MapTiler Manual Documentation

Release 11.2

MapTiler AG

Jun 17, 2021

Contents

1 Welcome **3**

2 Installation **5**
2.1 Windows . . . 5
2.2 macOS . . . 5
2.3 Linux . . . 5

3 Usage **7**
3.1 Getting started . . . 7
3.2 Output . . . 7
3.3 MapTiler Engine Command Structure . . . 8
3.4 Available output options . . . 8
3.5 The input files and related options . . . 13
3.6 Advanced options . . . 16
3.7 Bug report . . . 20
3.8 Vector inputs . . . 20

4 Hosting **23**
4.1 Standard web server . . . 23
4.2 Cloud hosting . . . 23

5 Demo version **27**
5.1 Setting CPU limits . . . 27
5.2 Demo trial extension . . . 27
5.3 Software activation online . . . 28
5.4 Software deactivation online . . . 28
5.5 License information . . . 29

6 Indices and tables **31**

Contents:

CHAPTER 1

Welcome

MapTiler Engine is a software for map tile rendering. It transforms supplied raster geodata from existing coordinate system (SRS - Spatial Reference System) into map tiles suitable for Google Maps API mashups, native mobile applications with Apple MapKit, open-source RouteMe library for iOS or OSMDroid for Google Android platform. Supported is a direct export to Google Earth products as well. The produced tiles can be used for online publishing according to the OpenGIS WMTS standard as well.

The produced map tiles can be easily served from existing in-house web servers, from practically any standard web-hosting provider, and from a public or private cloud. Hosting of maps is also possible from an external content distribution network (such as the Akamai's CDN with over 100.000 servers in 78 countries) to serve the geodata with higher speed and reliability by automatically caching it geographically closer to your online visitors.

MapTiler Engine can be used for processing a large quantity of input files with high-resolution. The tool has been designed for producing seamless maps and aerial photo layers covering whole countries. The rendering is fast and efficient, and it can fully utilize multiple CPUs to 100%. It is heavily optimized and directly working with the raw input data and computer memory mapping for producing the map tiles - without any intermediate steps and layers (such as HTTP requests / transcoding of the rasters / etc).

This and other internal tricks makes this tool a magnitude faster than any other existing solution.

Thank you for choosing MapTiler Engine.

CHAPTER 2

Installation

MapTiler Engine is part of MapTiler Desktop PRO, available as command line utility *maptiler*

2.1 Windows

Download the setup.exe (sent by an email) and start the installation wizard. After it is finished a new section is added to Start -> All Programs -> MapTiler Desktop Pro, and it is possible to use the command *maptiler* from the standard Command Prompt application in Windows.

2.2 macOS

Download the installation disk image (DMG, sent by an email) and after opening the disk image in macOS you should drag the "MapTiler Desktop Pro" icon into your "Application" folder. The command line interface can be started from Terminal application as: */Applications/MapTilerDesktopPro.app/Contents/MacOS/maptiler*.

All versions of macOS higher than 10.9 (Mavericks) are supported.

2.3 Linux

Download the installation package (.deb or .rpm) for your Linux distribution from the link sent by an email. The packages are compiled against the standard out-of-the-box GDAL installation available for your Linux distribution (typically it is the package gdal automatically available via apt or yum/dnf).

In case a different version of the GDAL library is required for your project, we can provide you with binaries compiled for such a version on request.

Supported Linux distributions are: Debian, Ubuntu, Redhat and Fedora (the rpms are dependent on GDAL from EPEL repository). Another option for running the latest MapTiler Engine with the latest GDAL on any Linux distribution is via the Docker container.

CHAPTER 3

Usage

3.1 Getting started

The main purpose of the software is to generate tiles in a defined tiling system, given some options and inputs. The only mandatory option is -o for the output directory. This directory must not exist yet when you run MapTiler Engine, to avoid overwriting existing data by mistake. As input, you can just provide the source dataset filename(s)[1].

```
maptiler -o output_directory input_file.ext
```

an example:

```
maptiler -o tiles map.tif
```

To render more files at once, just specify them one after another:

```
maptiler -o output_directory input1.tif input2.tif input3.tif
```

If you start the maptiler without arguments or with -help option, it will print all available commands:

```
maptiler -help
```

3.2 Output

The default behaviour of MapTiler Engine is to write tiles each into its own file, under the directory structure:

output_directory/z/x/y.ext

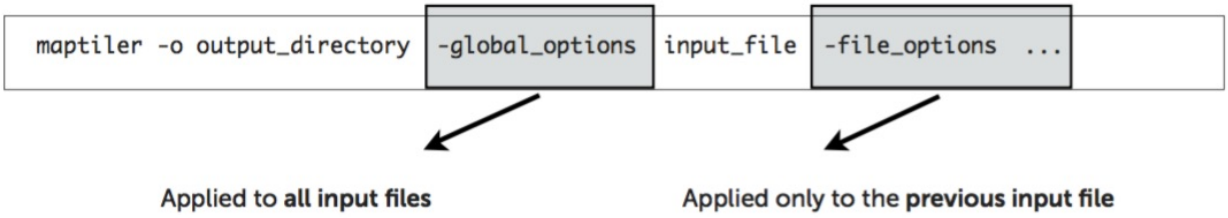
Where z is the zoom level, and x and y tile coordinates on relevant zoom level in the tiling profile.

The produced directory structure contains also a simple HTML viewer and description of the dataset in metadata.json, compatible with mb-util and TileServer project supporting OGC WMTS standard.

[1] Depending on your operating system you may need to call the command differently than just maptiler, typically on Linux and Mac in the actual directory as ./maptiler and on Windows as maptiler.exe.

MapTiler Engine supports direct output of the rendered map tiles into an SQLite database (MBTiles or GeoPackage format). This simplifies transfer and management of the tilesets and is practical for mobile applications.

