getmap.sty

v1.11

Downloading maps from OpenStreetMap, Google Maps or Google Street View

2019/03/25

Package author:
Josef Kleber

germap@jklatex.de
http://getmap.jklatex.de/
1 Options

1.1 General options ........................................... 5
  1.1.1 mode ............................................. 5
  1.1.2 inputencoding ...................................... 5
  1.1.3 overwrite .......................................... 5
  1.1.4 file ................................................ 5

1.2 osm mode .................................................. 5
  1.2.1 key .................................................. 5
  1.2.2 scale ............................................... 6
  1.2.3 zoom ................................................. 6
  1.2.4 xsize ............................................... 6
  1.2.5 ysize ............................................... 6
  1.2.6 imagetype ......................................... 6
  1.2.7 type ................................................ 6
  1.2.8 color ................................................. 6
  1.2.9 number .............................................. 7

1.3 gm mode ................................................... 7
  1.3.1 scale ............................................... 7
  1.3.2 zoom ............................................... 7
  1.3.3 xsize ............................................... 7
  1.3.4 ysize ............................................... 7
  1.3.5 imagetype ......................................... 7
  1.3.6 type ................................................ 7
  1.3.7 color ................................................. 8
  1.3.8 number .............................................. 8
  1.3.9 language .......................................... 8
  1.3.10 markers ........................................... 8
  1.3.11 visible ........................................... 9
  1.3.12 path .............................................. 10
  1.3.13 pathfile ........................................... 11

1.4 gsv mode .................................................. 11
  1.4.1 xsize ............................................... 11
  1.4.2 ysize ............................................... 11
  1.4.3 heading ............................................ 12
  1.4.4 pitch ............................................... 12
  1.4.5 fov ................................................ 12

2 Command(s) .................................................. 12
  2.1 \getmap ............................................... 12

3 Examples ..................................................... 14

4 The getmapdl Lua script ................................... 17

5 How to define routes ....................................... 18
  5.1 OpenStreetMap ....................................... 18
  5.2 Google Maps ......................................... 19
    5.2.1 Long routes .................................... 22
6 Implementation 24
7 References 32
8 Change History 33
9 Index 34
Abstract

The goal of this package is the simplest possible provision of map images (OpenStreetMap, Google Maps and Google Street View are supported). In the simplest case, it is sufficient to specify an address. The package loads the map using the `\write18` feature, which you must activate to use this package. The image will be downloaded by an external Lua script. You can use this script also from the command line.

Acknowledgment

I want to thank Norbert Preining, who did most of the recoding of osmimage (Bash → Lua). Moreover many thanks to Taco Hoekwater, Akira Kakuto, Reinhard Kotucha and Heiko Oberdiek for their valuable contributions. Finally, I want to thank Doug Currie for helping me to implement the algorithm for encoded polylines in Lua.
1 Options

The following options can be used as package options with global scope, as well as options for the \getmap command with local scope!

1.1 General options

1.1.1 mode (osm|gm|gsv)

This option sets the mode, that is the source of the images. OpenStreetMap, Google Maps or Google Street View! Please note that – if used as local option (mixed modes) – the default values of the scale, zoom, type and color options for the respective mode are reset to guarantee correct download URLs!

1.1.2 inputencoding

This option specifies the input encoding of your file. The download script requires the strings encoded in utf8. For the safe conversion the input encoding of the file is required. Normally, you don’t have to specify an encoding. The package tries to evaluate the encoding given to inputenc or assumes utf8. Usually that should work.

1.1.3 overwrite (false|true)

With this option, you can specify whether the image should be downloaded in any case. By default, the option is set to false in order to save bandwidth and compilation time. Nevertheless a check is performed on the existence of the image and the image will be downloaded, if it is not present. In the case of true, the image will be downloaded anyway! BTW, overwrite is equivalent to overwrite=true.

1.1.4 file (getmap)

This option allows you to specify the name of the image (without extension).

1.2 osm mode

1.2.1 key (Fmjtd|luur20u22d,75=o5-9aylh6)

In osm mode, the download script requires a key in order to use the service of MapQuest. By default, it uses a key, which is registered for getmap. But you can register and use your own key with this option. The default key is stored in
getmap.cfg. You can copy this file to your local \TeX{} tree and store your own key there\footnote{Mapquest will deliver an url-encoded key, which must be decoded to ASCII, e.g. by \texttt{Url decode}}! This file will be found after running \texttt{texhash}!

