GTiff -- GeoTIFF File Format

Most forms of TIFF and GeoTIFF files are supported by GDAL for reading, and somewhat less varieties can be written.

When built with internal libtiff or with libtiff >= 4.0, GDAL also supports reading and writing BigTIFF files (evolution of the TIFF format to support files larger than 4 GB).

Currently band types of Byte, UInt16, Int16, UInt32, Int32, Float32, Float64, CInt16, CInt32, CFloat32 and CFloat64 are supported for reading and writing. Paletted images will return palette information associated with the band. The compression formats listed below should be supported for reading as well.

As well, one bit files, and some other unusual formulations of GeoTIFF file, such as YCbCr color model files, are automatically translated into RGBA (red, green, blue, alpha) form, and treated as four eight bit bands.

Georeferencing

Most GeoTIFF projections should be supported, with the caveat that in order to translate uncommon Projected, and Geographic coordinate systems into OGC WKT it is necessary to have the EPSG .csv files available. They must be found at the location pointed to by the GDAL_DATA or GEOTIFF_CSV environment variable.

Georeferencing from GeoTIFF is supported in the form of one tiepoint and pixel size, a transformation matrix, or a list of GCPs.

If no georeferencing information is available in the TIFF file itself, GDAL will also check for, and use an ESRI world file with the extension .tfw, .tifw/.tiffw or .wld, as well as a MapInfo .tab file.

By default, information is fetched in following order (first listed is the most prioritary): PAM (Persitant Auxiliary metadata) .aux.xml sidecar file, INTERNAL (GeoTIFF keys and tags), TABFILE (.tab), WORLDFILE (.tfw, .tifw/.tiffw or .wld).

Starting with GDAL 2.2, the allowed sources and their priority order can be changed with the GDAL_GEOREF_SOURCES configuration option (or GEOREF_SOURCES open option) whose value is a comma-separated list of the following keywords : PAM, INTERNAL, TABFILE, WORLDFILE, NONE. First mentioned sources are the most prioritary over the next ones. A non mentioned source will be ignored.

For example setting it to "WORLDFILE,PAM,INTERNAL" will make a geotransformation matrix from a potential worldfile prioritary over PAM or GeoTIFF.

GDAL can read and write the *RPCCoefficientTag* as described in the RPCs in GeoTIFF proposed extension. The tag is written only for files created with the default profile GDALGeoTIFF. For other profiles, a .RPB file is created. In GDAL data model, the RPC coefficients are stored into the RPC metadata domain. For more details, see the RPC Georeferencing RFC. If .RPB or _RPC.TXT files are found, they will be used to read the RPCs, even if the *RPCCoefficientTag* tag is set.

Internal nodata masks

(from GDAL 1.6.0)

TIFF files can contain internal transparency masks. The GeoTIFF driver recognizes an internal directory as being a transparency mask when the FILETYPE_MASK bit value is set on the TIFFTAG_SUBFILETYPE tag. According to the TIFF specification, such internal transparency masks contain 1 sample of 1-bit data. Although the TIFF specification allows for higher resolutions for the internal transparency mask, the GeoTIFF driver only supports internal transparency masks of the same dimensions as the main image. Transparency masks of internal overviews are also supported.

When the GDAL_TIFF_INTERNAL_MASK configuration option is set to YES and the GeoTIFF file is opened in update mode, the CreateMaskBand() method on a TIFF dataset or rasterband will create an internal transparency mask. Otherwise, the default behaviour of nodata mask creation will be used, that is to say the creation of a .msk file, as per RFC 15

Starting with GDAL 1.8.0, 1-bit internal mask band are deflate compressed. When reading them back, to make conversion between mask band and alpha band easier, mask bands are exposed to the user as being promoted to full 8 bits (i.e. the value for unmasked pixels is 255) unless the GDAL_TIFF_INTERNAL_MASK_TO_8BIT configuration option is set to NO. This does not affect the way the mask band is written (it is always 1-bit).

Overviews

The GeoTIFF driver supports reading, creation and update of internal overviews. Internal overviews can be created on GeoTIFF files opened in update mode (with gdaladdo for instance). If the GeoTIFF file is opened as read only, the creation of overviews will be done in an external .ovr file. Overview are only updated on request with the BuildOverviews() method.