3.3 MapTiler Engine Command Structure



The global options apply to all input files, in other words:

Only arguments specified BEFORE the input filenames are applied to all files!

Arguments which should be applied only to a single file are specified AFTER the name of such file (for example zoom level range specific only to that file) and has higher priority than the global options.

3.4 Available output options

3.4.1 Tiling profile / Tile Matrix Set

A global option defining the output system of tiles - the target coordinate system, tile pixel size, etc. MapTiler Engine comes with these predefined most popular systems and possibility to specify a custom profile.

-mercator DEFAULT. The spherical Mercator tile profile compatible with Google, Bing, Yahoo Maps, MapQuest, OpenStreetMap, and mobile maps on iOS and Android. This is the most commonly used profile. It uses coordinate system defined as EPSG:3857 or EPSG:900913. Details available here.

In case you wish to use different tiling system, you must specify it as the first command on the command line. These are the alternatives:

-gearth Tile profile specific for Google Earth according to the KML SuperOverlay definition.

-raster Rendering raster files without a need of georeferencing is also possible.

-garmin To produce the format for Garmin GPS devices, with a size 1024x1024 pixels and output to .kml file. Afterwards, pack the tiles and the .kml tiles into a .zip archive, change its extension to .kmz and save it into the device.

-custom You can specify your own tiling system. See the appropriate section under the Advanced options chapter for more information.

Example: command for producing tiles for use with Google Earth:

```
maptiler -gearth -o tiles map.tif
```

Example: command for producing tiles for use with Geodetic WGS84 Plate Caree:

```
maptiler -raster -o raster-tiles map.tif
```

3.4.2 Custom tiling presets

MapTiler Engine offers predefined custom tiling presets (custom tile grids) for Advanced customers. Each custom tiling preset has its own area coverage, some are for World, other covers only specifical States or group of States.

-preset geodetic WGS84 Plate Caree / Unprojected. Compatible with most existing WMS servers, OpenLayers base map.

Example:

```
maptiler -preset geodetic -o wgs84-tiles map.tif
```

-preset standard_grid Standard global tilegrid with a selected Coordinate system. This tiling preset keeps the original coordinate system (SRS), and cuts the input map into tiles according to the spherical mercator tile profile. **Note** this is not compatible with Google Mercator profile *-mercator*!

Example:

```
maptiler -preset standard_grid -o st-grid-tiles map.tif
```

-preset baidu Tilegrid defined for China customers. This preset cover only China region and is compatible with Baidu Maps service.

Example:

```
maptiler -preset baidu -o tiles map-china.tif
```

-preset yandex Custom tiling preset used in Russian web mapping service, compatible with Yandex.Maps. Coverage is limitted to Russia and Ukraine.

Example:

```
maptiler -preset yandex -o tiles map-russia.tif
```

-preset cz_jtsk National tiling grid for Czechia and Slovakia with a precision up to 1 meter per pixel.

Example:

```
maptiler -preset cz_jtsk -o cz-tiles map-czechia.tif
```

-preset fr_rgf93 Tilegrid defined for precise overlay of maps in France, using Lambert 93 conic projection.

Example:

```
maptiler -preset fr_rgf93 -o fr-tiles map-france.tif
```

-preset nl_rdnew National tiling grid for Netherlands - Rijksdriehoekstelsel New / Amersfoort.

Example:

```
maptiler -preset nl_rdnew -o netherlands-map map-amsterdam.tif
```

-preset uk_osgb [zoom_group] National tiling grid for the United Kingdom using Ordnance Survey projection. This custom preset requires a specific zoom_group, which limits output zoom levels of this grid. Supported values with zoom levels in the bracket are: *0* (z0), *1* (z1 - z2), *2* (z3 - z6), *3* (z7 - z8), *4* (z9 - z10).

Example:

```
maptiler -preset uk_osgb 1 -o gb-z1 map-london.tif -zoom 1 2
maptiler -preset uk_osgb 2 -o gb-z3 map-london.tif -zoom 3 6
```

-preset ch_lv03 [zoom_group] Swiss national tiling grid used in Switzerland and Liechtenstein with high precision. This custom preset requires a specific zoom group, with values from 0 to 21. These values are mostly represented for the specific zoom level. Output tiles could be combined and are compatible with SwissTopo maps.

Example:

```
maptiler -preset ch_lv03 3 -o ch-z3 map-zurich.tif -zoom 3 3
maptiler -preset ch_lv03 4 -o ch-z4 map-zurich.tif -zoom 4 4
```

Note that, *-zoom 3 3* is not required and it is automatically limited as defined for this zoom group.

-preset nz_nztm [zoom_group] New Zealand Geodetic Datum (NZGD2000), official geodetic datum for New Zealand and its offshore islands. This custom preset requires a specific zoom group, which limits output zoom levels of this grid. Supported values with zoom levels in the bracket are: *0* (z0-z7), *1* (z8-z10), *2* (z11-z13), *3* (z14-z16).

Example:

```
maptiler -preset nz_nztm 1 -o nz-z8 map-new-zealand.tif
```

3.4.3 Retina / HiDPI tiles

-scale [value] To create high-resolution Retina / HiDPI tiles with variable floating scale. Retina tiles are available for each profile and custom tiling presets listed above. Important note, the scale value cannot exceeded max allowed tile size in pixels: 4096 x 4096. It means, max available value for *-scale* is 16.0.

Example: the command for producing standard Retina tiles in Mercator profile

```
maptiler -mercator -scale 2.0 -o tiles@2x map.tif
```

Example: the command for producing Retina tiles at 1.5 scale in raster profile

```
maptiler -raster -scale 1.5 -o tiles-retina map.tif
```

3.4.4 Zoom levels

-zoom [min] [max] This option determines which layers of the tile pyramid will be generated. The default is the "native" level calculated from image resolution. In case you need to add additional zoom levels, you can either define them as absolute numeric values or as relative numbers to the "native" levels with prefix + and -.

