GmbH Informationssysteme Engineering & Consulting
Technoparkstrasse 1
8005 Zürich
Tel.: 044 / 350 10 10
Fax.: 044 / 350 10 19

GeoShop Mapping API

Abstract

This documentation describes the GeoShop Mapping API (GSAPI).

Table of Contents

1. Introduction

1.1. Overview

The **GeoShop Mapping API (GSAPI)** is a JavaScript based programming interface to the GeoShop Server. With the GSAPI a JavaScript programmer can easily integrate GeoShop generated maps in his or her application. The GSAPI consists of the following main objects:

GSMap
: A visual respresentation of a GeoShop. The GSMap object may be integrated in a HTML DIV element in the user application.

GSMapZoomSliderControl
: A zoom control object to manipulate the GSMap object.

GSMapTools
: Some tools to query or select information from a GSMap object.

To fully understand this documentation the reader should be familiar with JavaScript programming and the general structure of HTML documents.

1.2. Content of this documentation

This documentation contains the following information:

- Chapter 2: Detailed description of the GSMap object.
- Chapter 3: Detailed description of the GSMapZoomSliderControl object.
- Chapter 4: Detailed description of the GSMapTools object.

1.3. Supplementary documentation

[1] GeoShop Server Konfigurationshandbuch (in German).

1.4. Conventions

In this documentation the following formatting conventions are used:

italics	filenames
fat	new concepts, names of functions or methods
`courier`	program text or inputs in the operating system

2. GSMap Object

2.1. Introduction

The GSMap object is the main object of an GSAPI application. Any JavaScript application has to instantiate at least one GSMap object on the HTML page.

2.2. Minimal HTML Skeleton Code

The following minimal skeleton code shows the basic structure of an GSAPI application:

```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html>
    <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>infoGrips GeoShop GSAPI Example</title>
(1) <script type="text/javascript" src="<SERVER-URL>/gsapi/gsapi.js"></script>
(2) <script type="text/JavaScript">
        function MyInfoToolHandler(objects){
           // user handler funcion for InfoTool events
        }
        function MySelectionToolHandler(objects){
           // user handler function for SelectionTool events
        }
    </script>
    <script type="text/JavaScript">
(3)    function mapLoad() {

          // check browser
          if (!GSBrowserCheck()) return;

          // create map object GSMap
          var map = new GSMap("map");

          // set user and center
          map.setUser(<USER>,<PASSWORD>);

          // add zoom controls to GSMap
          var zctrl = new GSMapZoomSliderControl();
          zctrl.setSliderResolution(<MINRES>,<MAXRES>.<STEPS>,<ACTSTEP>);
          map.addZoomControl(zctrl);

          // add tools to GSMap
          var tools = new GSMapTools();
          tools.addInfoTool();
          tools.setInfoToolHandler(MyInfoToolHandler);
          tools.addSelectionTool();
          tools.setSelectionToolHandler(MySelectionToolHandler);
          map.addTools(tools);

          // display data in the map with a given center
          map.setCenter(<X-COORD>,<Y-COORD>);
```

```
      }
    </script>
    </head>
(4) <body onload="mapLoad()">
(5)    <DIV id="map" style="position:absolute; width:500px; height:300px;
                            background-color: #CCCCCC; left: 100px; top: 100px;">
       </DIV>
    </body>
    </html>
```

Description:

1. To use the GSAPI the JavaScript library `gsapi.js` has to be included in the HTML page. The library is loaded from the GeoShop Server at `<SERVER-URL>`. If the GSAPI resides on the same server as the main page, the path can be set relativ to the main page,
2. The user can specify handler functions to handle InfoTool or SelectionTool events.
3. A user written JavaScript-function `mapLoad()` creates the Map object and initializes the GUI controls.
4. In the HTML `<body>` element the user function `mapLoad()` is called once by the HTML `DIV onload` trigger.
5. The HTML `DIV` element determines the size and position of the GSMap object in the HTML page.

2.3. GSMap Methods

Constructor	**`GSMap(String divid)`**
Description	Creates a new GSMap object. The `divid` argument has to correspond to the `id=` of a DIV-element on the HTML page.
Example	`var map = new GSMap("map");`

Method	**`setServer(String server)`**
Description	Sets the GeoShop Server which delivers the data for the map. If the GeoShop Server has the same location as the map html, the server must not be set.
Example	`map.setServer("http://www.myserver.com");`

Method	**`setIconUrl(String url)`**
Description	Sets the url for the icons of the map. If the icons of the GSAPI are used, the icon url has not to be defined. More about the icons see later.
Example	`map.setIconUrl("http://www.myserver.com/icons");`

Method	**`setIconExtension(String extension)`**
Description	Sets the extension for the icons of the map. If the icons of the GSAPI are used, the icon extension has not to be defined. More about the icons see later.
Example	`map.setIconUrl("gif");`

Method	**`setUser(String user,String password)`**

Description Sets the map object to a certain GeoShop `user/password`. The user must exist on the GeoShop server and must have privileges to access maps. If no user is set, a proxy server has to be set. The proxy server has to add the user to the URLS's for the server. See more where the proxy server is described.