(From GDAL 1.6.0) If a GeoTIFF file has a transparency mask and the GDAL_TIFF_INTERNAL_MASK environment variable is set to YES and the GeoTIFF file is opened in update mode, BuildOverviews() will automatically create overviews for the internal transparency mask. These overviews will be refreshed by further calls to BuildOverviews() even if GDAL_TIFF_INTERNAL_MASK is not set to YES.

(From GDAL 1.8.0) The block size (tile width and height) used for overviews (internal or external) can be specified by setting the GDAL_TIFF_OVR_BLOCKSIZE environment variable to a power-of-two value between 64 and 4096. The default value is 128.

Metadata

GDAL can deal with the following baseline TIFF tags as dataset-level metadata :

- TIFFTAG_DOCUMENTNAME
- TIFFTAG_IMAGEDESCRIPTION
- TIFFTAG_SOFTWARE
- TIFFTAG_DATETIME
- TIFFTAG_ARTIST
- TIFFTAG_HOSTCOMPUTER
- TIFFTAG_COPYRIGHT
- TIFFTAG_XRESOLUTION
- TIFFTAG_YRESOLUTION
- TIFFTAG_RESOLUTIONUNIT
- TIFFTAG_MINSAMPLEVALUE (read only)
- TIFFTAG_MAXSAMPLEVALUE (read only)
- GEO_METADATA: This tag may be used for embedding XML-encoded instance documents prepared using 19139-based schema (GeoTIFF DGIWG) (GDAL >= 2.3)
- TIFF_RSID: This tag specifies a File Universal Unique Identifier, or RSID, according to DMF definition (GeoTIFF DGIWG) (GDAL >= 2.3)

The name of the metadata item to use is one of the above names ("TIFFTAG_DOCUMENTNAME", ...). On creation, those tags can for example be set with

```
gdal_translate in.tif out.tif -mo {TAGNAME}=VALUE
```

Other non standard metadata items can be stored in a TIFF file created with the profile GDALGeoTIFF (the default, see below in the Creation issues section). Those metadata items are grouped together into a XML string stored in the non standard TIFFTAG_GDAL_METADATA ASCII tag (code 42112). When BASELINE or GeoTIFF profile are used, those non standard metadata items are stored into a PAM .aux.xml file.

The value of GDALMD_AREA_OR_POINT ("AREA_OR_POINT") metadata item is stored in the GeoTIFF key RasterPixelIsPoint for GDALGeoTIFF or GeoTIFF profiles.

Starting with GDAL 1.9.0, XMP metadata can be extracted from the file, and will be stored as XML raw content in the xml:XMP metadata domain.

Starting with GDAL 1.10, EXIF metadata can be extracted from the file, and will be stored the EXIF metadata domain.

Color Profile Metadata

Starting with GDAL 1.11, GDAL can deal with the following color profile metadata in the COLOR_PROFILE domain:

- SOURCE_ICC_PROFILE (Base64 encoded ICC profile embedded in file. If available, other tags are ignored.)
- SOURCE_PRIMARIES_RED (xyY in "x,y,1" format for red primary.)
- SOURCE_PRIMARIES_GREEN (xyY in "x,y,1" format for green primary)
- SOURCE_PRIMARIES_BLUE (xyY in "x,y,1" format for blue primary)
- SOURCE_WHITEPOINT (xyY in "x,y,1" format for whitepoint)
- TIFFTAG_TRANSFERFUNCTION_RED (Red table of TIFFTAG_TRANSFERFUNCTION)
- TIFFTAG_TRANSFERFUNCTION_GREEN (Green table of TIFFTAG_TRANSFERFUNCTION)
- TIFFTAG_TRANSFERFUNCTION_BLUE (Blue table of TIFFTAG_TRANSFERFUNCTION)
- TIFFTAG_TRANSFERRANGE_BLACK (Min range of TIFFTAG_TRANSFERRANGE)

- TIFFTAG_TRANSFERRANGE_WHITE (Max range of TIFFTAG_TRANSFERRANGE)

Note that these metadata properties can only be used on the original raw pixel data. If automatic conversion to RGB has been done, the color profile information cannot be used.

All these metadata tags can be overridden and/or used as creation options.