Each input file can have its own explicit option for zoom levels.

Example: zoom levels are automatically calculated as eg. 1 - 5

```
maptiler -o tiles map.tif
```

Example: zoom levels are explicitly set to be 3 - 5

```
maptiler -o tiles map.tif -zoom 3 5
```

Example: zoom levels are set to be 1 - 6 with relative value to native zoom levels

```
maptiler -o tiles map.tif -zoom +0 +1
```

Example: zoom levels are set to be 2 - 4 with relative value to native zoom levels

```
maptiler -o tiles map.tif -zoom +1 -1
```

Example: zoom levels are set to 0 - 4, as explicit minimum, relative maximum to native zoom level

```
maptiler -o tiles map.tif -zoom 0 -1
```

3.4.5 Tile formats

The produced tiles can be saved in one of several image format. MapTiler Engine includes optimization of the final filesize and used a number of colors (quantization), to minimize the disk size occupied by the rendered maps as well as the time necessary to transfer the maps to clients once the tiles are online.

Formats with support for transparency are:

-f png8a DEFAULT. Paletted RGBA PNG image.

-f png or -f png32 RGBA PNG image

-f webp or -f webp32 RGBA WebP image

Non-transparent formats are:

-f jpg or -f jpeg Progressive JPEG image in the YCbCr color space

-f png8 Paletted RGB PNG image

-f png24 RGB PNG image

-f webp24 RGB WebP image

3.4.6 Tile transparency or a background color

No matter what input datasets you specify, after transforming them into the tiling profile projection, MapTiler Engine will handle them as RGBA images. The transparency can come from the image itself as an alpha channel (with support for partly transparent areas), it can be derived from a selected color (so-called NODATA color), or can be just a result of the transformation with the GDAL warping algorithm - for areas without available input data.

If the tile is completely transparent it is never saved to the disk to save the storage space.

If all of the pixels are fully visible (eg. opaque, maximum alpha is 255), the alpha channel is discarded and the tile is marked as non-transparent / opaque. Otherwise, the tile is marked as partly transparent with alpha.

If partly transparent tiles are saved in a tile format without support for transparency (such as JPEG specified with -f jpg option) then the background color is applied. Default background color is white (255,255,255), but you can specify your own with the option:

-bg [r] [g] [b] The color of the background replacing transparency in the non-transparent tile formats.

For example:

```
maptiler -f png8 -bg 0 128 0 ...
```

-ignore_alpha If your dataset contains four channels, but the fourth channel is not alpha channel, you can use this option to ignore this channel.

For example:

```
maptiler -f png32 -ignore_alpha input_4bands.tif ...
```

3.4.7 Tile store format

-store dir|mbtiles|geopackage This option enforces the form of storage which is used for saving the rendered tiles. Possible options are the directory (dir), the MBTiles (mbtiles) and the GeoPackage (geopackage). The default is the directory, but in case the -o parameter ends with .mbtiles or .gpkg then rendering into MBTiles or GeoPackage is selected, respectively. This option specifies the store form explicitly.

Note: for more details on this subject read the section Output in the chapter Usage above.

-sparse Skip the empty space between separate maps and don't create empty tiles. This option can improve the speed of rendering if there are huge areas between maps. This is the default option for *-store dir*.

-no_sparse Fills the empty space between separate maps (if there is some) with empty tiles in the background colour. This option can take longer to render and take more disk space, if there are huge areas between maps, as these have to be created. This is a default option for *-store mbtiles* and *-store geopackage*.

Setting the sparse option in GUI is in Advanced options dialog.

3.4.8 Hybrid tile format

MapTiler Engine allows rendering into a hybrid tile format which allows transparent tiles using transparent format (such as PNG) and tiles without any transparency at all are saved in a different format (such as JPEG). For aerial photos overlays or other datasets, this can mean a significant saving of the storage. Generated files are without extensions. This is done to simplify the generated OpenLayers viewer.

Example of usage:

```
maptiler -f hybrid <opaque> <transparent> ...
maptiler -f hybrid jpg png8a ...
```

3.4.9 Tile quality

There are some options to specify parameters of the conversion into image formats, which can significantly reduce the size of produced tiles by degrading the output.

-jpg_quality [value] The quality of JPEG compression. A number between 10 and 95. The default is 85.

-quant_quality [value] The quality of quantization. A number between 1 and 100. The default is 100.

-quant_speed [value] Higher speed levels disable expensive algorithms and reduce quantization precision. Speed 1 gives marginally better quality at significant CPU cost. Speed 10 has usually 5% lower quality but is 8 times faster than speed 8. The default is 10.

If you experience issues with the visual quality of generated tiles with quantization involved try to set -quant_speed to lower values.

-webp_quality [value] The quality of WebP compression. A number between 1 and 100. Level 100 means lossless compression. The default is 75.

-webp_alpha_quality [value] The quality of WebP alpha channel compression. A number between 1 and 100. Level 100 means lossless compression. The default is 100.

-webp_lossless Lossless WebP compression switch.

-webp_lossy Lossy WebP compression switch.

-webp_preset [default|picture|photo|drawing|icon|text] WebP compression presets that use optimal algorithm parameters for a given data type.

Example of the rendering of a seamless map out of file map1.tif and map2.tif into tiles with an internal palette with optimal colors with higher visual :

```
maptiler -o tiles -f png8a -quant_quality 90 -quant_speed 4 map1.tif map2.tif
```

3.4.10 Watermark

-watermark [image_file.png] It is possible to place your own watermark over rendered tiles to protect the online maps. The file should be smaller than a size of tiles. It is placed in a random position and burned into tiles.