### 1.2.2 scale (3385)

This option allows you to specify a display scale for the map image in the range of 1692 – 221871572. You will not necessarily see a difference between 5000 and 5500. A scale value of 3385 corresponds to a zoom level of 17.

### 1.2.3 zoom

This option allows you to specify a zoom level in the range of 1 – 18. This option overwrites a possibly given scale.

### 1.2.4 xsize (600)

This option specifies the width of the map in pixels. If you only want to slightly increase or decrease the map extract, you should adjust the size of the map. You still have full control over the size of the map in the document with the options of \texttt{\includegraphics}. (max: 3840)

### 1.2.5 ysize (400)

This option specifies the height of the map in pixels. (max: 3840)

### 1.2.6 imagetype (png|jpeg|jpg|gif)

This option allows you to specify the type of the image.

### 1.2.7 type (map|sat|hyb)

This option specifies the type of the map. It seems as if there would be only a few regions of Mother Earth, for which satellite and hybrid images are available.

### 1.2.8 color (yellow\_1)

This option specifies the color of the marker. Possible colors:

\url{http://open.mapquestapi.com/staticmap/icons.html}
1.2.9 number (1)

This option specifies the number of the marker.

1.3 gm mode

From 2018 on, Google Maps APIs are no longer accessible without a key. Since getmap.sty is an open source project, it is impossible to finance an infinite number of requests, but Google allows a limited number of requests for “free”. As a consequence, the following restrictions arise:

<table>
<thead>
<tr>
<th>Service</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>2000 requests per day</td>
</tr>
<tr>
<td>Google Street View</td>
<td>1000 requests per day</td>
</tr>
</tbody>
</table>

1.3.1 scale (1)

For the free version of Google Maps the image size is limited to 640x640. You can set scale to a value of 2, to get exactly the same map in doubled size in pixels.

1.3.2 zoom (17)

This option allows you to specify a zoom level in the range of 0 – 21.

1.3.3 xsize (600)

This option specifies the width of the map in pixels. If you only want to slightly increase or decrease the map extract, you should adjust the size of the map. You still have full control over the size of the map in the document with the options of \includegraphics. (max: 640)

1.3.4 ysize (400)

This option specifies the height of the map in pixels. (max: 640)

1.3.5 imagetype (png|png8|png32|gif|jpg (progressive)|jpg-baseline (flat))

This option allows you to specify the type of the image.

1.3.6 type (roadmap|satellite|hybrid|terrain)

This option specifies the type of the map.
1.3.7 color (blue)

This option specifies the color of the marker. Possible colors:
black, brown, green, purple, yellow, blue, gray, orange, red, white or in hex format 0x3399FF

1.3.8 number (1)

This option specifies the number of the marker. Google Maps also allows uppercase letters: [A-Z]!

1.3.9 language (en)

This option specifies the language of the map labels. Of course, not all languages are supported for all countries. At least, english and one of the national languages should be supported. Possible option values: en, de, fr, es, it, fi, ...

1.3.10 markers

This option allows you to set more than just the standard marker, which will no longer be used! You don’t have to specify an address, as Google Maps will deliver an image with all markers on the map. Nevertheless, you can specify an address, which will define the center of the map. This option expects one or more URL parameters like:

&markers=size:mid|color:blue|label:S|loc1|loc2|...
Earlier versions of this document used POIs for all museums. After an update from Google Maps, the quality of the geo codings deteriorated, at least in this example from Berlin. Using addresses or geographical coordinates usually solves this problem. See [7] for more information.

The parameters size, color and label are optional!

**size** tiny, mid, small

**color** red, black, brown, green, purple, yellow, blue, gray, orange, white, 0x1188FF

**label** [0-9][A-Z] (only in mid size!)

The default is a mid-sized red bubble with a black point!

**1.3.11 visible**

With this option you can specify a list of locations (separated by a pipe), which must be on the map!
\getmap{
file=bmus2, mode=gm
markers={&markers=size:mid|label:B|color:green|52.521847,13.394398
 &markers=label:P|color:green|Pergamonmuseum, Berlin
 &markers=label:N|color:blue|52.520063,13.397525},%
visible={Brandenburger Tor, Berlin|Reichstagsufer 1, Berlin}}{}
\includegraphics[width=10cm]{bmus2}

1.3.12 path

With this option you can define one or more paths! It expects one or more URL parameters like:
\&path=weight:5|color:orange|loc1|loc2|...