Nodata value

GDAL stores band nodata value in the non standard TIFFTAG_GDAL_NODATA ASCII tag (code 42113) for files created with the default profile GDALGeoTIFF. Note that all bands must use the same nodata value. When BASELINE or GeoTIFF profile are used, the nodata value is stored into a PAM .aux.xml file.

Sparse files

GDAL makes a special interpretation of a TIFF tile or strip whose offset and byte count are set to 0, that is to say a tile or strip that has no corresponding allocated physical storage. On reading, such tiles or strips are considered to be implictly set to 0 or to the nodata value when it is defined. On writing, it is possible to enable generating such files through the Create() interface by setting the SPARSE_OK creation option to YES. Then, blocks that are never written through the IWriteBlock()/IRasterIO() interfaces will have their offset and byte count set to 0. This is particularly useful to save disk space and time when the file must be initialized empty before being passed to a further processing stage that will fill it. To avoid ambiguities with anoter sparse mechanism discussed in the next paragraphs, we will call such files with implicit tiles/strips "TIFF sparse files". They will be likely **not** interoperable with TIFF readers that are not GDAL based and would consider such files with implicit tiles/strips as defective.

Starting with GDAL 2.2, this mechanism is extended to the CreateCopy() and Open() interfaces (for update mode) as well. If the SPARSE_OK creation option (or the SPARSE_OK open option for Open()) is set to YES, even an attempt to write a all 0/nodata block will be detected so that the tile/strip is not allocated (if it was already allocated, then its content will be replaced by the 0/nodata content).

Starting with GDAL 2.2, in the case where SPARSE_OK is **not** defined (or set to its default value FALSE), for uncompressed files whose nodata value is not set, or set to 0, in Create() and CreateCopy() mode, the driver will delay the allocation of 0-blocks until file closing, so as to be able to write them at the very end of the file, and in a way compatible of the filesystem sparse file mechanisms (to be distinguished from the TIFF sparse file extension discussed earlier). That is that all the empty blocks will be seen as properly allocated from the TIFF point of view (corresponding strips/tiles will have valid offsets and byte counts), but will have no corresponding physical storage. Provided that the filesystem supports such sparse files, which is the case for most Linux popular filesystems (ext2/3/4, xfs, btfs, ...) or NTFS on Windows. If the file system does not support sparse files, physical storage will be allocated and filled with zeros.

Raw mode

For some TIFF formulations that have "odd" photometric color spaces, on-the-fly deconding as RGBA is done. This might not be desirable in some use cases. This behaviour can be disabled by prefixing the filename with GTIFF_RAW:

For example to translate a CMYK file to another one :

```
gdal_translate GTIFF_RAW:in.tif out.tif -co PHOTOMETRIC=CMYK
```

Open options

- **NUM_THREADS=number_of_threads/ALL_CPUS**: (From GDAL 2.1) Enable multi-threaded compression by specifying the number of worker threads. Worth it for slow compression algorithms such as DEFLATE or LZMA. Will be ignored for JPEG. Default is compression in the main thread.
- **GEOREF_SOURCES=string**: (GDAL > 2.2) Define which georeferencing sources are allowed and their priority order. See *Georeferencing* paragraph.
- **SPARSE_OK=TRUE/FALSE** ((GDAL > 2.2): Should empty blocks be omitted on disk? When this option is set, any attempt of writing a block whose all pixels are 0 or the nodata value will cause it not to be written at all (unless there is a corresponding block already allocated in the file). Sparse files have 0 tile/strip offsets for blocks never written and save space; however, most non-GDAL packages cannot read such files. The default is FALSE.

Creation Issues

GeoTIFF files can be created with any GDAL defined band type, including the complex types. Created files may have

any number of bands. Files of type Byte with exactly 3 bands will be given a photometric interpretation of RGB, files of type Byte with exactly four bands will have a photometric interpretation of RGBA, while all other combinations will have a photometric interpretation of MIN_IS_BLACK. Starting with GDAL 2.2, non-standard (regarding to the intrinsics TIFF capabilities) band color interpretation, such as BGR ordering, will be handled in creation and reading, by storing them in the GDAL internal metadata TIFF tag.