A nice watermark file can be easily generated online by calling the Google Chart API: http://chart.apis.google.com/chart?chst=d_text_outline&chld=FFFFFF\T1\textbar{}11\T1\textbar{}h\T1\textbar{}000000\T1\textbar{}b\T1\textbar{}%C2%A9%20ABC

By replacing ABC at the end of this URL a custom text phrase can be specified. We recommend setting the transparency of such watermark file by using a Photoshop or similar tool before applying it with MapTiler Engine.

Example of usage of the watermark:

```
maptiler -o tiles -watermark watermark_image.png map.tif
```

3.5 The input files and related options

3.5.1 Supported input file formats

MapTiler Engine is able to open and process a large number of raster geodata formats, including: GeoTIFF, Erdas Imagine, ECW, MrSID, JPEG2000, SDTS, DTED, NITF, HDF4/5, BSB/KAP, OziExplorer, etc.

The complete list of supported formats is available online here.

3.5.2 Spatial reference system

Practically any modern existing georeferencing coordinate system (SRS - spatial reference system, e.g. geodetic datum + map projection with parameters) is supported, which means the software can process almost any geodata you may have available from all over the world.

In case the input files contain already the definition of a used coordinate system (SRS) then MapTiler Engine is able to load it and directly use this information for the transformation of the maps. In case this information is missing in the supplied file or it is incorrect (the maptiler place the maps on a wrong location, you can still assign the information about the spatial reference system with an option:

-srs [definition] Dataset projection. Can be WKT, EPSG code in the form of 'epsg:XXXX', PROJ.4 string. Beware of escaping. To search for identifiers or definitions use spatialreference.org or EPSG.io.

Example of assigning the United Kingdom spatial reference OSGB to a GeoTIFF file before rendering:

```
maptiler -o tiles -srs EPSG:27700 map_in_osgb.tif
```

3.5.3 Transparency from a color

-nodata [r] [g] [b] This command is typically used to eliminate borders of multiple map sheets that are stitched together. You can set a specific color of the map to be considered fully transparent during rendering.

Example for removing fully black border around a map:

```
maptiler -o tiles map.tif -nodata 0 0 0
```

3.5.4 Georeference / calibration

Georeferencing can also be done visually using GUI or online tool.

For proper rendering of the maps the location of supplied input files in the known coordinate system (SRS) must be available. MapTiler Engine is loading the geolocation automatically from the internal headers of the input files (such as GeoTIFF) or from external supportive files (such as ESRI WorldFile) if they are available.

To enforce a custom selected georeference information or loading from external files these options are available:

-bbox [minx] [miny] [maxx] [maxy] To manually set bounds of a file in the specified spatial reference system.

-geotransform [posX] [scaleX] [rotX] [posY] [rotY] [scaleY] To assign affine transformation directly. This option can be also used with its short name *-gt*.

-georeference [path_to_file] An option to load external georeference from World File, Tab File, OziExplorer Map File or .prj file.

-corners [east1] [north1] [east2] [north2] [east3] [north3] To assign affine transformation with 3 corner points: [0, 0], [width, 0], [width, height]. This option can be used with WGS84 Coordinate System (EPSG:4326) as arguments *lng1 lat1 lng2 lat2 lng3 lat3*, which will set up -srs EPSG:4326 for files without a specified Coordinate system.

The geolocation can be specified using three or more control points - GCP (Ground Control Point). Each GCP is defined by the position on the raster (pixel_x and pixel_y), which is associated with a georeferenced location (easting northing [elevation]). The last element (elevation) is mostly zero.

-gcp [x_pixel] [y_pixel] [easting] [northing] [elevation] To assign a ground control point. At least three control points are required. The last elemet *[elevation]* is optional value.

-order [value] An option to set the polynomial order for transformation method of assigned GCPs. Supported orders are 0 (auto), 1 (affine) and 2 (polynomial of second order). By default, the automatic order is selected based on a number of GCP points.

-tps Force the use of Thin Plate Spline transformer based on assigned GCP points. This option cannot be used with *-order*. This option is recommended for more than 10 assigned GCPs.

Example for using TPS transformation with assigned GCPs:

```
maptiler -o tiles map.tif -srs EPSG:26712 -tps -gcp 0 0 386638.171 3999090.834 -gcp␣
↪5400 0 399627.528 3999090.834 -gcp 5400 6800 399627.528 3982553.605
```

3.5.5 Cutline (Crop)

There are two command line options for cutline: *-cutline* and *-cutline_proj*. They specify the cutline (a clipping path) for an input image in pixels or in projected coordinates. They both expect a file name. The file can be either CSV or an OGR dataset (such as ESRI ShapeFile .shp).

From an OGR file, MapTiler Engine will load all polygons and multi-polygons from all features of the first layer.

The CSV format with pixel coordinates of nodes of a triangle, more lines will create polygon:

```
X1,Y1
X2,Y2
X3,Y3
```

-cutline [path] A pixel-based cutline is specific for each input file - so the parameter should be used after a filename (see section MapTiler Engine Command Structure).

Example of use of such a pixel-based cutline:

```
maptiler -o outputdir input.tif -cutline polygon.csv
```

-cutline_proj [path] A cutline with geocoordinates can be used for multiple files if it is specified before the first input file.

Another example of cutline with geocoordinates stored in a .shp file (may require accompanying .prj file with a coordinate system):

```
maptiler -o outputdir input.tif -cutline_proj shape.shp
```

-cutline IGNORE Ignore embedded cutline of the file.

Example:

```
maptiler -o outputdir input_with_cutline.tif -cutline IGNORE
```

3.5.6 Color correction

MapTiler Engine allows you to specify several parameters in order to improve the colors of the output map. The MapTiler Desktop Pro (GUI) is able to estimate these values interactively, but you can also use the following options to specify them manually.