\getmap{file=bmus3, mode=gm, language=de,
markers={&markers=size:mid|label:B|color:green|52.521847,13.394398
 &markers=label:P|color:green|Pergamonmuseum, Berlin
 &markers=label:N|color:blue|52.520063,13.397525},%
path=&path=weight:5|color:orange|52.521847,13.394398|% Pergamonmuseum, Berlin|52.520063,13.397525|% James-Simon-Park,Berlin|52.522649,13.402523|
 &path=weight:5|color:purple|James-Simon-Park, Berlin|% Weinmeisterstraße 6, Berlin}}{}
\includegraphics[width=10cm]{bmus3}

You can also use fillcolor to mark areas! In paths, you can also specify RGB32 colors, in which the last byte defines opacity, e.g. 55 (33%).
With small enough spaces between way points you can also defines routes!

1.3.13 pathfile

This option specifies the file holding the path specification. It will be loaded by the Lua script. You can use the filecontents* environment to keep the definition in your document. It should be a one line utf8-encoded file!

1.4 gsv mode

1.4.1 xsize (600)

This option specifies the width of the map in pixels. (max: 640)

1.4.2 ysize (400)

This option specifies the height of the map in pixels. (max: 640)
1.4.3 heading (\textdegree)

This option specifies the heading (direction) in degrees in the range of 0 – 360. (0: north, 90: east, ...)

1.4.4 pitch (\textdegree)

This option specifies the pitch (angle) of the camera view in degrees in the range of -90 – 90.

1.4.5 fov (\textdegree)

This option specifies the field of horizontal view (kind of zoom) in degrees in the range of 0 – 120.

2 Command(s)

2.1 \getmap

\getmap\{\text{options}\}\{\text{address}\}

With the \texttt{\getmap} command you can download a map, if you enable \texttt{\write18} (TeXLive: -shell-escape, MiKTeX: --enable-write18). This is only necessary if you actually download an image. You can use the options described above to specify the properties of the downloaded image. After executing the command, the image is available in the current working directory!

In the simplest case, you only need an address, a POI or geographic coordinates (latitude, longitude) to download the map. \{\texttt{\textbf{address}}\} must be fully expanded and must not contain macros! By default, the image is saved under the name \texttt{getmap.png}. If you need only one map (e.g. the office of Dante e.V.) in your document, it can be as simple as:

\begin{center}
\includegraphics{getmap.png}
\end{center}
\getmap{Bergheimer Straße 110A, 69115 Heidelberg, Germany}
\includegraphics[width=9cm]{getmap}
3 Examples

The same map as before from Google Maps:

\getmap[\text{file=dantegm, mode=gm}]{\text{Bergheimer Straße 110A, 69115 Heidelberg, Germany}}

The same map as satellite image:

\getmap[\text{file=dantegmsat, mode=gm, type=satellite}]{\text{Bergheimer Straße 110A, 69115 Heidelberg, Germany}}
L’afrique, mon amour!

L’amour, ...

1 \getmap\[file=africa,mode=gm,type=terrain,xsize=500,ysize=500,\
% scale=2,zoom=3\]{0,16}
3 \includegraphics[width=9cm]{africa}

1 \getmap\[file=paris,mode=gm,type=hybrid,xsize=500,ysize=300,\
% scale=2,zoom=16\]{Tour Eiffel, Paris}
3 \includegraphics[width=9cm]{paris}
Street View now:

View from Olympic Tower Munich (Olympic Stadium and Park):
4 The getmapdl Lua script

Basically, the getmapdl Lua script downloads static map images depending on command line options and allows to parse kml, gpx and gps (a plain list of geographical coordinate pairs (latitude, longitude) on each line) files and outputs gps or encoded polylines (epl). The script offers the following modes (-m):

- **osm** downloads a static map image based on OpenStreetMap data
- **gm** downloads a static map image based on Google Maps data
- **gsv** downloads an image based on Google Street View data
- **kml2epl** parses a kml file and outputs geographical coordinates of places and encoded polylines (epl) for routes and lines to STDOUT
- **kml2gps** parses a kml file and outputs geographical coordinates
- **gpx2epl** parses a gpx file and outputs encoded polylines
- **gpx2gps** parses a gpx file and outputs a list of geographic coordinate pairs (gps)
- **gps2epl** parses a gps file and outputs epl
- **gps2gps** parses a gps file and outputs – based on a given bound – a reduced list of gps coordinates

The first three modes are used by \getmap. You may use the script also from the command line! getmapdl -h will give you a list of available command line options.