Note that the GeoTIFF format does not support parametric description of datums, so TOWGS84 parameters in coordinate systems are lost in GeoTIFF format.

You may want to read hints to generate and read cloud optimized GeoTIFF files

Creation Options

- **TFW=YES**: Force the generation of an associated ESRI world file (.tfw).See a World Files section for details.
- **RPB=YES**: Force the generation of an associated .RPB file to describe RPC (Rational Polynomial Coefficients), if RPC information is available. If not specified, this file is automatically generated if there's RPC information and that the PROFILE is not the default GDALGeoTIFF.
- **RPCTXT=YES**: (GDAL >=2.0) Force the generation of an associated _RPC.TXT file to describe RPC (Rational Polynomial Coefficients), if RPC information is available.
- **INTERLEAVE=[BAND,PIXEL]**: By default TIFF files with pixel interleaving (PLANARCONFIG_CONTIG in TIFF terminology) are created. These are slightly less efficient than BAND interleaving for some purposes, but some applications only support pixel interleaved TIFF files.
- **TILED=YES**: By default stripped TIFF files are created. This option can be used to force creation of tiled TIFF files.
- **BLOCKXSIZE=n**: Sets tile width, defaults to 256.
- **BLOCKYSIZE=n**: Set tile or strip height. Tile height defaults to 256, strip height defaults to a value such that one strip is 8K or less.
- **NBITS=n**: Create a file with less than 8 bits per sample by passing a value from 1 to 7. The apparent pixel type should be Byte. From GDAL 1.6.0, values of n=9...15 (UInt16 type) and n=17...31 (UInt32 type) are also accepted. From GDAL 2.2, n=16 is accepted for Float32 type to generate half-precision floating point values.
- **COMPRESS=[JPEG/LZW/PACKBITS/DEFLATE/CCITTRLE/CCITTFAX3/CCITTFAX4/LZMA/ZSTD/NONE]**: Set the compression to use. JPEG should generally only be used with Byte data (8 bit per channel). But starting with GDAL 1.7.0 and provided that GDAL is built with internal libtiff and libjpeg, it is possible to read and write TIFF files with 12bit JPEG compressed TIFF files (seen as UInt16 bands with NBITS=12). See the "8 and 12 bit JPEG in TIFF" wiki page for more details. The CCITT compression should only be used with 1bit (NBITS=1) data. LZW and DEFLATE compressions can be used with the PREDICTOR creation option. ZSTD is available since GDAL 2.3 when using internal libtiff and if GDAL built against libzstd >=1.0, or if built against external libtiff with zstd support. None is the default.
- **NUM_THREADS=number_of_threads/ALL_CPUS**: (From GDAL 2.1) Enable multi-threaded compression by specifying the number of worker threads. Worth for slow compressions such as DEFLATE or LZMA. Will be ignored for JPEG. Default is compression in the main thread.
- **PREDICTOR=[1/2/3]**: Set the predictor for LZW or DEFLATE compression. The default is 1 (no predictor), 2 is horizontal differencing and 3 is floating point prediction.
- **DISCARD_LSB=nbits or nbits_band1,nbits_band2,...nbits_bandN**: (GDAL >= 2.0) Set the number of least-significant bits to clear, possibly different per band. Lossy compression scheme to be best used with PREDICTOR=2 and LZW/DEFLATE compression.
- **SPARSE_OK=TRUE/FALSE** (From GDAL 1.6.0): Should newly created files (through Create() interface) be allowed to be sparse? Sparse files have 0 tile/strip offsets for blocks never written and save space; however, most non-GDAL packages cannot read such files. Starting with GDAL 2.2, SPARSE_OK=TRUE is also supported through the CreateCopy() interface. Starting with GDAL 2.2, even an attempt to write a block whose all pixels are 0 or the nodata value will cause it not to be written at all (unless there is a corresponding block already allocated in the file). The default is FALSE.
- **JPEG_QUALITY=[1-100]**: Set the JPEG quality when using JPEG compression. A value of 100 is best quality (least compression), and 1 is worst quality (best compression). The default is 75.
- **JPEGTABLESMODE=0/1/2/3**: (From GDAL 2.0) Configure how and where JPEG quantization and Huffman tables are written in the TIFF JpegTables tag and strip/tile. Default to 1.