-color_gamma [r] [g] [b] Specify gamma correction of the individual channels, higher values result in brighter pixels (1 = unchanged).

-color_contrast [contrast] [bias] Higher values of "contrast" result in bigger different between dark and light areas (0 = unchanged).

Use "bias" if you want to keep more details in the dark/light areas (0.5 = equal, <0.5 = details in light areas, >0.5 = details in dark areas)

-color_saturation [saturation] Modify saturation of the map (1 = unchanged, 0 = grayscale, >1 = colorful)

3.5.7 Multiple files into multiple MBTiles or Folders

MapTiler Engine is designed to produce a single merged layer from multiple input files. If you need to process multiple files and for each produce separate tileset then a batch processing is recommended.

Example:

This command processes every .tif file in a local directory and creates .mbtiles from each in the output directory. If .mbtiles is removed from the command, it produces separate directories instead. The command differs on operating systems:

Windows

```
for %f in (*.tif) do ( echo %f && maptiler -o output/%f.mbtiles %f )
```

When used in a batch file the %f must be %%f.

Linux / macOS

```
for %f in *tif; do echo $f; maptiler -o output/`basename $f .tif`.mbtiles $f; done;
```

3.6 Advanced options

3.6.1 Options in the optfile

In case you have a large number of arguments to pass to MapTiler Engine, such as many input files (total amount is unlimited for MapTiler Engine or MapTiler Desktop Pro), you can prepare a text file with all the arguments and call it with *–optfile myarguments.mtp*. List of files can be easily created with ls or dir commands.

–optfile [myarguments.mtp] Any arguments normally passed on the command line could be part of the *–optfile* text file. MapTiler Engine can combine arguments on the command line with arguments in the text file, such as:

```
maptiler -o output_directory --optfile myarguments.mtp
```

The *.mtp* extension is acronym for **MapTiler Project**, which can be used in MapTiler Desktop Pro GUI, see our tutorial page.

3.6.2 Temporary directory location

During rendering, MapTiler Engine also writes a substantial amount of data to a temporary directory. Not as much as will be in the output directory, but still. Please make sure there is enough space in the filesystem for it.

By default, the temporary directory will be created in the system default temporary location (*/tmp/* on Unix-like systems, or path from the environment variable%TEMP% on Windows-like systems). You can override this with the option:

-work_dir [directory] The location where to store temporary data during rendering. By default the system temporary directory.

Example:

```
maptiler -work_dir /tmp -o /mnt/data/tiles /mnt/maps/*.tif
```

3.6.3 Setting metadata for the output

It is possible to set certain metadata of the output generated with MapTiler Engine. The option is supported for all available output formats.

-name [string] Name of the generated map.

-description [string] Description of the generated map.

-legend [string] Legend of the generated map.

-attribution [string] Attribution of the generated map, contains author or data sources.

3.6.4 Resampling methods

The visual quality of the output tiles is also defined by the resampling method. Selected method is used for interpolation of the values of individual pixels and it affects the sharpness vs smoothness of the produced maps.

-resampling near Nearest neighbor resampling. Rarely makes sense for production data. Can be used for quick testing, since it is much faster than the others.

-resampling bilinear DEFAULT. Bilinear resampling (2x2 pixel kernel).

-resampling cubic Cubic convolution approximation (4x4 pixel kernel).

-resampling cubic_spline Cubic B-Spline Approximation (4x4 pixel kernel).

-resampling average Average resampling, computes the average of all non-NODATA contributing pixels. (GDAL >= 1.10.0)

-resampling mode Mode resampling, selects the value which appears most often of all the sampled points. (GDAL >= 1.10.0)

Resampling overviews produced by MapTiler Engine are using the average method, by default. Another possible method is Nearest neighbor.

-overviews_resampling near Nearest neighbor overviews resampling. Mostly used for elevation maps or similar.

-overviews_resampling average Average overviews resampling, computes the averate of all non-NODATA contributing pixels.

3.6.5 Defining a custom tiling profile for a specified coordinate system

MapTiler Engine allows defining a custom system of tiles which should be rendered. Such tiling scheme, or in the terminology of OGC WMTS service the TileMatrixSet is for the MapTiler Engine defined with parameters which must follow the tile profile option: *-custom*. This is **global option**, need to be specified BEFORE the input filenames

-tiling_srs [definition] The spatial reference system, e.g. the coordinate system in which the tiles are created. Follows the definitions known from -srs.

-tiling_bbox [minx] [miny] [maxx] [maxy] The area which should be split into tiles defined in the tiling_srs coordinates.

-tiling_resolution [zoomlevel] [resolution] Resolution in units of the tiling spatial reference system per pixel on the given zoom level. MapTiler Engine will automatically compute values for all other zoom levels, each having half the resolution of the previous one.

-tiling_resolution from_output Resolution is calculated so as to fit whole input mapset into one tile on zoom level 0 with respect to bbox, srs, and tile size.

-tiling_resolution from_input The default behavior if the resolution is not specified. Resolution is calculated so as to not supersample the largest input map with respect to bbox, srs and tile size.

-tile_size [width] [height] The pixel dimensions of one tile.

-tiling_centered Tile (0, 0) is in the center of the world.

3.6.6 Tiling scheme - naming of tiles

MapTiler Engine uses Google XYZ naming of tiles, by default. It supports also the OSGEO TMS naming (with flipped Y axis), QuadKey naming (known by Microsoft Bing Maps), and ZYX naming (known by Microsoft Bing Maps). These tiling schemes are supported only for tile store in the directory (*-store dir*). This is **global option**, need to be specified BEFORE the input filenames.

-xyz* or *-zxy Google XYZ (top-left origin) naming of tiles. Folder path as *output_directory/{z}/{x}/{y}.{ext}*.

