# Tiled Image Layer Format

The last thing that a map can contain is the image layer. This is the easiest layer to implement.

If you take a look at <http://doc.mapeditor.org/reference/tmx-map-format> and look at <imageLayer> you will see

## <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)

## Description

Now if you create an image layer then open it in a text editor or xml editor you will see something similar to this.

<imagelayer name="Image Layer 1" offsetx="160" offsety="0">

<image source="Tilesets/Liberated Pixel Cup/outdoor.png" width="288" height="576"/>

</imagelayer>

## Code

### TMXImageLayer.cs

namespace TileMapXML.Layers

{

/// <summary>

/// <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, image

/// </summary>

public class TMXImageLayer : TMXLayer

{

#region attributes

// All are in TMXLayer

#endregion

// Properties is in TMXLayer

public Tileset.TMXImage image;

}//public class TMXImageLayer

}//namespace TileMapXML.Layers

The only thing not already done for use is the image

### TMXTest.cs

Fill in the TMXImageLayerLoaded method

#region Image Layer Loaded

private void TMXImageLayerLoaded(TMXImageLayer imageLayer)

{

//Name of the layer must not be null

Assert.IsNotNullOrEmpty(imageLayer.name, "imagelayer must have a name");

// verify that the image is vaild

TMXImageLoaded(imageLayer.image);

}//void TMXImageLayerLoaded(TMXImageLayer imageLayer)

#endregion

Make sure the image layer has a name and the image is valid.