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

GeoShop Mapping API

Zusammenfassung

This documentation describes the GeoShop Mapping API version 2.0 (GSAPI).

Inhaltsverzeichnis

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 maps or send data orders from his application. The GSAPI consists of the following main objects:

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

GSUser
: The GSUser object can be queried to get information about GeoShop user properties (preferences, products, views).

GSOrder
: Used for GeoShop order processing.

GSHashtable
: A key/value data structure to pass information to / from GS* objects.

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

1.2. Supplementary documentation

[1] GeoShop Server Konfigurationshandbuch (in German).

[2] GeoShop Batch Client (in German).

1.3. 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. Example HTML 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>

    <script type="text/JavaScript">
(3)    function GSAPIInitializedCallback() {

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

          // create user object GSUser
          var user = new GSUser();

          user.login(<USER>,<PASSWORD>, function() {
                // create map object GSMap
                var map = new GSMap("map");

                // set user for authentification of GSMap functions
                map.setUser(user);

                // 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>);
          }, function(errorMessage) {
             alert(errorMessage);
          });
       }
    </script>
(1) <script type="text/javascript" src="<SERVER-URL>/gsapi2/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>
    </head>
    <body>
(4)    <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. The JavaScript-function `GSAPIInitializedCallback()` is called after the necessary Javascript-Files for the GSAPI are loaded. In this example it creates the Map object and initializes the GUI controls.
4. The HTML `DIV` element determines the size and position of the GSMap object in the HTML page.

2.2. 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.setIconExtension("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.

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 `setTileBorder(Boolean status)`

Description Displays a border around each tile of the map. Can be used to see the tiles to find the best size of the tiles.

Example
```
map.setTileBorder(true);
```

Method `Boolean getTileBorder()`

Description Delivers the tile border setting.

Example
```
var border = map.getTileBorder();
```

Method `setTileBackgroundColor(String color)`

Description Sets the background color of the tiles while the image is loaded and is not displayed already. The value of the color must be set as a RGB-HTML hexadecimal value like #RRGGBB. RR=red, GG=green,BB=blue, i.e. #FF0000 for the color red.

Example
```
map.setTileBackgroundColor("#FFFFFF");
```

Method `String getTileBackgroundColor()`

Description Delivers the background color of the tiles.

Example
```
var color = map.getTileBackgroundColor();
```

Method `setTileResetMode(Boolean status)`

Description Sets the tile reset mode. By default this mode is activ. If this mode is activ, for each command zoom in, zoom out, query, selection or info the raster of the tiles for the view is reseted. This mode makes only sense if the size of

the tiles is bigger than the size of the map. In this case this mode causes that only one image has to be loaded. If the size of the tiles is smaller than the size of the map this mode should be deactivated.

Example
```
map.setTileResetMode(false);
```

Method `Boolean getTileResetMode()`

Description Delivers if the tile reset mode is activated or not activated.

Example
```
var mode = map.getTileResetMode();
```

Method `setPanAccelerationMode(Boolean status)`

Description Sets the pan acceleration mode. By default this mode is not activ. If this mode is activ, if you pan - mouse down and move - and the mouse goes up while moving, the map keeps moving for a while.

Example
```
map.setPanAccelerationMode(true);
```

Method `Boolean getPanAccelerationMode()`

Description Delivers if the tile reset mode is activated or not activated.

Example
```
var mode = map.getPanAccelerationMode();
```

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 `sendQuery(String query,String arguments, function successCallback(Boolean), function errorCallback(String))`

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"`. If objects are found `successCallback()` is called asynchronously with `true` else with `false`. Found objects are displayed in center of the GSMap object.

Example

```
map.sendQuery("parzelle","gemeinde=Bern&nummer=500", function(found) {
      if (found) {
         // object found
      }
      else {
         // no object found
      };
   }, function(errorMessage) {
      // some error occured
   }
);
```

Method `resetQuery()`

Description Resets the query result. The highlighted object is not highlighted anymore.

Example

```
map.resetQuery();
```

Method `placeSelectionBox()`

Description Starts the placement of a selection box. The user can digitize the box by clicking into the map (2 points).

Example

```
map.placeSelectionBox();
```

Method `placeSelectionPolygon()`

Description Starts the placement of a selection polygon. The user can digitize the polygon by clicking into the map. A double click closes the polygon.

