Position s.r.o.
Londýnská 665/45, 120 00 Praha 2
http://www.position.cz

WMS POSITION - ROUTING

WMS POSITION – ROUTING CHARACTERISTICS 4.10.2010

Documentation version 2012_1 EN

Position provides WMS Position – Routing service in addition to mapping services Web Map Service (WMS) which allows using of map data distributed using Internet in end users' mapping and GIS applications and other services WMS Position family. WMS Position – Routing provides routing functions for optimization and route planning.

The service can be called by SOAP or by HTTP GET method with parameters included in URL.
If HTTP GET is used, there are two variants of the output:

- XML
- JSON

The Service security is guaranteed by communication via secured channel SSL - certification THAWTE.
The tradition and quality of THAWTE certification allowes full client´s confidence.

The access security is resolved by Username and Password.
The access is possible by classical access parameters or simply by specification in URL address:

A) **https://routing.position.cz/?cmd=ZZZZ&auser=XXXX&apass=YYYY**
B) **https://routing.position.cz/XXXX/YYYY/?cmd=ZZZZ** (name/password/function name)

Following functions are available to work with the service.

- **Routing (places, params)**
  It calculates route. The output is selectively controlled by params object. A client fills information to get. The service returns only selected data (summary, itinerary, geometries).
- **DeleteSession (SID, routing_id)**
  It deletes data from a session. The session can include more routes with different identifiers.
- **DestroySession (SID)**
  It removes all session data. The session dies.

SOAP INTERFACE DESCRIPTION

function **Routing**(places, params)

Input:

places = an array of names where the planed route passes. It contains objects of RoutingPlace type. Their structure:

{

coords: string; coordinates in text format, separated by semicolon, a name of projection can be included at first place and separated by semicolon, example: JTSK;5587266.64;3380924.03. If a name of projection is not set and the parameter Projection (params argument) is set, it suspected the point is in that projection. Or coords property can include place name. The service tries to find the place in database of addresses. Coordinates are filled automatically if the place is found. If there is more than one result, the first one is applied.

descr: string; optional name or description of location, a new name predefinition

}

params = object contains parameters of route planner. Structure:

{

ReqID: a string; optional user parameter, it passes unchanged to output. It can be used to identify client's request and response.

Sums: boolean; optional parameter, if it is true, the function returns summaries: Total time, total distance of calculated route in result Sums object.

Itineraire: boolean; optional parameter, if it is true, the itinerary is calculated and returned. There are two types of itinerary: basic and detailed. It can be affected by ItineraireType value.

ItineraireType: string; optional parameter. There are two types of itinerary: basic and detailed. Possible values (case insensitive):

0, "basic", "basic view" basic itinerary

1, "detail", "detailed", "detailed view" detailed itinerary

The default value is "Basic view". If Itineraire is not true, ItineraireType value is ignored.

Line: string; optional parameter, it defines if route geometry is calculated and returned. Possible values for Line parameter:

c, client - route geometry is returned to client, no session storing

s, server – stored to the session, no return to client

cs, clientserver – returned to client and stored in session

none – no geometry calculation, equal to empty string

Notice: The geometry is stored in session if Session parameter is set true.

Session: boolean; if true, route planer status is stored in session.

SID: string; session identifier, It identifies the session to read status and to store route planner's status.

RoutingID: string; It includes route identifier. It identifies route identifier to store route planner's status in the session. The service can store more statuses in one session. The statuses are identified by this value. Optional parameter.

RoutingMode: string; route calculating mode – shortest / fastest

Values:

0, "f", "fastest" – the fastest route

1, "s", "shortest" – the shortest route

If the value contains an empty string or an unknown value, currently used mode depends on previous mode. The value can be read from session (if SID is known) or it can be randomly set by previous of other client (!)

Projection: string; name of input projection for coordinates where name of projection is missing. Optional parameter.

OutputProjection: string; name of output projection, optional parameter

Places: boolean; if true, the output contains input locations, optional parameter

User: string; user name for authentication

Password: string; password for authentication

}

Output:

Structure of output object:

{

ReqID: string; value from input parameter ReqID, if it is sent

Session: boolean; equal to input Session parameter.

SID: string; session identifier, if used

RoutingMode: string; routing mode, the route is calculated in this mode. Possible values: "**fastest**", "**shortest**".

RoutingID: string; route identifier for storing and reading from session. It is same as input RoutingID parameter. If missing, the input value is void.

Error: number; error number, if 0, result is OK.

ErrorMessage: string; error message, if success, empty string.

Sums: object; route summary, it contains summary data about route length and time. The value is returned, if Sums property is set true.