Example
```
map.setUser("test","test");
```

Method `setLanguage(String language)`

Description Sets the language of the map. For example for the tooltips. Valid value: EN,DE,FR,IT.

Example
```
map.setLanguage("DE");
```

Method `String getLanguage()`

Description Delivers the language of the map.

Example
```
var language = map.getLanguage();
```

Method `setTileSize(Integer tilesize)`

Description Sets the size of the tiles of the map.

Example
```
map.setTileSize(300);
```

Method `Integer getTileSize()`

Description Delivers the size of the tiles of the map.

Example
```
var tilesize = map.getTileSize();
```

Method `setView(String view)`

Description Displays a GeoShop view in the map component. Without `setView()` the default view of the user is displayed. The user needs to have enough privileges to access to the GeoShop view.

Example
```
map.setView("av");
```

Method `String getView()`

Description Delivers the `view` of the map.

Example
```
var view = map.getView();
```

Method `setZoomFactor(Float zoomfactor)`

Description Sets the zoom factor for zoom in and zoom out. A zoom factor of 2.0 doubles the range of the map by a zoom out. A zoom factor of 2.0 halves the range of the map by a zoom in. If a zoom control object of type GSMapZoomSliderControl is set, the zoom factor is given by the the zoom control object.

Example
```
map.setZoomFactor(2.0);
```

Method `Float getZoomFactor()`

Description Delivers the zoom factor of the map.

Example
```
var factor = map.getZoomFactor();
```

Method `setZoomMoveFactor(Float zoommovefactor)`

Description Sets the zoom move factor for zoom up, down, left and right of the map. A zoom factor of 0.25 moves the range of the map by a factor of 0.25 of the height or the width.

Example
```
map.setZoomMoveFactor(0.25);
```

Method `Float getZoomMoveFactor()`

Description Delivers the zoom move factor of the map

Example
```
var factor = map.getZoomMoveFactor();
```

Method `setResolution(Float resolution)`

Description Sets the resolution of the map. The resolution determines the size of the displayed area. Resolution is specified in meter / pixel. Larger values display larger areas, whereas smaller values display smaller areas. If a zoom control object of type GSMapZoomSliderControl is set, the resolution is adjusted to a valid resolution of the zoom control object.

Example
```
map.setResolution(0.1);
```

Method `Float getResolution()`

Description Delivers the actual resolution of the map.

Example
```
var r = map.getResolution();
```

Method `setCenter(Float x, Float y)`

Description Centers the map view around a given x/y coordinate. This function causes to update the data in the the map.

Example
```
map.setCenter(600000.0,20000.0);
```

Method `Float getCenterX()`

Description Delivers the actual center x coordinate of the map.

Example
```
var x = map.getCenterX();
```

Method `Float getCenterY()`

Description Delivers the actual center y coordinate of the map.

Example
```
var y = map.getCenterY();
```

Method `setRange(Float minx, Float miny, Float maxx, Float maxy)`

Description Positions the map between minx/miny and maxx/maxy. This function causes to update the data in the the map.

Example
```
map.setRange(600000.0,200000,650000.0,250000.0);
```

Method `Float getRangeMinX()`

Description Delivers the X-coordinate of the left side of the map.

Example
```
var minx=map.getRangeMinX();
```

Method `Float getRangeMinY()`

Description Delivers the Y-coordinate of the lower side of the map.

Example
```
var miny=map.getRangeMinY();
```

Method `Float getRangeMaxX()`

Description Delivers the X-coordinate of the right side of the map.

Example
```
var maxx=map.getRangeMaxX();
```

Method `Float getRangeMaxY()`

Description Delivers the Y-coordinate of the upper side of the map.

Example
```
var maxy=map.getRangeMaxY();
```

Method `zoomUp()`

Description Moves the range of the map up by the zoom move factor. This function causes to update the data in the the map.

Example
```
map.zoomUp();
```

Method `zoomDown()`

Description Moves the range of the map down by the zoom move factor. This function causes to update the data in the the map.