The other modes are useful for creating encoded polylines (epl), which is the route format of Google Maps. You can parse the following example from Google Maps in gpx format

```xml
<trkseg>
  <trkpt lon="-120.2" lat="38.5"/>
  <trkpt lon="-120.95" lat="40.7"/>
  <trkpt lon="-126.453" lat="43.252"/>
</trkseg>
```

with

```
$ getmapdl -m gpx2epl -G test.gpx
   _p-if-ps|U_ullnnqClqNvxq’@
```

This encoded polyline can be used for the path or pathfile option of \getmap.
5 How to define routes

Routes are described by so called encoded polylines and can be used with `enc:polyline_data` as location specifier in a path. This string can contain all sorts of troublesome characters for \LaTeX. `\getmap` can deal with them, with the exception of curly braces! These will break your \LaTeX document. As a work-around, use the `pathfile` option. Please note that the length of the URL is limited to 2048 bytes. So, there’s no way to support extreme long paths!

5.1 OpenStreetMap

OpenStreetMap does not offer routing service directly, but you can use an OpenStreetMap based route service\(^2\) to create your route and export it to a `.gpx` file\(^3\). It’s basically a xml-packaged list of geographical coordinates. You can use the `getmapdl` script to convert a route to encoded polylines, e.g. a pedestrian route from Berlin Central Station to Brandenburg Gate:

\begin{filecontents*}{berlin.epl}
&path=weight:5|color:purple|enc:_xq_IcgrpA?AFE@?^BFE@A^U@CLQXEZU?gCR?B?DFe@?vA?D?D?BAHE@JBN@JLGFCG[DC–C?q?F?q7vA?p@iB@iFe@JWRSTO
f@Gh@C^A?e@?gE?w@r@?lB@hA?^@??M?aA?]dI??0?0?Cn@cBfBeF|AeEHNVBc@H
\end{filecontents*}

\(^2\)http://maps.openrouteservice.org
\(^3\)This also means that you can visualize your own routes tracked with hardware or a software app!
5.2 Google Maps

One possible way is to use Google Maps’ online interactive map tool.\footnote{https://developers.google.com/maps/documentation/utilities/polylineutility}

\begin{filecontents*}{muc.epl}
\&path=weight:5|color:orange|enc:okydHa\peAXOH\Hi\jANV@A@kJJmFAiDJuA
\end{filecontents*}

\getmap[\textit{file}=muc, \textit{mode}=gm, \\
\textit{markers}={\&markers=size:mid|label:H|color:green|München, Hbf \\
\&markers=label:S|color:yellow|Stachus, München \\
\&markers=label:M|color:blue|Marienplatz, München}, \\
\textit{pathfile}={muc.epl})}
\end{filecontents*}

You can also use the \texttt{new} version of \texttt{My Maps}. It allows you to define markers, routes and arbitrary lines on different layers and to export these into a \texttt{kml} file, e.g \texttt{berlin.kml}\footnote{https://bitbucket.org/kleberj/getmap/downloads/Berlin.kml}:
5 How to define routes

$ getmapdl -m kml2epl -K Berlin.kml

Route: Route von Berlin Hbf, Moabit nach Brandenburger Tor, Pariser Platz, Berlin

Point: Berlin Hbf, Moabit [Europaplatz 1, 10557 Berlin, Deutschland]
52.52581820000001, 13.3695451

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.5159999, 13.3773202

Point: Berlin Hbf [arrive with train]
52.5249948, 13.368988

Point: Reichstag [nice view from the roof]
52.5185973, 13.3758974

Point: Brandenburger Tor [once behind the wall]
52.5163514, 13.3789873

Route: Route von Berlin Hbf, Moabit nach Pariser Platz, Berlin

Point: Berlin Hbf, Moabit [Europaplatz 1, 10557 Berlin, Deutschland]
52.52581820000001, 13.3695451

