-1)*20	Big directory entry structure (see Table 4)	Last tag data in a TIFF field
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 (n) (it is not the total number of bytes but the number of values)
12	n * Tag data or uint64 offset	Tag data (if the size of n * Tag data is lower or equal to 64 bits) or offset to tag data (otherwise)
[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 *Number of values* (n). 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.

Some good practices inherited from TIFF but not specified in [TIFF Revision 6.0 Final] are:

- *Number of values* (n) cannot be zero.
- Offset to tag data in multiple tags cannot be shared (e.g. Tag A and Tag B that not point to the same offset).
- Offset to tag data cannot point to the same fragment of the file (e.g. Tag A points to an area of m bytes, Tag B cannot point to part of the same area).

The following table contains the *Tag* datatypes that are defined in BigTIFF, in addition to the ones defined in the [TIFF Revision 6.0 Final] (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

identifier

/per/core/per-core-long8

part

StripOffsets, StripByteCounts, TileOffsets, and TileByteCounts tags MAY have the datatype LONG8 in BigTIFF when necessary. LONG, and SHORT are still valid possibilitities.

part

Tags that point to other IFDs, such as the SubIFDs tag (see TIFF PageMaker version 6), MAY have the datatype IFD8. The datatypes IFD (and the not recommendable LONG) are still valid possibilities.

NOTE	For future versions it would be useful to add a new datatype "UTF8". This will allow for better metadata handling and internationalization.

Media Types for any data encoding(s)

IANA defines the image/tiff media type for TIFF (https://www.iana.org/assignments/media-types/image/tiff). This Standard suggests the use of the same media type for BigTIFF, since it is a superset of TIFF. As a result, the media type image/tiff is used for both Classic TIFF and BigTIFF files. In principle, clients do not need to negotiate with the servers between classic TIFF or BigTIFF. In fact, this kind of negotiation will enter in contradiction with the first recommendation in this Standards that states that servers and writers should only use BigTIFF for files bigger than 4 Gibibytes (2^{32}) or when using the data types introduced in BigTIFF. Old clients not supporting BigTIFF will only receive BigTIFF when Classic TIFF will not work for them anyway. Since a modern software is able to programmatically differentiate between a Classic TIFF and a BigTIFF, they can share the same media time for the BigTIFF.

This Standard is also suggesting that same file extension ('tif' or 'tiff') is used for both Classic TIF and BigTIFF, despite other previous proposals such as 'tf8' and 'btf' (discussed some time ago in the LibTiff mailing list).

Appendix 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

Conformance Class Core

identifier

https://www.opengis.net/spec/BigTIFF/1.0/conf/core

target

https://www.opengis.net/spec/BigTIFF/1.0/req/core

classification

Target Type:BigTIFF File

abstract-test

/conf/core/big-offsets

abstract-test

/conf/core/bifh

abstract-test

/conf/core/bifd

big-offset

identifier

/conf/core/big-offsets

target

/req/core/big-offsets

test-purpose

Verify that the section *big image file header* uses 8 byte offsets

test-method

Inspect that the BigTIFF begins with a 16 byte *big image file header* (that replaces the TIFF v6 8 byte *image file header*). In particular verify that the 2 first bytes correspond to the byte order expected content (0x4949 or 0x4D4D) and the second 2 bytes represent a 42 or a 43. If it is a 42, consider the file as classic TIFF instead of a BigTIFF.

test-method

Inspect that the *big image file header* offset points to an area the content corresponds with a *big image file directory* (BIFD) structure.

bifh

identifier

/conf/core/bifh

target

/req/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.

bifd

identifier

/conf/core/bifd

target

/req/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.

test-method

Inspect that a classic *image file directory* is not present in a TIFF file.

test-method

Inspect that a *big image file directory* follows the structure of *Big Directory Entries* presented in Table 3.

test-method

Inspect that *big directory entries* in a BIFD are sorted by *Tag code* (in the same way than classic TIFF does) .

test-method

Inspect that each *big directory entry* follows the structure presented in Table 4. This can be done by examining the *Tag codes* and the *data types* and of the TIFF fields and recognize at least some of them (comparing with tables of accepted values).

test-method

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 an offset written as the *value*.

test-method

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.

test-method

Inspect that the value of *Offset to next BIFD* in a BIFD does not point to a previous BIFD.

test-method

Inspect that the `type` is compatible with the expected value. Since TIFF allows for more than one valid type for some fields, inspect that all types specified as unsigned integer, the `type` is set to BYTE, SHORT, LONG or LONG8.