  - 0: JpegTables is not written. Each strip/tile contains its own quantization tables and use optimized Huffman coding.
  - 1: JpegTables is written with only the quantization tables. Each strip/tile refers to those quantized tables and use optimized Huffman coding. This is generally the optimal choice for smallest file size, and consequently is the default.
  - 2: JpegTables is written with only the default Huffman tables. Each strip/tile refers to those Huffman tables (thus no optimized Huffman coding) and contains its own quantization tables (identical). This option has no anticipated practical value.
  - 3: JpegTables is written with the quantization and default Huffman tables. Each strip/tile refers to those tables (thus no optimized Huffman coding). This option could perhaps with some data be more efficient than 1, but this should only occur in rare circumstances.

- **ZLEVEL=[1-9]**: Set the level of compression when using DEFLATE compression. A value of 9 is best, and 1 is least compression. The default is 6.

- **ZSTD_LEVEL=[1-22]**: Set the level of compression when using ZSTD compression. A value of 22 is best (very slow), and 1 is least compression. The default is 9.

- **PHOTOMETRIC=[MINISBLACK/MINISWHITE/RGB/CMYK/YCBCR/CIELAB/ICCLAB/ITULAB]**: Set the photometric interpretation tag. Default is MINISBLACK, but if the input image has 3 or 4 bands of Byte type, then RGB will be selected. You can override default photometric using this option.

- **ALPHA=[YES/NON-PREMULTIPLIED/PREMULTIPLIED/UNSPECIFIED]**: The first "extrasample" is marked as being alpha if there are any extra samples. This is necessary if you want to produce a greyscale TIFF file with an alpha band (for instance). For GDAL < 1.10, only the YES value is supported, and it is then assumed as being PREMULTIPLIED alpha (ASSOCALPHA in TIFF). Starting with GDAL 1.10, YES is an alias for NON-PREMULTIPLIED alpha, and the other values can be used.

- **PROFILE=[GDALGeoTIFF/GeoTIFF/BASELINE]**: Control what non-baseline tags are emitted by GDAL.

  - With `GDALGeoTIFF` (the default) various GDAL custom tags may be written.
  - With `GeoTIFF` only GeoTIFF tags will be added to the baseline.
  - With `BASELINE` no GDAL or GeoTIFF tags will be written. BASELINE is occasionally useful when writing files to be read by applications intolerant of unrecognized tags.

- **BIGTIFF=YES/NO/IF_NEEDED/IF_SAFER**: Control whether the created file is a BigTIFF or a classic TIFF.

  - YES forces BigTIFF.
  - NO forces classic TIFF.
  - IF_NEEDED will only create a BigTIFF if it is clearly needed (in the uncompressed case, and image larger than 4GB. So no effect when using a compression).
  - IF_SAFER will create BigTIFF if the resulting file *might* exceed 4GB. Note: this is only a heuristics that might not always work depending on compression ratios.

  BigTIFF is a TIFF variant which can contain more than 4GiB of data (size of classic TIFF is limited by that value). This option is available if GDAL is built with libtiff library version 4.0 or higher. The default is IF_NEEDED.

  When creating a new GeoTIFF with no compression, GDAL computes in advance the size of the resulting file. If that computed file size is over 4GiB, GDAL will automatically decide to create a BigTIFF file. However, when compression is used, it is not possible in advance to known the final size of the file, so classical TIFF will be chosen. In that case, the user must explicitly require the creation of a BigTIFF with BIGTIFF=YES if the final file is anticipated to be too big for classical TIFF format. If BigTIFF creation is not explicitly asked or guessed and the resulting file is too big for classical TIFF, libtiff will fail with an error message like "TIFFAppendToStrip:Maximum TIFF file size exceeded".

- **PIXELTYPE=[DEFAULT/SIGNEDBYTE]**: By setting this to SIGNEDBYTE, a new Byte file can be forced to be written as signed byte.

- **COPY_SRC_OVERVIEWS=[YES/NO]**: (GDAL >= 1.8.0, CreateCopy() only) By setting this to YES (default is NO), the potential existing overviews of the source dataset will be copied to the target dataset without being recomputed. If overviews of mask band also exist, provided that the GDAL_TIFF_INTERNAL_MASK configuration option is set to YES, they will also be copied. Note that this creation option will have no effect if general options (i.e. options which are not creation options) of gdal_translate are used.