Point: Deutscher Bundestag Redaktion Das Parlament, Berlin [Plat...
How to define routes

Point: Pariser Platz, Berlin  [Pariser Platz 1, 10117 Berlin, Deutschland]
52.5160749, 13.3783013

Now, you can take these data for your map:

```latex
\begin{filecontents*}{berlin2.epl}
&path=weight:5|color:orange|enc:k}q_IufrpA?iFQ?&hH@H@H@F@B@B@?DB?\?
|B@F@DB@B@–FCB?X@V??Q?0O?0wF\W?o?k?@?Y?eAyAD’?@?@ARCFAP?hAAZ?B?D?
J7B?DAJB^~T?–B?MsAAIAEAM?5Q@0M7A@IKBKDKFKFIJEFCJAJATAL?XAN?P?VAL
BCbB?L@RB?kHq]}GPQ]|@Oa@8Az@JTF“LZN?wF@aCE}–Am@\KREJCNANAHz@BJHJHF
LP\DBFFDB@@?D@F?JAJCKiEAO?K@?KBSXC&path=weight:5|color:purple|enc:k}q_IufrpA?iFQ?gCDQBMHSm@Wq@GQIK{\AUaA{AaICIMm@Kg@EUiByIi@cD5
qA_uBIa@a?qBpB[fAMhCS|@GdqEvEEp@GfAIpC{bAMr@bAMpG]\gtBudCSp@MrA
K–AOpAmhAKx@Ijc@VAB?D@7?dCU‘AKjAMRCPLGNetEBhAByBVF‘@D’@\rB{Z
KDAHCLAHABHBFHFFDNHRHJHDDHDHBF7FVJCJILQJUJg@HQMJKHAFAJ7X?bCR
H@R?AWQ?KaiBkg@GYAK@EBIFEFEEFCFAD
\end{filecontents*}
\getmap{file=berlin2, language=de, xsize=400, ysize=600, 
scale=2, mode=gm, 
markers={&markers=size:mid|label:H|color:green}52.5249948, 13.3689
```
5.2.1 Long routes

Long routes are defined by a huge number of way points, but the URL length is limited to 2048 bytes. The following example (Stachus, Munich → Brandenburg Gate, Berlin) consists of more than 6000 way points. A created polyline would hugely exceed the URL length limit!

After transferring the kml file into a gps file, you can use the gps2gps mode of getmapdl to reduce the number of way points by a given bound. It takes a new pair of gps coordinates only if the difference of latitude or longitude is larger than the given bound! Finally, you can use the new gps file to create an encoded polyline.

6https://bitbucket.org/kleberj/getmap/downloads/MucBer.kml
$ getmapdl -m kml2gps -K MucBer.kml >MucBer.gps

$ cat MucBer.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin
48.13903, 11.56556
48.1392, 11.56562
[ ... many, many way points ...]
52.5159, 13.37735
52.51591, 13.37732

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732

$ getmapdl -m gps2gps -G MucBer.gps -B 0.025 >MucBerR.gps

Route 1: reduced gps coordinates (Bound = 0.025): 6119 -> 193

$ getmapdl -m gps2epl -G MucBerR.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732

$ getmapdl -m kml2gps -K MucBer.kml >MucBer.gps

$ cat MucBer.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin
48.13903, 11.56556
48.1392, 11.56562
[ ... many, many way points ...]
52.5159, 13.37735
52.51591, 13.37732

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732

$ getmapdl -m gps2gps -G MucBer.gps -B 0.025 >MucBerR.gps

Route 1: reduced gps coordinates (Bound = 0.025): 6119 -> 193

$ getmapdl -m gps2epl -G MucBerR.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732

$ getmapdl -m kml2gps -K MucBer.kml >MucBer.gps

$ cat MucBer.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin
48.13903, 11.56556
48.1392, 11.56562
[ ... many, many way points ...]
52.5159, 13.37735
52.51591, 13.37732

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732

$ getmapdl -m gps2gps -G MucBer.gps -B 0.025 >MucBerR.gps

Route 1: reduced gps coordinates (Bound = 0.025): 6119 -> 193

$ getmapdl -m gps2epl -G MucBerR.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732