-tms OSGEO TMS (bottom-left origin), flipped Y axis as oppose to Google XYZ. This tiling scheme is defined as a standard for MBTiles.

-quadkey Microsoft Bing QuadKey (top-left origin). MapTiler Engine generates files named as quadkey separated into directories named as zoom level (*output_directory/{z}/{quadkey}.{ext}*). Details at this microsoft website.

-zyx Microsoft Bing ZYX (top-left origin) naming of tiles. Folder path as *output_directory/{z}/{y}/{x}.{ext}*.

3.6.7 Choose bands from multiple channels for RGBA color model

MapTiler Engine allows to choose bands for RGB(A) color model from multiple map channels. The example is aerial images such as Sentinel 2 sources, which contains multiple spectral bands (channels) with different bandwidth, like Near Infra-Red, vegetation, cloud detection, etc. Only three classical bands are used for rendering via MapTiler Engine - RGB, Red Green and Blue bands, to construct True Color Images. This is **file options**, need to be specified AFTER the name of the input file.

-b [red] -b [green] -b [blue] -b [alpha] Select an input band for the processing color model RGBA. The last part *-b [alpha]* is optional to select Alpha channel. Bands are numbered from 1. This allows to reorder source bands.

Example for Sentinel 2 image, where RGB bands are 4th, 3rd and 2nd, respectively for Red Green Blue colors:

```
maptiler -o tiles sentinel2-image.multiband.tif -b 4 -b 3 -b 2
```

Example for generating red-only (1st band) image with alpha channel (4th band):

```
maptiler -o red-tiles image.tif -b 1 -b 1 -b 1 -b 4
```

-band_desc [string] Select an input band for the processing color model RGBA by the band description. The *[string]* parameter can be a substring of the full band description string. Usage is the same as for the *-b* command.

3.6.8 Define band data scale

It is possible to set the data scale for each of the color bands in the output generated with MapTiler Engine.

-data_scale [min] [max] Set data scale range for a band. The *min* parameter is optional. If omitted, the value will be set to 0. If the command is given only once, the same values will be set for all bands. To set different values for separate bands, the command has to be given 3 times (4 in case the alpha channel is used).

Example for setting all bands to range 0 - 400:

```
maptiler -o scaled-bands image.tif -data_scale 400
```

Example for setting different values for RGBA bands:

```
maptiler -o scaled-bands image.tif -data_scale 100 -data_scale 200 -data_scale 300 -
↪data_scale 400
```

3.6.9 Interrupt and resume long-time rendering

The long-time rendering job can be interrupted by the end-user or a system failure (power-failure, no free space on the disk). MapTiler Engine supports only simple resume mode - render process can be continued on the same computer with the same options.

-keep_unfinished To prevent deleting the existing output tiles and temporary files created by the application.

-resume To continue in the unfinished or interrupted rendering process. Requires the same arguments on the same computer. It skips encoding of the existing tiles. This option can be used also for the startup of the rendering process, it will automatically keep unfinished tiles.

3.6.10 Advanced warping arguments

The advanced warping algorithms parameters can be specified with the option:

-wo "NAME=VALUE" The warp options. See the papszWarpOptions field at GDAL.

Example:

```
maptiler -o tiles -wo "SAMPLE_GRID=YES" t.tif -wo "SOURCE_EXTRA=16"
```

3.6.11 Watch progress in a frontend

MapTiler Engine can produce progress easily parsed in a frontend application. Simply use the first argument *-progress* and application output the progress on the standard output in the TSV (tabulator separated values) format: Stage TAB Percentage TAB Iteration TAB Total

Example:

```
maptiler -progress -o tiles map1.tif map2.tif map3.tif

Opening    16 %    1    6
Opening    33 %    2    6
Opening    50 %    3    6
Opening    66 %    4    6
Opening    83 %    5    6
Opening   100 %    6    6
Warping     0 %    0    4
Warping    25 %    1    4
Warping    50 %    2    4
Warping    75 %    3    4
Warping   100 %    4    4
Rendering   0 %    0    512
...
Rendering   100 %    512    512
```

3.6.12 Usage on a computer cluster

MapTiler Engine can run on an MPI cluster if a cluster-specific binary has been requested. If you have the MPI version, a shell wrapper to run it on a cluster is delivered as well.

A version of MapTiler Engine utilizing Map Reduce approach and Hadoop is under development, this will replace the older MPI.

More details are available on MapTiler Cluster page.

3.6.13 Merge MBTiles utility

This feature is available in MapTiler Desktop PRO and MapTiler Engine editions with an activated license only, not in MapTiler Desktop PRO Demo. Merging MBTiles in GUI is described in this tutorial article.

The utility allows to update a previously rendered dataset and replace a small existing area with a different newly rendered raster data. The typical use-case is fixing of a small geographic area in a large seamed dataset previously rendered by MapTiler Engine from many input files.

The utility also extend the bounding box of the tiles - it is usable for merging two just partly overlapping maps into one bigger map covering larger extent.

Usage:

```
merge_mbtiles [OPTION] BASE.mbtiles DETAIL.mbtiles [DETAIL_2.mbtiles]...
```

Typical usage:

1) render a large dataset with MapTiler Engine - from several input files and produce large MBTiles (with JPEG or PNG tiles internally): *large.mbtiles*
2) if you want to update one of the previously rendered input files in the existing dataset render just this file into MBTiles - with the PNG32 format and zoom-levels on which you want it to appear in the large dataset. Save the new small MBTiles with just one file to *patch.mbtiles*

Example:

```
merge_mbtiles large.mbtiles patch.mbtiles
```

Existing tiles available in both *large.mbtiles* and the *patch.mbtiles* are going to be merged. On same zoom levels, patch.mbtiles will replace the original large.mbtiles - so the *large.mbtiles* will be updated in-place.