- **GEOTIFF_KEYS_FLAVOR=[STANDARD/ESRI_PE]**: (GDAL >= 2.1.0) Determine which "flavor" of GeoTIFF keys must be used to write the SRS information. The STANDARD way (default choice) will use the general accepted formulations of GeoTIFF keys, including extensions of the values accepted for ProjectedCSTypeGeoKey to new EPSG codes. The ESRI_PE flavor will write formulations that are (more) compatible of ArcGIS. At the time of writing, the ESRI_PE choice has mostly an effect when writing the EPSG:3857 (Web Mercator) SRS. For other SRS, the standard way will be used, with the addition of a ESRI_PE WKT string as the value of

PCSCitationGeoKey.

About JPEG compression of RGB images

When translating a RGB image to JPEG-In-TIFF, using PHOTOMETRIC=YCBCR can make the size of the image typically 2 to 3 times smaller than the default photometric value (RGB). When using PHOTOMETRIC=YCBCR, the INTERLEAVE option must be kept to its default value (PIXEL), otherwise libtiff will fail to compress the data.

Note also that the dimensions of the tiles or strips must be a multiple of 8 for PHOTOMETRIC=RGB or 16 for PHOTOMETRIC=YCBCR

Streaming operations

Starting with GDAL 2.0, the GeoTIFF driver can support reading or writing TIFF files (with some restrictions detailed below) in a streaming compatible way.

When reading a file from /vsistdin/, a named pipe (on Unix), or if forcing streamed reading by setting the TIFF_READ_STREAMING configuration option to YES, the GeoTIFF driver will assume that the TIFF Image File Directory (IFD) is at the beginning of the file, i.e. at offset 8 for a classical TIFF file or at offset 16 for a BigTIFF file. The values of the tags of array type must be contained at the beginning of file, after the end of the IFD and before the first image strip/tile. The reader must read the strips/tiles in the order they are written in the file. For a pixel interleaved file (PlanarConfiguration=Contig), the recommended order for a writer, and thus for a reader, is from top to bottom for a strip-organized file or from top to bottom, which a chunk of a block height, and left to right for a tile-organized file. For a band organized file (PlanarConfiguration=Separate), the above order is recommended with the content of the first band, then the content of the second band, etc... Technically this order corresponds to increasing offsets in the TileOffsets/StripOffsets tag. This is the order that the GDAL raster copy routine will assume.

If the order is not the one described above, the UNORDERED_BLOCKS=YES dataset metadata item will be set in the TIFF metadata domain. Each block offset can be determined by querying the "BLOCK_OFFSET_[xblock]_[yblock]" band metadata items in the TIFF metadata domain (where xblock, yblock is the coordinate of the block), and a reader could use that information to determine the appropriate reading order for image blocks.

The files that are streamed into the GeoTIFF driver may be compressed, even if the GeoTIFF driver cannot produce such files in streamable output mode (regular creation of TIFF files will produce such compatible files for streamed reading).

When writing a file to /vsistdout/, a named pipe (on Unix), or when definiting the STREAMABLE_OUTPUT=YES creation option, the CreateCopy() method of the GeoTIFF driver will generate a file with the above defined constraints (related to position of IFD and block order), and this is only supported for a uncompressed file. The Create() method also supports creating streamable compatible files, but the writer must be careful to set the projection, geotransform or metadata before writing image blocks (so that the IFD is written at the beginning of the file). And when writing image blocks, the order of blocks must be the one of the above paragraph, otherwise errors will be reported.

Some examples :

```
gdal_translate in.tif /vsistdout/ -co TILED=YES | gzip | gunzip | gdal_translate /vsistdin/ out.tif -co TILED=YES -co COMPRESS=DEFLATE
```

or

```
mkfifo my_fifo
gdalwarp in.tif my_fifo -t_srs EPSG:3857
gdal_translate my_fifo out.png -of PNG
```

Note: not all utilities are compatible with such input or output streaming operations, and even those which may deal with such files may not manage to deal with them in all circumstances, for example if the reading driver driven by the output file is not compatible with the block order of the streamed input.