Example
```
map.zoomDown();
```

Method `zoomLeft()`

Description Moves the range of the map left by the zoom move factor. This function causes to update the data in the the map.

Example
```
map.zoomLeft();
```

Method `zoomRight()`

Description Moves the range of the map right by the zoom move factor. This function causes to update the data in the the map.

Example
```
map.zoomRight();
```

Method `zoomInCenter()`

Description Zooms in at the center of the map by the zoom factor. This function causes to update the data in the the map.

Example
```
map.zoomInCenter();
```

Method `zoomOutCenter()`

Description Zooms out at the center of the map by the zoom factor. This function causes to update the data in the the map.

Example
```
map.zoomOutCenter();
```

Method `Float update()`

Description Updates the data in the map with the actual settings. Can be used if properties are changed which do not update the map it selves.

Example
```
map.update();
```

Method	`addZoomControl(GSMapZoomSliderControl zoomcontrol)`
Description	Adds a zoom control object to the map object (i.e. zoom in, zoom out, etc.).
Example	`map.addZoomControl(new GSMapZoomSliderControl());`

Method	`Boolean sendQuery(String query,String arguments)`
Description	Sends a `query` to the GeoShop server. The arguments have to be submitted in the same format as in WebClient function `image2`, i.e. `"gemeinde=Bern&nummer=500"`. Found objects are displayed in center of the GSMap object. If objects are found then the function returns true, else the function returns false.
Example	`var success = map.sendQuery("parzelle","gemeinde=Bern&nummer=500");`

Method	`resetQuery()`
Description	Resets the query result. The highlighted object is not highlighted anymore.
Example	`map.resetQuery();`

3. GSMapZoomSliderControl Object

3.1. GSMapZoomSliderControl Methods

Constructor	**`GSMapZoomSliderControl()`**
Description	Creates a new `GSMapZoomSliderControl` object. The `GSMapZoomSliderControl` is represented on the map like the zoom control in Google maps.
Example	`var zctrl = new GSMapZoomControl();`

Method	**`setSliderResolution(Float minresolution, Float maxresolution, int steps, int actstep)`**
Description	Sets the resolution range of the zoom slider. Sets the steps between minresolution and maxresolution and the actual step to display. Resolution values are specified in meter / pixel units.
Example	`zctrl.setSliederResolution(1.0,1000.0, 20, 10);`

3.2. Icons

The Object `GSMapZoomSliderControl` uses icons for the user interface. Self designed icons can be used.

The icons have to be created. The url for the icons has to be set with the method `GSMap.setIconUrl(<url>)` . The Extension for the icons has to be set with the method `GSMap.setIconExtension(<extension>)` .

The used Icons are:

Icon name	**`slidercontrol.gif`**
Size	54x54
Description	Is the Icon with the zoom left, right, top, down and back elements. The size of the icons is divided into a 3x3 matrix. The fields of the matrix are used like: [0,1]=zoom up, [1,0]=zoom left, [1,1]=zoom back, [1,2]=zoom right, [2,1]=zoom down.

Icon name	**`sliderzoomin.gif`**
Size	24x24
Description	Is the Icon for the zoom in button.

Icon name	**`sliderzoomout.gif`**
Size	24x24
Description	Is the Icon for the zoom out button.

Icon name	**`sliderbar.gif`**
Size	12x272
Description	Is the Icon for slider bar.

Icon name	**`sliderpoint.gif`**

Size	6x6
Description	Is the Icon for the points on the slider bar which indicates the steps on the slider bar.

Icon name	`sliderpusher.gif`
Size	18x12
Description	Is the Icon for the slider pusher, which can be moved on the slider bar to changed the zoom factor.

4. GSMapTools Object

4.1. GSMapTools Methods

Constructor `GSMapTools()`

Description Creates a new `GSMapTools` object. The `GSMapTools` object is represented in the map as a set of icon buttons (info button, selection button).

Example
```
var tools = new GSMapTools();
```

Method `addInfoTool()`

Description Creates an info tool in the tool box. After pressing the InfoTool button, objects are selected by clicking on the map. The InfoTool button changes color to indicate, that the Map is in selection mode. Selected objects are highlighted in the map.When info is finished the user has to click on the infoTool button again. The queried results are sent as an array of objects to the info tool handler function for further processing. Each new selection clears the existing array of objects and adds the new objects. Note: Not needed by EGRIS application.

Example
```
tools.addInfoTool();
```

Method `setInfoToolHandler(Function myfunction)`

Description Sets the info tool handler function. This function is a user function which handles the selected objects. The function is called each time an object is selected. The function receives the objects as an array of objects. Each object contains the attributes of the object.