Further options:

-P [n] Set limit on the defined number of cores.

-no_sparse Fills the empty space between separate maps (if there is some) with empty tiles in a background color. This option can take longer to render, if there are huge areas between maps, as these have to be created. In case the maps overlap each other, there is no extra action involved. Default behavior without this option does not fill the empty space between separate maps.

-reencode This option is useful when the 2 merged maps have a different format (e.g. jpeg and png). By default, the result is a hybrid format (combination of both of them). If reencode option is used, the chosen file is encoded to the actual format (which can slow down the process).

3.7 Bug report

-report The argument *-report* generates the text report, which should be sent via the web form.

Attaching this file if you are reporting a bug is very important becuase this information helps us to identify the problem and quickly come up with a solution

Sending a bug report from MapTiler Desktop GUI application is described here.

3.8 Vector inputs

MapTiler Engine v10.0 and higher version supports rendering of Vector inputs into MVT (Mapbox Vector tile) format. Vector rendering support requires an underlying GDAL library version 2.3.0 or higher, which is limited on the native Linux distribution. Using a docker image with MapTiler Engine is recommended way for vector rendering on Linux OS. MapTiler Desktop offers GUI for Vector layers with a practical sample how-to article.

Vector input consist of one or more layers, which are rendered into the specific target layer in MVT format. Each feature of the source layer contains *key=value* attributes, that could be processed or renamed into the final attributes of the target layer. The following arguments are supported for the Vector input: *-srs*, *-zoom*, and *-bbox*, as they are described above. Other arguments are not respected for Vector rendering yet.

-layer [src_name] Select the **source layer** for the further processing of the vector input by the name. This argument is required for the arguments below.

-target [name] Select (or create a new) **target layer** in the final tiles of MVT format. This is the name of the layer, which could be styled. This argument may be repeated more times to process features into a separate target layer with a different list of fields.

-field [output_name] [src_name] Set the attribute field with the name **src_name** from the **source_layer** to be presented in the final **target layer** as a attribute key **output_name**. The attribute value for each features from **source layer** is copied. This argument may be repeated more times to copy more attributes.

-vector_tile_size [size] Set the output vector tile size in pixels.

-vector_compress Use compression of the output vector tiles.

-vector_no_compress Do not use compression of the output vector tiles.

Let assume, we have one Vector input with two **source layers**: *lines* and *polygons*. The source layer *lines* consists of the streets with these attributes keys *Name*, *Identifier* and *Main*. The source layer *polygons* contains geometry of some buildings with attributes keys *Ident*, *Num* and *Name*. We do want to create two **target layers** with renamed attributes.

Example

```
maptiler -o mymap.mbtiles \
input.shp \
-layer lines \
  -target streets \
  -field id Identifier \
  -field name Name \
  -field is_main Main \
-layer polygons \
  -target buildings \
  -field id Ident \
  -field name Name \
  -field number Num
```

The example above creates two new output layers: **streets** with attributes keys *id*, *name*, and *is_main*; and **buildings** layer with attributes keys *id*, *name*, and *number*.

CHAPTER 4

Hosting

4.1 Standard web server

On a standard hosting (such as an ordinary company web server) you can very simply host your maps. Just load the directory with tiles to your web hosting and the layer is automatically available.

Once uploaded, the produced maps can be opened in any viewer supporting OGC WMTS standard.

For hosting of MBTiles, you can use an open-source TileServer, that can be used with any standard hosting that supports PHP. Upload the created maps and get dozens of popular web viewers with interactivity, including Google Maps API, Leaflet, OpenLayers, WebGL Earth and MapBox JS. There is a standalone how-to describing the whole process of hosting with TileServer.

4.2 Cloud hosting

The CloudPush command can be used for upload to Amazon S3, Google Cloud Storage or Microsoft Azure Blob hosting. Examples are shown on the S3. If you need to use Google Cloud Storage or Microsoft Azure Blob, just change the "s3" to "gs" or "az", respectively. Full how-to with visual examples is available as a how-to article.

Cloud Push instance is initialized with the first uploaded map via this command line utility. It automatically creates an empty *index.json*, TileServer in *index.html* and sets WebSite configuration for this bucket. To get the required credentials, see the section Access credentials below.

Upload tiles from an MBTiles file to S3

```
cloudpush --access_key ACCESS_KEY --secret_key SECRET_KEY s3://bucket_name add␣
↪filename.mbtiles
```

List all maps in the cloudpush tile storage

```
cloudpush --access_key ACCESS_KEY --secret_key SECRET_KEY s3://bucket_name list
```

Delete a map

```
cloudpush --access_key ACCESS_KEY --secret_key SECRET_KEY s3://bucket_name delete␣
↪filename
```

Delete whole cloudpush storage

```
cloudpush --access_key ACCESS_KEY --secret_key SECRET_KEY s3://bucket_name destroy
```

4.2.1 Access credentials

The Amazon access and the secure key are available via IAM service administration interface. The credentials for the Google Cloud Storage are under "Enable interoperable access" in the menu of the service. The Azure Blob Storage requires the **Storage account name** as Access Key and the **Key** from the Microsoft Azure Portal - Storage Accounts - Access keys.

Instead of providing the access credentials in every command these can be set as system environment variables.