Example

```
map.placeSelectionPolygon();
```

Method `GSPolygon getSelectionPolygon()`

Description Returns the recently digitized selection polygon or null if no selection polygon exists. The selection polygon can be used as GSOrder parameter.

Example

```
var selection = map.getSelectionPolygon();
```

Method `clearSelectionPolygon();`

Description Clears the selection polygon.

Example

```
map.clearSelectionPolygon();
```

Method `getInfo(Float x, Float y, String classes, function successCallback(Array), function errorCallback(String))`

Description Returns asynchronously an array of objects in `successCallback()` for the point x, y and the selected classes. See GSMapTools.setInfoToolClasses how to use the argument classes. See GSMapTools.setInfoToolHandler how to process the returned array of objects.

Example

```
map.getInfo(675500.000,245500.000,"*", function(objects) {
      // display object information
   }, function(errorMessage) {
      // some error occured
   }
);
```

Method `setInfoSize(Float size)`

Description By default the method getInfo returns only objects which are within the range of the actual map and where Information is available for the selected size of the map . That means x, y must be within getRangeMinX/Y() and getRangeMaxX/Y() and the Information must be available for this size . With this method the size can be set by the size for a side of a square and the point x,y for the method getInfo must not be within an actual map.

Example

```
map.setInfoSize(100.0);
```

Method `resetInfo()`

Description Resets the getInfo method so that no objects are selected and the method setInfoSize is reset to the default behaviour.

Example

```
map.resetInfo();
```

Method `getSelection(Float x, Float y, String classes, function successCallback(Array), function errorCallback(String))`

Description Returns asynchronously an array of objects in `successCallback()` for the point x, y and the selected classes. See GSMapTools.setInfoToolClasses how to use the argument classes. See GSMapTools.setInfoToolHandler how to process the returned array of objects. If a new selection should be started, the method resetSelection() should be called first to reset the array of objects. Repeating calls of getSelection with other points x.y add the new found objects to the existing array of objetcs. If an already selected object is selected by another call of the method again, the object is removed (deselected) from the array of objects.

Example

```
map.getSelection(675500.000,245500.000,"*", function(objects) {
      // do something with objects
   }, function(errorMessage) {
      // some error occured
   }
);
```

Method `setSelectionSize(Float distance)`

Description By default the method getSelection returns only objects which are within the range of the actual map and where Information is available for the selected size of the map . That means x, y must be within getRangeMinX/Y() and getRangeMaxX/Y() and the Information must be available for this size . With this method the size can be set by the size for a side of a square and the point x,y for the method getSelection must not be within an actual map.

Example

```
map.setSelectionSize(100.0);
```

Method `setSelectionDeselection(Boolean status)`

Description	The possibility of a deselection of selected objects can be switched off by the call of map.setSelectionDeselection(false). A once selected objects stays selected.
Example	`map.setSelectionDeselection(false);`

Method	**`resetSelection()`**
Description	Resets the getSelection method so that no objects are selected and the method setSelectionSize is reset to the default behaviour.
Example	`map.resetSelection();`

2.3. 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 GSMapZoomSliderControl();`

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.setSliderResolution(1.0,1000.0, 20, 10);`

Method	**`setZoomDoubleClicktoCenter(Boolean status)`**
Description	If `true` a double click of left/right button sets the point of the mouse to the center and zooms in/out. If `false` a double click of left/right button zooms in/out around the point of the mouse. Default is `false` .
Example	`zctrl.setZoomDoubleClicktoCenter(true)`

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

2.4. 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.setSelectionToolHandler(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");
```

The Object `GSMapTools` uses icons for the user interface. Self designed icons can be used. 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>)`. Possible icons are:

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

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

3. GSUser Object

3.1. Introduction

The GSUser object represents a GeoShop user. GSUser objects have to be initialized by the `login()` method.

3.2. GSUser Methods

Constructor	**`GSUser()`**
Description	Creates a new GSUser object.
Example	`var gsuser = new GSUser();`

Method	**`login(String user,String password, function successCallback(), function errorCallback(String))`**
Description	Initializes the GSUser object. All other methods of GSUser can only be used after `successCallback()` is called. Calls `successCallback()`asynchronously if the user exists, calls `errorCallback()` asynchronously if the user does not exist or the password is wrong.
Example	