Example
```
tools.setInfoToolHandler(myfunction);
:
function myfunction (objects)
{
   if (objects == null) return;

   alert("Objects found:" + objects.length);

   for (i=0;i<objects.length;i++) {
      o = objects[i];

      message = 'Object '+ (i+1) + ':\n'
      for (a in o) message = message + a + '=' + o[a] + '\n';

      alert(message);
   }
}
```

Method `setInfoToolClasses(String myclasses)`

Description Sets the info tool classes. Only objects of the classes are returned by a selection. The classes must be available in the actual view of the map. To define a single class, call the function with a single class. To define multiple classes, call the function with the classes separated by commas. To define all classes, call the function with "*" .

Example
```
tools.setInfoToolClasses("Abwasser,Wasser");
```

Method	**`addSelectionTool()`**
Description	Creates a selection tool for a given layer in the tool box. After pressing the SelectionTool button, objects are selected by clicking on the map. The SelectionTool button changes color to indicate, that the Map is in selection mode. Selected objects are highlighted in the map. When selection is finished the user has to click on the SelectionTool button again. Each time the user selects an object, all selected objects are sent as an array of objects to the selection tool handler function for further processing.
Example	`tools.addSelectionTool();`

Method	**`setSelectionToolHandler(Function myfunction)`**
Description	Sets the selection tool handler function. This function is a user function which handles the selected objects. The function is called each time an object is selected. The function receives the objects as an array of objects. Each object contains the attributes of the object.

Example	`tools.setInfoToolHandler(myfunction);`
	`:`
	`function myfunction (objects)`
	`{`
	`  // see also setInfoToolHandler`
	`}`

Method	**`setSelectionToolClasses(String myclasses)`**
Description	Sets the selection tool classes. Only objects of the classes are returned by a selection. The classes must be available in the actual view of the map. To define a single class, call the function with a single class. To define multiple classes, call the function with the classes separated by commas. To define all classes, call the function with `"*"` .
Example	`tools.setSelectionToolClasses("Parzelle");`

4.2. Icons

The Object `GSMapTools` uses icons for the user interface. Self designed icons can be used.

The icons have to be created. The url for the icons has to be set with the method `GSMap.setIconUrl(<url>)` . The Extension for the icons has to be set with the method `GSMap.setIconExtension(<extension>)` .

The used Icons are:

Icon name	**`info1.gif, info2.gif`**
Size	24x24
Description	Is the Icon for the Info Button. Not selected state: `info1.gif` . Selected Sate: `info2.gif` .

Icon name	**`selection1.gif, selection2.gif`**
Size	24x24
Description	Is the Icon for the Selection Button. Not selected state: `selection1.gif` . Selected Sate: `selection2.gif` .

5. Other Methods

Constructor	`GSBrowserCheck()`
Description	Tests if the browser is compatible with the GSAPI.
Example	

```
if (!GSBrowserCheck()) {
   return;
}
```

6. Languages

Messages and texts (i.e. tool tips) depending a language can be set in the following files.

Language EN	`<GSAPIDIR>/GSLangEN.js`
Language DE	`<GSAPIDIR>/GSLangDE.js`
Language FR	`<GSAPIDIR>/GSLangFR.js`
Language IT	`<GSAPIDIR>/GSLangIT.js`

Each file contains a language hashtable where the code and text has to be inserted. The code is a placeholder for the text.

```
GSLang<language>.put(<code>,<text);
```

Example for language EN in `<GSAPIDIR>/GSLangEN.js` and the code `tooltip.sliderzoomin` , which is the tool tip for the button zoom in of the slider zoom control elements.

```
GSLangEN.put("tooltip.sliderzoomin","Zoom In");
```

7. GeoShop URL's via a Proxy Server

If you want to connect the GeoShop Server (maps and other data) via a Proxy Server to the map, you have to define the URL of the Proxy Server with the method `GSMap.setServer(<Proxy-Url>)`. In the Proxy server you have to redirect the Urls to the GeoShop Server. If you do not set a GeoShop User with `GSMap.setUser(<user>,<password>)`then you have to add the user/password in the Proxy Server.

Proxy incoming URL	`<Proxy-Url>/webclient?<arguments>`
Proxy outgoing URL	`<GeoShop-url>/webclient?<arguments>[&user=<user>&password=<password>].`
Description URL	Redirecting `<Proxy-Url>` to `<GeoShop-Url>`
Description User	Adding arguments `&user=<user>&password=<password>` if not already there.