Configuration options

This paragraph lists the configuration options that can be set to alter the default behaviour of the GTiff driver.

- GTIFF_IGNORE_READ_ERRORS : (GDAL >= 1.9.0) Can be set to TRUE to avoid turning libtiff errors into GDAL errors. Can help reading partially corrupted TIFF files
- ESRI_XML_PAM: Can be set to TRUE to force metadata in the xml:ESRI domain to be written to PAM.
- JPEG_QUALITY_OVERVIEW: Integer between 0 and 100. Default value : 75. Quality of JPEG compressed overviews, either internal or external.
- GDAL_TIFF_INTERNAL_MASK: See *Internal nodata masks* section. Default value : FALSE.
- GDAL_TIFF_INTERNAL_MASK_TO_8BIT: See *Internal nodata masks* section. Default value : TRUE
- USE_RRD: Can be set to TRUE to force external overviews in the RRD format. Default value : FALSE
- TIFF_USE_OVR: Can be set to TRUE to force external overviews in the GeoTIFF (.ovr) format. Default value : FALSE

- GTIFF_POINT_GEO_IGNORE: Can be set to TRUE to revert back to the behaviour of GDAL < 1.8.0 regarding how PixelIsPoint is interpreted w.r.t geotransform. See RFC 33: GTiff - Fixing PixelIsPoint Interpretation for more details. Default value : FALSE
- GTIFF_REPORT_COMPD_CS: (GDAL >= 1.9.0). Can be set to TRUE to avoid stripping the vertical CS of compound CS when reading the SRS of a file. Does not affect the writing side. Default value : FALSE
- GDAL_ENABLE_TIFF_SPLIT : Can be set to FALSE to avoid all-in-one-strip files being presented as having. Default value : TRUE
- GDAL_TIFF_OVR_BLOCKSIZE : See *Overviews* section.
- GTIFF_LINEAR_UNITS: Can be set to BROKEN to read GeoTIFF files that have false easting/northing improperly set in meters when it ought to be in coordinate system linear units. (Ticket #3901).
- TAB_APPROX_GEOTRANSFORM=YES/NO: (GDAL >= 2.0) To decide if an approximate geotransform is acceptable when reading a .tab file. Default value: NO
- GTIFF_DIRECT_IO=YES/NO: (GDAL >= 2.0) Can be set to YES to use specialized RasterIO() implementations when reading un-compressed TIFF files (un-tiled only in GDAL 2.0, both un-tiled and tiled in GDAL 2.1) to avoid using the block cache. Setting it to YES even when the optimized cases do not apply should be safe (generic implementation will be used). Default value:NO
- GTIFF_VIRTUAL_MEM_IO=YES/NO/IF_ENOUGH_RAM: (GDAL >= 2.0) Can be set to YES to use specialized RasterIO() implementations when reading un-compressed TIFF files to avoid using the block cache. This implementation relies on memory-mapped file I/O, and is currently only supported on Linux (64-bit build strongly recommended) or, starting with GDAL 2.1, on other POSIX-like systems. Setting it to YES even when the optimized cases do not apply should be safe (generic implementation will be used), but if the file exceeds RAM, disk swapping might occur if the whole file is read. Setting it to IF_ENOUGH_RAM will first check if the uncompressed file size is no bigger than the physical memory. Default value:NO. If both GTIFF_VIRTUAL_MEM_IO and GTIFF_DIRECT_IO are enabled, the former is used in priority, and if not possible, the later is tried.
- GDAL_GEOREF_SOURCES=comma-separated list with one or several of PAM, INTERNAL, TABFILE or WORLDFILE. (GDAL >= 2.2). See *Georeferencing* paragraph.
- GDAL_NUM_THREADS=number_of_threads/ALL_CPUS: (GDAL >= 2.1) Enable multi-threaded compression by specifying the number of worker threads. Worth it for slow compression algorithms such as DEFLATE or LZMA. Will be ignored for JPEG. Default is compression in the main thread. Note: this configuration option also apply to other parts to GDAL (warping, gridding, ...).

See Also:

- GeoTIFF Information Page
- libtiff Page
- Details on BigTIFF file format
- How to generate and read cloud optimized GeoTIFF files