TMX Map Format

The TMX (Tile Map XML) map format used by [Tiled](http://www.mapeditor.org/) is a flexible way to describe a tile based map. It can describe maps with any tile size, any amount of layers, any number of tile sets and it allows custom properties to be set on most elements. Beside tile layers, it can also contain groups of objects that can be placed freely.

Note that there are many [libraries and frameworks](http://doc.mapeditor.org/reference/support-for-tmx-maps/) available that can work with TMX maps.

In this document we'll go through each element found in this map format. The elements are mentioned in the headers and the list of attributes of the elements are listed right below, followed by a short explanation. Attributes or elements that are deprecated or unsupported by the current version of Tiled are formatted in italics.

Have a look at the [changelog](http://doc.mapeditor.org/reference/tmx-changelog/) when you're interested in what changed between Tiled versions.

*A DTD-file (Document Type Definition) is served at* [*http://mapeditor.org/dtd/1.0/map.dtd*](http://mapeditor.org/dtd/1.0/map.dtd)*. This file is not up-to-date but might be useful for XML-namespacing anyway.*

<map>

* **version:** The TMX format version, generally 1.0.
* **orientation:** Map orientation. Tiled supports "orthogonal", "isometric", "staggered" (since 0.9) and "hexagonal" (since 0.11).
* **renderorder:** The order in which tiles on tile layers are rendered. Valid values are right-down (the default), right-up, left-down and left-up. In all cases, the map is drawn row-by-row. (since 0.10, but only supported for orthogonal maps at the moment)
* **width:** The map width in tiles.
* **height:** The map height in tiles.
* **tilewidth:** The width of a tile.
* **tileheight:** The height of a tile.
* **hexsidelength:** Only for hexagonal maps. Determines the width or height (depending on the staggered axis) of the tile's edge, in pixels.
* **staggeraxis:** For staggered and hexagonal maps, determines which axis ("x" or "y") is staggered. (since 0.11)
* **staggerindex:** For staggered and hexagonal maps, determines whether the "even" or "odd" indexes along the staggered axis are shifted. (since 0.11)
* **backgroundcolor:** The background color of the map. (since 0.9, optional, may include alpha value since 0.15 in the form #AARRGGBB)
* **nextobjectid:** Stores the next available ID for new objects. This number is stored to prevent reuse of the same ID after objects have been removed. (since 0.11)

The tilewidth and tileheight properties determine the general grid size of the map. The individual tiles may have different sizes. Larger tiles will extend at the top and right (anchored to the bottom left).

A map contains three different kinds of layers. Tile layers were once the only type, and are simply called layer, object layers have the objectgroup tag and image layers use the imagelayer tag. The order in which these layers appear is the order in which the layers are rendered by Tiled.

Can contain: [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties), [tileset](http://doc.mapeditor.org/reference/tmx-map-format/#tileset), [layer](http://doc.mapeditor.org/reference/tmx-map-format/#layer), [objectgroup](http://doc.mapeditor.org/reference/tmx-map-format/#objectgroup), [imagelayer](http://doc.mapeditor.org/reference/tmx-map-format/#imagelayer)

<tileset>

* **firstgid:** The first global tile ID of this tileset (this global ID maps to the first tile in this tileset).
* **source:** If this tileset is stored in an external TSX (Tile Set XML) file, this attribute refers to that file. That TSX file has the same structure as the <tileset> element described here. (There is the **firstgid** attribute missing and this **source** attribute is also not there. These two attributes are kept in the TMX map, since they are map specific.)
* **name:** The name of this tileset.
* **tilewidth:** The (maximum) width of the tiles in this tileset.
* **tileheight:** The (maximum) height of the tiles in this tileset.
* **spacing:** The spacing in pixels between the tiles in this tileset (applies to the tileset image).
* **margin:** The margin around the tiles in this tileset (applies to the tileset image).
* **tilecount:** The number of tiles in this tileset (since 0.13)
* **columns:** The number of tile columns in the tileset. For image collection tilesets it is editable and is used when displaying the tileset. (since 0.15)

If there are multiple <tileset> elements, they are in ascending order of their firstgid attribute. The first tileset always has a firstgid value of 1. Since Tiled 0.15, image collection tilesets do not necessarily number their tiles consecutively since gaps can occur when removing tiles.