```
gsuser.login("test","test", function() {
      // successful login
   }, function(errorMessage) {
      // invalid user / password
   }
);
```

Method	**`GSHashtable getProducts()`**
Description	Delivers a GSHashtable with (name / display_name) entries of all available data products for the user.
Example	`var products = gsuser.getProducts();`

Method	**`getProductByName(String name, function successCallback(GSHashtable), function errorCallback(String))`**
Description	Calls `successCallback()`asynchronously with a detailed description of the product by name. Calls `errorCallback()` asynchronously if the product is not found or some other error occurs.
Example	

```
gsuser.getProductByName("dxf_geobau", function(product) {
      // do somthing with product
   }, function(errorMessage) {
      // some error occured
   }
);
```

Method	**`getProductByDisplayName(String display_name, function successCallback(GSHashtable), function errorCallback(String))`**
Description	Variant of getProductByName(). Delivers a detailed description of the product by display_name.

Example

```
gsuser.getProductByDisplayName("DXF GeoBau", function(product) {
      // do something with product
   }, function(errorMessage) {
      // display an error message
   });
```

Method `GSHashtable getPreferences()`

Description Delivers a all user preferences as a GSHashtable.

Example

```
var preferences = gsuser.getPreferences();
```

4. GSOrder Object

4.1. Introduction

The GSOrder object is used for GeoShop order processing.

4.2. GSOrder Methods

Constructor	**`GSOrder()`**
Description	Creates a new GSOrder object.
Example	`var gsorder = new GSOrder();`

Method	**`setUser(GSUser user)`**
Description	Sets the user authentication for sendOrder().
Example	`gsorder.setUser(gsuser);`

Method	**`setProduct(String product)`**
Description	Selects the product to be ordered by name.
Example	`gsorder.setProduct("dxf_geobau");`

Method	**`setParameterModelByProduct(String product)`**
Description	Sets the "model" parameter by product name.
Example	`gsorder.setParameterModelByProduct("dxf_geobau");`

Method	**`setParameter(String parameter,String value)`**
Description	Sets a product parameter value. Possible parameter names depend on the product definition.
Example	

```
gsorder.setParameter("name1","infoGrips GmbH");
gsorder.setParameter("adr1","Technoparkstrasse 1");
gsorder.setParameter("zip","8005");
gsorder.setParameter("city","Zürich");
gsorder.setParaemter("email","info@infogrips.ch");
gsorder.setParameter("selection_polygon",gsmap.getSelectionPolygon());
```

Method	**`sendOrder(function successCallback(GSHashtable), function errorCallback(String))`**
Description	Sends the order to the GeoShop server. If the order is successfully send to the server, `successCallback()` is called asynchronously, else `errorCallback()`.
Example	

```
gsorder.sendOrder(function(order) {
      // order was submitted successfully
   }, function(errorMessage) {
      // some error occured
   }
);
```

Method	**`String getInfoMessage()`**
Description	Gets some information about the state of the last `sendOrder()` as String.
Example	`alert(gsorder.getInfoMessage());`

More information on available product parameters can be found in the separate "GeoShop Batch Client" documentation.

5. Other Objects and Methods

5.1. GSHashtable Methods

Constructor	`GSHashtable()`
Description	Creates a new GSHashtable object.
Example	`var gshash = new GSHashtable();`

Method	`put(String key,Object value)`
Description	Stores a key / value pair in the GSHashtable.
Example	`gshash.put("city","Zürich");`

Method	`Object get(String key)`
Description	Delivers the value identified by key. If the key does not exist, get() returns null.

Example	`var city = gshash.get("city");`
	`if (city == null) {`
	`  // there is no key "city"`
	`}`

Method	`Array keys()`
Description	Delivers all keys of the GSHashtable as an Array.

Example	`for (k in gshash.keys()) {`
	`  // process values`
	`  value = ghash.get(k);`
	`}`

5.2. GSPolygon Methods

Constructor	`GSPolygon()`
Description	Creates a new GSPolygon object.
Example	`var gspolygon = new GSPolygon();`

Method	`add(GSPoint p)`
Description	Adds a GSPoint to the polygon.
Example	`gspolygon.add(new GSPoint(600000.000,200000.000));`

5.3. Utility 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");
```