**TIFF CHANGE INFORMATION**

**Current Version**: v4.0.0  
**Previous Version**: [v3.9.5](http://docs.google.com/v3.9.5.html)  
**Master FTP Site**:  [download.osgeo.org](ftp://download.osgeo.org/libtiff), directory pub/libtiff  
**Master HTTP Site**: <ftp://download.osgeo.org/libtiff>

This document describes the changes made to the software between the *previous* and *current* versions (see above). If you don't find something listed here, then it was not done in this timeframe, or it was not considered important enough to be mentioned. Please consult the ChangeLog file in the source package for full change details. The following information is located here:

* [Major Changes](#tyjcwt)
* [Changes in the software configuration](#30j0zll)
* [Changes in libtiff](#1fob9te)
* [Changes in the tools](#3znysh7)
* [Changes in the contrib area](#2et92p0)

**MAJOR CHANGES:**

BigTIFF support changes:

* The options parameter in the TIFFOpen and TIFFClientOpen funcs has been extended. When creating new files, you can add option '4' to specify you want to create a ClassicTIFF file, though that is the default and the option is not strictly necessary. (As such, old calling code will continue to function and create ClassicTIFF files.) Or you can add option '8' to specify you want to create a BigTIFF file instead. This new option is also reflected in some of the tools we already upgraded. For instance, you can use the -8 option on tiffcp to have tiffcp produce BigTIFF files instead of the default ClassicTIFF. (Whilst on additional option is provided for version selection when creating new files, no such option is necessary when reading TIFF files. LibTiff reads ClassicTIFF and BigTIFF both, and the application does not need to be aware which TIFF version an opened file is.)
* Although the tag count in BigTIFF is 64bit, we restricted the count in the implementation to a much more reasonable size. This is necessary in current implementation, because all tag data gets read automatically in the IFD reading stage, so if there's half a dozen private tags with multiple gigabytes of data that causes considerable overhead even if the application level is never interested in these tags. Our choice to ignore tags with data longer then a certain sanity value is much needed as things stand. We also recommend to step away from writing tiles that are 8 kilobyte in their uncompressed form, or writing single-line strips, in really big files, resulting in mega's of tiles or strips. It's much more efficient to choose bigger tile or strip sizes, up to several megabyte if needed, and have a few kilo of tiles or strips instead.
* Although it's rare, some application code does directly access file offsets. Some of these are automatically upgraded because they used the toff\_t type, others need to be aware that the datatype changed and need to start using toff\_t or uint64. This impacts access to tags like the EXIF IFD tag, for example, or the SubIfds tag, or to StripOffsets or TileOffsets, the return type of functions like TIFFCurrentDirOffset, and a parameter type to functions like TIFFSetSubDirectory.
* Although it's rare, some application code does use structures like TIFFHeader or TIFFDirEntry that used to be an exact binary representation of TIFF structures. These need to change. The old TIFFHeader structure is replaced by the new TIFFHeaderClassic, TIFFHeaderBig, and TIFFHeaderCommon structures that are an exact binary representation of the ClassicTIFF and BigTIFF header, and of the part that is common to both. There is no new equivalent for the old TIFFDirEntry structure (or more precisely, there is still a TIFFDirEntry structure, but it is changed, moved to library-private definition, and no longer an exact binary representation of the tag structure of either TIFF version).
* Sizer functions, like TIFFTileSize or TIFFScanlineSize and the like, return a tmsize\_t value (tmsize\_t is defined as int32 on 32bit machines, and int64 on 64bit machines, and as such it is meant to represent signed memory sizes). This is because we figure 98% of the calling code uses the return value as sizes in allocations and the like. So, any overflow that is theoretically possible with BigTIFF when LibTiff is running on a 32bit system, is best detected inside the sizer functions and it is best to return a type that makes sense as a memory size. If your calling code is the exception and is interested in actual file size, you best use the newer TIFFTileSize64 or TIFFScanlineSize64 function that returns an uint64 type.
* These TIFF tags require a 64-bit type as an argument in libtiff 4.0.0:
  + TIFFTAG\_FREEBYTECOUNTS
  + TIFFTAG\_FREEOFFSETS
  + TIFFTAG\_STRIPBYTECOUNTS
  + TIFFTAG\_STRIPOFFSETS
  + TIFFTAG\_TILEBYTECOUNTS
  + TIFFTAG\_TILEOFFSETS