Can contain: [tileoffset](http://doc.mapeditor.org/reference/tmx-map-format/#tileoffset) (since 0.8), [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties) (since 0.8), [image](http://doc.mapeditor.org/reference/tmx-map-format/#image), [terraintypes](http://doc.mapeditor.org/reference/tmx-map-format/#terraintypes) (since 0.9), [tile](http://doc.mapeditor.org/reference/tmx-map-format/#tile)

<tileoffset>

* **x:** Horizontal offset in pixels
* **y:** Vertical offset in pixels (positive is down)

This element is used to specify an offset in pixels, to be applied when drawing a tile from the related tileset. When not present, no offset is applied.

<image>

* **format:** Used for embedded images, in combination with a data child element. Valid values are file extensions like png, gif, jpg, bmp, etc. (since 0.9)
* *id:* Used by some versions of Tiled Java. Deprecated and unsupported by Tiled Qt.
* **source:** The reference to the tileset image file (Tiled supports most common image formats).
* **trans:** Defines a specific color that is treated as transparent (example value: "#FF00FF" for magenta). Up until Tiled 0.12, this value is written out without a # but this is planned to change.
* **width:** The image width in pixels (optional, used for tile index correction when the image changes)
* **height:** The image height in pixels (optional)

Can contain: [data](http://doc.mapeditor.org/reference/tmx-map-format/#data) (since 0.9)

<terraintypes>

This element defines an array of terrain types, which can be referenced from the terrain attribute of the tile element.

Can contain: [terrain](http://doc.mapeditor.org/reference/tmx-map-format/#terrain)

<terrain>

* **name:** The name of the terrain type.
* **tile:** The local tile-id of the tile that represents the terrain visually.

Can contain: [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties)

<tile>

* **id:** The local tile ID within its tileset.
* **terrain:** Defines the terrain type of each corner of the tile, given as comma-separated indexes in the terrain types array in the order top-left, top-right, bottom-left, bottom-right. Leaving out a value means that corner has no terrain. (optional) (since 0.9)
* **probability:** A percentage indicating the probability that this tile is chosen when it competes with others while editing with the terrain tool. (optional) (since 0.9)

Can contain: [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties), [image](http://doc.mapeditor.org/reference/tmx-map-format/#image) (since 0.9), [objectgroup](http://doc.mapeditor.org/reference/tmx-map-format/#objectgroup) (since 0.10), [animation](http://doc.mapeditor.org/reference/tmx-map-format/#animation) (since 0.10)

<animation>

Contains a list of animation frames.

As of Tiled 0.10, each tile can have exactly one animation associated with it. In the future, there could be support for multiple named animations on a tile.

Can contain: [frame](http://doc.mapeditor.org/reference/tmx-map-format/#frame)

<frame>

* **tileid**: The local ID of a tile within the parent [tileset](http://doc.mapeditor.org/reference/tmx-map-format/#tileset).
* **duration**: How long (in milliseconds) this frame should be displayed before advancing to the next frame.

<layer>

All <tileset> tags shall occur before the first <layer> tag so that parsers may rely on having the tilesets before needing to resolve tiles.

* **name:** The name of the layer.
* *x:* The x coordinate of the layer in tiles. Defaults to 0 and can no longer be changed in Tiled Qt.
* *y:* The y coordinate of the layer in tiles. Defaults to 0 and can no longer be changed in Tiled Qt.
* *width:* The width of the layer in tiles. Traditionally required, but as of Tiled Qt always the same as the map width.
* *height:* The height of the layer in tiles. Traditionally required, but as of Tiled Qt always the same as the map height.
* **opacity:** The opacity of the layer as a value from 0 to 1. Defaults to 1.
* **visible:** Whether the layer is shown (1) or hidden (0). Defaults to 1.
* **offsetx:** Rendering offset for this layer in pixels. Defaults to 0. (since 0.14)
* **offsety:** Rendering offset for this layer in pixels. Defaults to 0. (since 0.14)

Can contain: [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties), [data](http://doc.mapeditor.org/reference/tmx-map-format/#data)

<data>

* **encoding:** The encoding used to encode the tile layer data. When used, it can be "base64" and "csv" at the moment.
* **compression:** The compression used to compress the tile layer data. Tiled Qt supports "gzip" and "zlib".