$ getmapdl -m kml2gps -K MucBer.kml >MucBer.gps

$ cat MucBer.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin
48.13903, 11.56556
48.1392, 11.56562
[ ... many, many way points ...]
52.5159, 13.37735
52.51591, 13.37732

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732

$ getmapdl -m gps2gps -G MucBer.gps -B 0.025 >MucBerR.gps

Route 1: reduced gps coordinates (Bound = 0.025): 6119 -> 193

$ getmapdl -m gps2epl -G MucBerR.gps

Route: Route von Stachus, München nach Brandenburger Tor, Pariser Platz, Berlin

Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschland]
48.13903, 11.56556

Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21, 10117 Berlin, Deutschland]
52.51591, 13.37732
Taking a look into the log file, we find:

```lua
getmapdl.lua:
url = http://maps.googleapis.com/maps/api/staticmap? ...
url length = 1866 bytes
output = mucber.png
```

With 193 way points we almost reached the URL length limit of 2048 bytes. The accuracy of the encoded polyline is obviously good enough. So, about 200 way points seem to be a good choice. But the length of an encoded pair of gps coordinates depends on the space between two points and may vary between 2 and 8 bytes!

### 6 Implementation

First, we provide the \LaTeX\ package `getmap`.

```latex
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{getmap}[2019/03/25 v1.11 getmap.sty - Josef Kleber (C) 2014,2016,2018,2019]
```

We need a few packages!

```latex
\RequirePackage{xkeyval}\
\RequirePackage{stringenc}\
\RequirePackage{ifthen}
```

Newer versions of Lua\TeX\ v0.85+ no longer supports \texttt{\write18}! Therefore, we use \texttt{shellesc} instead.

```latex
\RequirePackage{shellesc}
```

Fix a bug with problematic url encodings coming up with luatex v1.0.7 (TeXLive 2018) fixed by Akira Kakuto (see: TeXLive mailing list)

```latex
\RequirePackage{ifxetex}\
\ifxetex
\def\StringEncodingConvert#1#2#3#4{}
\fi
\ifluatex
\def\StringEncodingConvert#1#2#3#4{}
\fi
```

We provide a macro \texttt{\@JK@define@key}, which defines package options with global scope and options for \texttt{\getmap} with local scope. It takes four arguments 
\texttt{\{\textit{prefix}\}, \{\textit{package}\}, \{\textit{option}\} and \{\textit{default}\}}.

\begin{verbatim}
\newcommand*{\GM@JK@define@key}[4]{
  \expandafter\gdef\csname#1@#3\endcsname{#4}
  \define@key{#2.sty}{#3}{#4}
}{
  \expandafter\gdef\csname#1@#3\endcsname{\detokenize{##1}}
  \define@key{#2.sty}{#3}{\detokenize{##1}}
}{
  \expandafter\def\csname#1@#3\endcsname{\detokenize{##1}}
  \define@key{#2.sty}{#3}{\detokenize{##1}}
}
\end{verbatim}

Now, we can use this macro to define our options.

\begin{verbatim}
\GM@JK@define@key{GM@JK}{getmap}{mode}{osm}
\GM@JK@define@key{GM@JK}{getmap}{key}{}
\GM@JK@define@key{GM@JK}{getmap}{xsize}{600}
\GM@JK@define@key{GM@JK}{getmap}{ysize}{400}
\GM@JK@define@key{GM@JK}{getmap}{scale}{3385}
\GM@JK@define@key{GM@JK}{getmap}{zoom}{}
\GM@JK@define@key{GM@JK}{getmap}{type}{map}
\GM@JK@define@key{GM@JK}{getmap}{imagetype}{png}
\GM@JK@define@key{GM@JK}{getmap}{color}{yellow_1}
\GM@JK@define@key{GM@JK}{getmap}{number}{1}
\GM@JK@define@key{GM@JK}{getmap}{heading}{0}
\GM@JK@define@key{GM@JK}{getmap}{fov}{90}
\GM@JK@define@key{GM@JK}{getmap}{pitch}{0}
\GM@JK@define@key{GM@JK}{getmap}{language}{en}
\GM@JK@define@key{GM@JK}{getmap}{markers}{}
\GM@JK@define@key{GM@JK}{getmap}{path}{}
\GM@JK@define@key{GM@JK}{getmap}{visible}{}
\GM@JK@define@key{GM@JK}{getmap}{pathfile}{}
\GM@JK@define@key{GM@JK}{getmap}{file}{getmap}
\end{verbatim}
For options without default value, we define reasonable default values! We overwrite the default for `overwrite`, because we don’t want `overwrite` to be true by default, but that `overwrite` is equivalent to `overwrite=true`!