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

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

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

Offset	Datatype	Value
4	uint32	Number of values (n)
8	n * Tag data or uint32 offset	Tag data (if the size of n * Tag data is lower or equal to 32 bits) or offset to tag data (otherwise)

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.

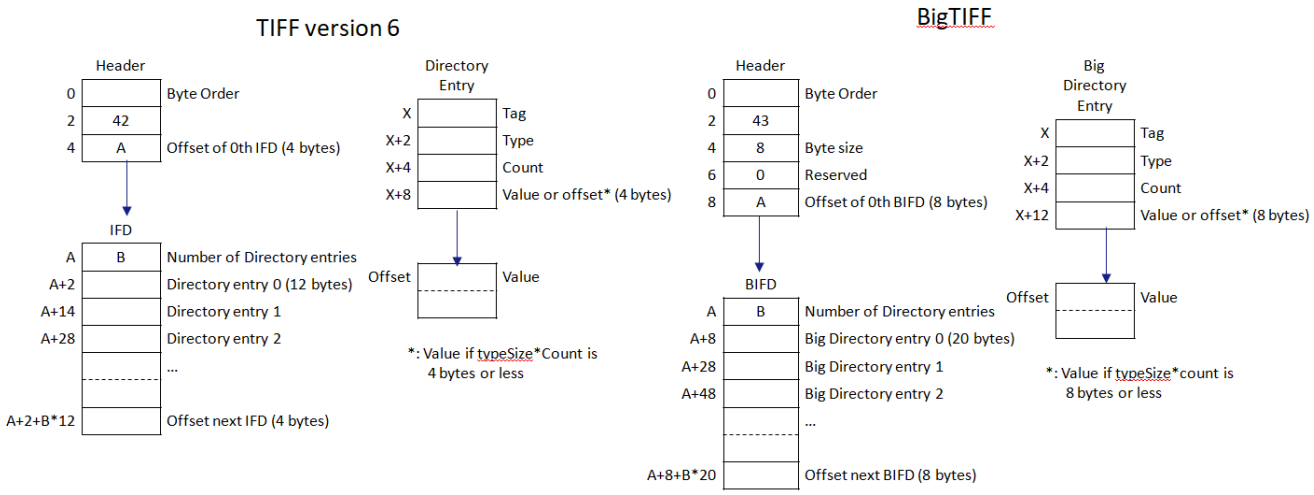


Figure 1. TIFF and BigTIFF data structures side by side.

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

Value	Name	Description	Source
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
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.

Appendix C: Revision History

Date	Release	Editor	Primary clauses modified	Description
2021-09-07	21-061 0.1	G. Editor	all	initial proposal from the OGC Testbed 17 work
2025-09-09	21-061 0.7	G. Editor	all	First full version
2026-03-25	21-061 0.8	G. Editor	all	Transformation to metanorma

Date	Release	Editor	Primary clauses modified	Description
2026-04-24	21-061r1 0.9	G. Editor	all	Version ready to be submitted to OGC for public comments. Reclassified as an OGC standard (instead of an OGC community standard)
2026-06-30	21-061r1 0.9	G. Editor	all	Version with all comments addressed ready for voting
2026-09-12	21-061r2 1.0	G. Editor	all	Final version submitted for its vote for approval by the OGC TC

Bibliography

- [BigTIFF Format], Joris Van Damme. The BigTIFF File Format. In Internet: https://web.archive.org/web/20240706160214/https://www.awaresystems.be/imaging/tiff/bigtiff.html
- [Welcome to BigTIFF], Welcome to the BigTIFF version of the libtiff library. In Internet: http://bigtiff.org/