Other important backward incompatible changes in the public API:

* TIFFRewriteField() renamed into \_TIFFRewriteField() and moved out from the public interface (from tiffio.h to tiffiop.h). Type of its 'count' parameter changed from uint32 to tmsize\_t.
* TIFFMergeFieldInfo() returns non-void result now. It returns 0 if successful and -1 if failed. Though this is now obsoleted function and should not be used in new programs. Use the new tag extension scheme instead.
* TIFFFieldWithTag() and TIFFFieldWithName() functions now return pointer to TIFFField constant object instead of TIFFFieldInfo.
* TIFFReassignTagToIgnore() function and TIFFIgnoreSense enumeration have been removed. They was unused and never been used properly. Should be unneeded for high-level applications.
* TIFFTagValue structure removed from the public tiffio.h to private tif\_dir.h and not accessible anymore. It should be unneeded for high-level applications.

**CHANGES IN THE SOFTWARE CONFIGURATION:**

* Updated autotools: Autoconf 2.68, Automake 1.11.1, libtool 2.4.
* Enabled support for Automake silent build rules (--enable-silent-rules or 'make V=0')
* Enabled support for Automake colorized and parallel tests.
* Added detection of 64-bit integer types since libtiff 4.0 requires use of 64-bit signed and unsigned integer types.
* Libtiff now provides a more comprehensive test suite with over 72 tests, which may be executed on Unix-like systems, or under Microsoft Windows using MinGW/MSYS or Cygwin.
* --disable-lzma configure option to disable use of liblzma.
* --enable-defer-strile-load configure option to enable experimental deferred strip/tile offset/size loading. May cause some extremely sophisticated uses of libtiff to fail.
* --enable-chunky-strip-read configure option to enable experimental enable reading large strips in chunks in TIFFReadScanline().
* Now always uses WIN32 native I/O functions for Microsoft Windows except for under Cygwin.
* Now provides a pkg-config support file (libtiff-4.pc).

**CHANGES IN LIBTIFF:**

* Patches/fixes made to stable libtiff (v3.9.X) are also applied to 4.0.0. There are too many to list here. See the distribution ChangeLog for a detailed change list.
* There is considerable change in some files like tif\_dirread and tif\_dirwrite. These changes don't impact backwards compatibility, they are mostly a clean rewrite that does allow BigTIFF support as well as somewhat more robust reading of the unexpected already and will also serve future API extension but does not impact current API or functionality in a negative way that you need to know about.
* Although there is still a functional definition for types like toff\_t (file offset), tstrip\_t (strip index number), etc, we recommend against using these in newer code. We have learned that it is next to impossible to use these consistently and make real abstraction of the binary format of these types. Instead, at a certain level we always end up doing casts anyway, and taking the exact binary format into account, so these types are nothing but dangerously misleading and obfuscating. You do not need to update calling code that uses them, as 99.9% of such code will continue to work. But we recommend against using them in newer calling code, and we started replacing them with binary clear types like uint16, uint32 and such in the library.
* We do use and will continue to use one functional type that is an exception to the above rule, being tmsize\_t. This is a signed memory size type, i.e. it is int32 on 32bit machines, or int64 on 64bit machines.
* Optionally support LZMA compression via TIFF tag 34925. Tiffcp supports compression levels similar to "-c lzma:p1" or "-c zip:p9 for setting the LZMA compression parameters.
* Optionally defer the load of strip/tile offset and size tags for optimized scanning of directories. Enabled with the --enable-defer-strile-load configure option (DEFER\_STRILE\_LOAD #define in tif\_config.h).
* Optionally enable experimental support for reading big strips in chunks. Enabled with the --enable-chunky-strip-read configure option.

**CHANGES IN THE TOOLS:**

* tiffset: add -d and -sd switches to allow operation on a particular directory, not just the first.

**CHANGES IN THE CONTRIB AREA:**

Last updated $Date: 2016-09-25 20:05:47 $.