Moreover, we load `getmap.cfg` to set the default key. You can copy this file to your local TeX tree and replace the key with your own!

We try to use the input encoding specified for `inputenc` or `utf8` instead.

Later, we will need a switch, if \write18 is enabled.

We execute the package options to define and set the option macros.

We need to reset some defaults in `gm` mode.
We check if `\pdf@shellescape` is available to test if `\write18` is enabled. If false, we assume `\write18` is available and hope for the best. If true, we set the switch `\GM@JK@writexviii` accordingly!

We define a macro that is executed as `\ShellEscape` call. First, we test if `\write18` is enabled and issue a package error if not! Otherwise we execute `\ShellEscape` depending on the mode.
Here, we define the user command to download the map.

\getmap\[\langle\text{options}\rangle\]\{\langle\text{address}\rangle\}

We start a group to keep the setting of options local. Then we test the mode to
reset some defaults! Finally, we set the local options again to override defaults if necessary!

```latex
\begingroup
\setkeys{getmap}{#1}
\ifthenelse{\equal{\GM@JK@mode}{gm}}{%
  \def\GM@JK@scale{1}
  \def\GM@JK@zoom{17}
  \def\GM@JK@type{roadmap}
  \def\GM@JK@color{blue}
}%
\ifthenelse{\equal{\GM@JK@mode}{osm}}{%
  \def\GM@JK@scale{3385}
  \def\GM@JK@zoom{3385}
  \def\GM@JK@type{map}
  \def\GM@JK@color{yellow_1}
}%
\setkeys{getmap}{#1}
\PackageInfo{getmap}{using \GM@JK@inputencoding\space encoding}
\StringEncodingConvert{\GM@JK@location@string}{\detokenize\expandafter{\GM@JK@location}}{\GM@JK@inputencoding}{utf-8}
\StringEncodingSuccessFailure{%success
%failure
\errmessage{Converting to UTF-8 failed}}
\StringEncodingConvert{\GM@JK@key@string}{\detokenize\expandafter{\GM@JK@key}}{\GM@JK@inputencoding}{utf-8}
\StringEncodingSuccessFailure{%success
%failure
\errmessage{Converting to UTF-8 failed}}
\StringEncodingConvert{\GM@JK@location{\#2}}{\GM@JK@imagetype{jpg}}{\GM@JK@location}{\GM@JK@imagetype}{jpg}
```

In gs mode, we have an implicit imagetype=jpg. Therefore, we have to set it to allow the later test on the existence of the image file!

```latex
\ifthenelse{\equal{\GM@JK@mode}{gsv}}{%
  \def\GM@JK@imagetype{jpg}
%\PackageInfo{getmap}{using \GM@JK@inputencoding\space encoding}}
\def\GM@JK@imagetype{jpg}
\StringEncodingConvert{\GM@JK@location{\#2}}{\GM@JK@imagetype{jpg}}{\GM@JK@location}{\GM@JK@imagetype}{jpg}
texlua expects its arguments encoded in utf8!

```
We check, if overwrite is true and download the map. If not, we check if the image is already in the working directory. If not, we download the image!

\ifthenelse{\equal{\GM@JK@overwrite}{true}}%  
{\GM@JK@shellescape%  
}%  
\IfFileExists{\GM@JK@file.\GM@JK@imagetype}%  
{  
\PackageInfo{getmap}{overwrite=false; (\GM@JK@file.\GM@JK@imagetype)\ using existing file!}%  
}%  
{  
\PackageInfo{getmap}{overwrite=false; (\GM@JK@file.\GM@JK@imagetype)% file does not exist! downloading ...}%  
}
\G@JK@shellescape%
}
\endgroup%
\endgroup
</package>
7 References