When no encoding or compression is given, the tiles are stored as individual XML tile elements. Next to that, the easiest format to parse is the "csv" (comma separated values) format.

The base64-encoded and optionally compressed layer data is somewhat more complicated to parse. First you need to base64-decode it, then you may need to decompress it. Now you have an array of bytes, which should be interpreted as an array of unsigned 32-bit integers using little-endian byte ordering.

Whatever format you choose for your layer data, you will always end up with so called "global tile IDs" (gids). They are global, since they may refer to a tile from any of the tilesets used by the map. In order to find out from which tileset the tile is you need to find the tileset with the highest firstgid that is still lower or equal than the gid. The tilesets are always stored with increasing firstgids.

Can contain: [tile](http://doc.mapeditor.org/reference/tmx-map-format/#tile_1)

Tile flipping

When you use the tile flipping feature added in Tiled Qt 0.7, the highest two bits of the gid store the flipped state. Bit 32 is used for storing whether the tile is horizontally flipped and bit 31 is used for the vertically flipped tiles. And since Tiled Qt 0.8, bit 30 means whether the tile is flipped (anti) diagonally, enabling tile rotation. These bits have to be read and cleared *before* you can find out which tileset a tile belongs to.

When rendering a tile, the order of operation matters. The diagonal flip (x/y axis swap) is done first, followed by the horizontal and vertical flips.

The following C++ pseudo-code should make it all clear:

// Bits on the far end of the 32-bit global tile ID are used for tile flags const unsigned FLIPPED\_HORIZONTALLY\_FLAG = 0x80000000; const unsigned FLIPPED\_VERTICALLY\_FLAG = 0x40000000; const unsigned FLIPPED\_DIAGONALLY\_FLAG = 0x20000000; ... // Extract the contents of the <data> element string tile\_data = ... unsigned char \*data = decompress(base64\_decode(tile\_data)); unsigned tile\_index = 0; // Here you should check that the data has the right size // (map\_width \* map\_height \* 4) for (int y = 0; y < map\_height; ++y) { for (int x = 0; x < map\_width; ++x) { unsigned global\_tile\_id = data[tile\_index] | data[tile\_index + 1] << 8 | data[tile\_index + 2] << 16 | data[tile\_index + 3] << 24; tile\_index += 4; // Read out the flags bool flipped\_horizontally = (global\_tile\_id & FLIPPED\_HORIZONTALLY\_FLAG); bool flipped\_vertically = (global\_tile\_id & FLIPPED\_VERTICALLY\_FLAG); bool flipped\_diagonally = (global\_tile\_id & FLIPPED\_DIAGONALLY\_FLAG); // Clear the flags global\_tile\_id &= ~(FLIPPED\_HORIZONTALLY\_FLAG | FLIPPED\_VERTICALLY\_FLAG | FLIPPED\_DIAGONALLY\_FLAG); // Resolve the tile for (int i = tileset\_count - 1; i >= 0; --i) { Tileset \*tileset = tilesets[i]; if (tileset->first\_gid() <= global\_tile\_id) { tiles[y][x] = tileset->tileAt(global\_tile\_id - tileset->first\_gid()); break; } } } }

(Since the above code was put together on this wiki page and can't be directly tested, please make sure to report any errors you encounter when basing your parsing code on it, thanks.)

<tile>

* **gid:** The global tile ID.

Not to be confused with the tile element inside a tileset, this element defines the value of a single tile on a tile layer. This is however the most inefficient way of storing the tile layer data, and should generally be avoided.

<objectgroup>

* **name:** The name of the object group.
* **color:** The color used to display the objects in this group.
* *x:* The x coordinate of the object group in tiles. Defaults to 0 and can no longer be changed in Tiled Qt.
* *y:* The y coordinate of the object group in tiles. Defaults to 0 and can no longer be changed in Tiled Qt.
* *width:* The width of the object group in tiles. Meaningless.
* *height:* The height of the object group in tiles. Meaningless.
* **opacity:** The opacity of the layer as a value from 0 to 1. Defaults to 1.
* **visible:** Whether the layer is shown (1) or hidden (0). Defaults to 1.
* **offsetx:** Rendering offset for this object group in pixels. Defaults to 0. (since 0.14)
* **offsety:** Rendering offset for this object group in pixels. Defaults to 0. (since 0.14)
* **draworder:** Whether the objects are drawn according to the order of appearance ("index") or sorted by their y-coordinate ("topdown"). Defaults to "topdown".