Example on Windows OS:

```
REM for Amazon S3
set AWS_ACCESS_KEY_ID=[THE_ACCESS_KEY]
set AWS_SECRET_ACCESS_KEY=[THE_SECRET_KEY]
REM or for Google Cloud Storage
set GOOG_ACCESS_KEY_ID=[THE_ACCESS_KEY]
set GOOG_SECRET_ACCESS_KEY=[THE_SECRET_KEY]
REM or for Microsoft Azure Storage
set AZURE_STORAGE_ACCOUNT=[ACCOUNT NAME]
set AZURE_STORAGE_ACCESS_KEY=[KEY]
```

Example on Linux / macOS:

```
# for Amazon S3
export AWS_ACCESS_KEY_ID=[THE_ACCESS_KEY]
export AWS_SECRET_ACCESS_KEY=[THE_SECRET_KEY]
# or for Google Cloud Storage
export GOOG_ACCESS_KEY_ID=[THE_ACCESS_KEY]
export GOOG_SECRET_ACCESS_KEY=[THE_SECRET_KEY]
# or for Microsoft Azure Storage
export AZURE_STORAGE_ACCOUNT=[ACCOUNT NAME]
export AZURE_STORAGE_ACCESS_KEY=[KEY]
```

and call the utility without these arguments:

```
cloudpush s3://bucket_name list
cloudpush s3://bucket_name add filename.mbtiles
cloudpush gs://bucket_name list
cloudpush az://bucket_name list
```

4.2.2 Advanced options

It is possible to use further options such as:

--create-bucket	automatically creates bucket, if not existing
--no-index-json	not handling metadata in CloudPush instance index.json
--raw	same as –no-index-json

--basename <path>	sets custom basename (default: basename of MBTiles file)
--private	uploaded objects are private (default: public)
--emulator	Enable Azure Storage Emulator API

List of available parameters can be displayed by running ./cloudpush without any parameter

Example for using custom basename:

```
cloudpush --basename myfile s3://bucket_name add filename.mbtiles
```

uploads tiles with URL format: *myfile/z/x/y.ext*. Custom basename contains directory separators (slash), for example:

```
cloudpush --basename year/month/myfile s3://bucket_name add filename.mbtiles
```

result will have URL in format: *year/month/myfile/z/x/y.ext*.

Region-specific hosting can be set up via environment variable AWS_BUCKET_REGION=[value] or with parameter -R [value].

Example for EU (Ireland) region:

```
cloudpush -R eu-west-1 s3://bucket_name add filename.mbtiles
```

The list of S3 regions is provided by the utility with *–more-help* argument or visible at https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region

To enable uploading tiles into Azure Storage Emulator, you need to pass the parameter *–emulator* for each command:

Example for emulator, does not require credentials:

```
cloudpush --emulator az://bucket_name add filename.mbtiles
```

The Azure Storage uses the API of the version 2015-02-21.

CHAPTER 5

Demo version

5.1 Setting CPU limits

MapTiler Engine is a multi-threaded program. By default, it will use all the CPUs you have and print the information about the cores. You can also set a limit on the defined number of cores yourself with the *-P* option.

-P [num] Set number of CPU cores to be used for render process.

Example:

```
maptiler -P 4 ...
```

This is especially practical to evaluate the demo application before purchasing a license for a particular number of cores.

The modern CPUs has multiple cores and support of Hyperthreading - which provides multiple logical CPUs per core. This way MapTiler Engine can provide higher performance with *-P* 4 even on a dual-core computer.

5.2 Demo trial extension

The demo version of MapTiler Engine is fully usable for 30 days after the first start. In case this trial period for testing the software before purchase expires and you would like to continue to test the demo, it is necessary to contact Klokan Technologies GmbH and request a trial extension key.

Such key can be then used with the parameter:

-extend_trial [TRIAL-KEY] Extends MapTiler Engine demo trial period.

And the demo can be then used during an extended time period.

5.3 Software activation online

After a purchase of the software, when you receive your license key, it is necessary to activate MapTiler Engine. After activation, the demo version no longer enforces the "MapTiler DEMO" overlays.

To do the online activation, use the following command:

-activate [YOUR-OWN-LICENSE-KEY] Activates MapTiler Engine with proper license key. You can get your License key online here.

Example:

```
maptiler -activate YOUR-OWN-LICENSE-KEY
```

In case you require to reinstall the computer, use -deactivate **command to be able to re-activate the license later on.**

The software activation for MapTiler Desktop Pro via GUI is described here.

5.3.1 Software activation in a virtual machine

The activation process for MapTiler Engine running in a virtual machine requires online activation only via environment variable *MAPTILER_LICENSE*. The software will be automatically deactivated in the end.

Example on Windows OS

```
set MAPTILER_LICENSE=YOUR-OWN-LICENSE-KEY
maptiler ...
```

Example on Linux / macOS

```
export MAPTILER_LICENSE=YOUR-OWN-LICENSE-KEY
maptiler ...
```

Notice: This activation process via environment variable requires **not-activated MapTiler Desktop Pro** (or **not-activated MapTiler Engine**) instance on that computer! Starting MapTiler Desktop application without setting this environment variable *MAPTILER_LICENSE*, you should see either **MapTiler Desktop Pro Demo**, or **Your trial period has expired** error message.

5.4 Software deactivation online

You can deactivate MapTiler Engine via command line, in order to transfer the application to another computer. Note that after the deactivation, MapTiler Engine will continue to run in DEMO mode, but the trial period of 30 days will **NOT** start again.

To do the online deactivation, use the following command:

-deactivate Deactivates activated MapTiler Engine software. MapTiler Engine can be used in Demo mode, if trial period has not expired.

Example:

```
maptiler -deactivate
```

You can deactivate MapTiler Desktop Pro in GUI via License dialog described here.

5.5 License information

To check your license information, use the following command:

-license Print your license information for activated MapTiler Engine.

Example

```
> maptiler -license
Email: <your email>
License: <your license key>
Purchased CPU cores: 4
Maximum usable cores: 4


maptiler -license
Remaining 0 days on the trial.
Maximum usable cores: 4
```

This argument may print additional texts, depending on your License, for example:

```
Time limited license!
X days remaining.

X days remaining in your maintenance period.

Your maintenance period has expired. You can't upgrade to newer versions - please␣
↪contact support@maptiler.com.
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search