8 Change History

v1.0
General: CTAN upload  ............. 24

v1.1
\getmap: Bugfix: problem in URL when using \usepackage[utf8]{inputenc} 28

v1.10
General: getmapdl.lua: fix problem (unable to load socket.url module) caused by bug in luatex v1.0.7 (TeXLive2018)  ............... 24
Bugfix: fixes encoding problems with luatex v1.0.7 (TeXLive2018)  ............... 27

v1.11
General: getmapdl.lua: added API key for Google Maps API, which is no longer accessible without key  ............. 24

v1.2
General: added getmap.cfg to store default key (FR by Ulrike Fischer) ............... 26
added support for Google Maps 24
changed default values of xsize (600) and file (getmap) .... 25
renamed osmimage.lua to getmapdl.lua  ............. 24

v1.3
General: added support for Google Street View ............. 24

v1.4
General: getmapdl.lua: added options language, markers, visible, path and pathfile in gm mode  ............. 24

v1.5
General: added gp2gps bash script  ............. 24

v1.6
General: getmapdl.lua: added gp2epl, gps2epl and gpx2gps modes  ............. 24
removed gpx2gps bash script . 24

v1.7
General: getmapdl.lua: added kml2gps and kml2epl modes .
getmapdl.lua: added multi track support for gpx2gps and gpx2epl mode  ............. 24
getmapdl.lua: revised gps2epl mode  ............. 24

v1.8
General: getmapdl.lua: added gps2gps mode to reduce way points with a given bound (default: 0.1) ............. 24
getmapdl.lua: added check of URL length and output to log 24
getmapdl.lua: added multi route support for gps2epl mode  ............. 24
getmapdl.lua: added rounding of gpx coordinates to kml2gps and gpx2gps modes  ............. 24

v1.9
\getmap: Bugfix: restore mode defaults for scale, zoom, type and color options to guarantee correct download URLs  .... 28
General: LuaTgX v0.85+ fix: use shellesc’s \ShellEscape instead of \write18  .... 27
Symbols
\@ifpackageloaded ........ 77
\define@key ........ 24, 28, 36, 40
\detokenize .... 38, 42, 211, 222
\directlua .... 123, 140, 162
G
getmap (Package) .... 5, 24
getmap ........ 184
\GM@JK@color 97, 133, 149, 193, 201
\GM@JK@define@key 21, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 62, 63, 64, 65
\GM@JK@define@key@detok 33, 59, 60, 61
\GM@JK@file 135, 157, 170, 269, 271, 275
\GM@JK@fov .... 168
\GM@JK@heading .... 167
\GM@JK@imagetype 132, 148, 206, 269, 271, 275
\GM@JK@inputencoding .. 79, 82, 207, 212, 223, 234, 245, 256
\GM@JK@key .... 67, 74, 222
\GM@JK@key@string .. 126, 221
\GM@JK@language .... 151
\GM@JK@location .... 208, 211
\GM@JK@location@string .. 124, 141, 153, 163, 210
\GM@JK@markers .... 233
\GM@JK@markers@string 152, 232
\GM@JK@mode . 92, 121, 138, 160, 188, 196, 205
\GM@JK@number .... 134, 150
\GM@JK@overwrite .... 66, 264
\GM@JK@path .... 244
\GM@JK@path@string .. 154, 243
\GM@JK@pathfile .... 155
\GM@JK@pitch .... 169
\GM@JK@scale 94, 130, 146, 190, 198
\GM@JK@shellescape .. 118, 266, 277
\GM@JK@type . 96, 131, 147, 192, 200
\GM@JK@visible .... 255
\GM@JK@visible@string 156, 254
\GM@JK@writeviiiifalse 85, 114
\GM@JK@writeviiitrue 106, 112
\GM@JK@xsize .... 127, 143, 165
\GM@JK@ysize .... 128, 144, 166
\GM@JK@zoom . 95, 129, 145, 191, 199
I
\IfFileExists .... 69, 269
\ifGM@JK@writeviii .. 85, 120
\ifluatex .... 15, 123, 140, 162
\ifxetex . 9
\input ........ 71
inputenc (Package) .... 5, 26
inputencodingname .... 79
L
\ltx@IfUndefined .... 102
P
Package
getmap .......... 5, 24
inputenc ........ 5, 26
shellesc ........ 24
\pdf@shellescape 104, 109, 110
S
\setkeys .... 187, 204
shellesc (Package) .... 24
\ShellEscape .... 124, 141, 163
\StringEncodingConvert 10, 16, 209, 220, 231, 242, 253
\StringEncodingSuccessFailure 213, 224, 235, 246, 257
Style option
overwrite .... 26, 30