The object group is in fact a map layer, and is hence called "object layer" in Tiled Qt.

Can contain: [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties), [object](http://doc.mapeditor.org/reference/tmx-map-format/#object)

<object>

* **id:** Unique ID of the object. Each object that is placed on a map gets a unique id. Even if an object was deleted, no object gets the same ID. Can not be changed in Tiled Qt. (since Tiled 0.11)
* **name:** The name of the object. An arbitrary string.
* **type:** The type of the object. An arbitrary string.
* **x:** The x coordinate of the object in pixels.
* **y:** The y coordinate of the object in pixels.
* **width:** The width of the object in pixels (defaults to 0).
* **height:** The height of the object in pixels (defaults to 0).
* **rotation:** The rotation of the object in degrees clockwise (defaults to 0). (since 0.10)
* **gid:** An reference to a tile (optional).
* **visible:** Whether the object is shown (1) or hidden (0). Defaults to 1. (since 0.9)

While tile layers are very suitable for anything repetitive aligned to the tile grid, sometimes you want to annotate your map with other information, not necessarily aligned to the grid. Hence the objects have their coordinates and size in pixels, but you can still easily align that to the grid when you want to.

You generally use objects to add custom information to your tile map, such as spawn points, warps, exits, etc.

When the object has a gid set, then it is represented by the image of the tile with that global ID. The image alignment currently depends on the map orientation. In orthogonal orientation it's aligned to the bottom-left while in isometric it's aligned to the bottom-center.

Can contain: [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties), [ellipse](http://doc.mapeditor.org/reference/tmx-map-format/#ellipse) (since 0.9), [polygon](http://doc.mapeditor.org/reference/tmx-map-format/#polygon), [polyline](http://doc.mapeditor.org/reference/tmx-map-format/#polyline), *image*

<ellipse>

Used to mark an object as an ellipse. The existing x, y, width and height attributes are used to determine the size of the ellipse.

<polygon>

* **points:** A list of x,y coordinates in pixels.

Each polygon object is made up of a space-delimited list of x,y coordinates. The origin for these coordinates is the location of the parent object. By default, the first point is created as 0,0 denoting that the point will originate exactly where the object is placed.

<polyline>

* **points:** A list of x,y coordinates in pixels.

A polyline follows the same placement definition as a polygon object.

<imagelayer>

* **name:** The name of the image layer.
* **offsetx:** Rendering offset of the image layer in pixels. Defaults to 0. (since 0.15)
* **offsety:** Rendering offset of the image layer in pixels. Defaults to 0. (since 0.15)
* *x:* The x position of the image layer in pixels. (deprecated since 0.15)
* *y:* The y position of the image layer in pixels. (deprecated since 0.15)
* *width:* The width of the image layer in tiles. Meaningless.
* *height:* The height of the image layer in tiles. Meaningless.
* **opacity:** The opacity of the layer as a value from 0 to 1. Defaults to 1.
* **visible:** Whether the layer is shown (1) or hidden (0). Defaults to 1.

A layer consisting of a single image.

Can contain: [properties](http://doc.mapeditor.org/reference/tmx-map-format/#properties), [image](http://doc.mapeditor.org/reference/tmx-map-format/#image)

<properties>

Can contain: [property](http://doc.mapeditor.org/reference/tmx-map-format/#property)

Wraps any number of custom properties. Can be used as a child of the map, tile (when part of a tileset), layer, objectgroup and object elements.

<property>

* **name:** The name of the property.
* **type:** The type of the property. Can be string (default), int, float or bool. (since 0.16)
* **value:** The value of the property.

Boolean properties have a value of either "true" or "false".

When a string property spans contains newlines, the current versions of Tiled Java and Tiled Qt will write out the value as characters contained inside the property element rather than as the value attribute. However, it is at the moment not really possible to edit properties consisting of multiple lines with Tiled.

It is possible that a future version of the TMX format will switch to always saving property values inside the element rather than as an attribute.