Structure of Sum object:

{

TotalTime: string; total time of route

TotalLength: string; total length of route

}

Itineraire: object; It contains itinerary, if it is required in Itineraire parameter. Object structure:

{

TotalTime: string; total time

TotalLength: string; total length of route

Items: array of objects ItineraryItem type, }

Structure of ItineraireItem object:

{

descr: string; name or description of location

km: string; distance from begin of route in kilometers

length: string; length of section to next point in itinerary, in kilometers

duration: string; time from begin of the route

routeNum: string; road name

coords: object; location coordinates in CoordsObj object

}

CoordsObj contains name of projection and parts of coordinates

{
proj: string; projection name. Next properties are optional. There are only properties in according to selected projection: x,y,lat,lon,z,e,n,c1,c2,c3,c4. There are reserved properties c1-c4 for projection including next parts of coordinates.
}

ItineraireType: string; itinerary type, possible values:
"**Basic view**" = basic itinerary
"**Detailed view**" = detailed itinerary

Line: string; geometry of drawn line in WKB format, binary data are encoded to Base64. It is included, if Line parameter is equal to "c" or "cs".

LineProjection: string; name of projection, of Line's coordinates.

Places: array of objects; it is included, if input parameter Places is set true. Places array contains objects in following structure:

{
coords: object; location coordinates. Type CoordsObj, see above.
descr: string; name or description of location
searchresult: number; optional parameter. It is number of found locations according to input. If 0, no place found, error value is -2. The correct result is 1 only. If more than one location is found, the input must be better specified.
}

}

funkce **DeleteSession**(SID, routing_id)
input:
SID = id session
routing_id = id, pod kterým byly uloženy výsledky plánování trasy
output:
boolean; **true**, if planer's data with routing_id are successfully deleted. It returns also true, if required data are not included in session. It returns **false**, if an error has occurred (attempt to delete strange session).

funkce **DestroySession**(SID)
input:
SID = id session
output:
boolean:
true, if session is successfully deleted or session doesn't exist.
false, if an error has occurred (attempt to delete strange session).

Position s.r.o.
Londýnská 665/45, 120 00 Praha 2
http://www.position.cz

HTTP INTERFACE DESCRIPTION

HTTP Interface WMS POSITION - Routing Service allows you to make a simple manner with defined parameters passed as part of the URL. The service provides for this type of call two types of output: XML or JSON. The functions are identical to the version of SOAP, but the transmission parameters are adapted for use in the URL. Output in XML and JSON defines all the property names in lower case, and adds some "wrapping" property, which is a slight difference compared to the SOAP version.

Secure access to the HTTP interface is solved by using the user name and password. Access is possible using usual access parameters:

https://routing.position.cz/?cmd=r&output=xml&pt0=Praha&pt1=Prelouc&i=1&rt=f&auser=user&apass=password

Description of URL input parameters

Routing function:

cmd – name of required command.
Options:
"r" or "routing" – calling Routing function, calculating of route
"d" or "deletesession" - calling DeleteSession function
"ds" or "destroysession" - calling DestroySession function

output – type of output, optional parameter, default value = xml
Options:
xml – output to xml file
js – JSON output

Routing function parameters

reqid – same as ReeqId in SOAP, it passes to output with no change.

pt0, pt1, ..., ptN – tracking points. pt0 is starting point. ptN is ending point. N is number. Location can be defined by coordinates or by name, see SOAP version above. Coordinates and projection name are separated by semicolon. Location description can be added with "@" prefix.

Examples: 49.436076;16.575666@MyStart
(WGS coordinates, title "MyStart")
JTSK;5434872.00;3572972.00
(JTSK coordinates, no user title)
Wien@Austrian%20captial%20city
{Wien with user title "Austrian capital city")

pl - alternative input of tracking points in one string. If this value is set, pt0 .. ptN values are ignored. Each point is separated by "|" character. Coordinates or name are separated by "@" characters. It is same as pt0..ptN parameter's syntax.

proj – name of input projection, optional parameter (default value = WGS-84). This parameter is used for all input coordinates where projection name is missing in proj section.

outproj – name of output projection, optional parameter (default value = WGS-84). It specifies output projection to recalculate output coordinates and drawing lines.

s – like Sums, values 0, 1 (default 0). It decides, if summary is included in output.

i – like Itinerary, values 0, 1 (default 0). It decides, if itinerary is included in output.

it – itinerary type, see above: SOAP - input ItineraireType

l - like Line, see above: SOAP

ses – similar to Session in SOAP version, values 0, 1 (default 0).

SID - see SID see above: SOAP version, session id

rid - see RoutingID

rt - see RoutingMode

rpl – see Places

auser - see User

apass - see Password

enc – URL parameters encoding, UTF-8 is standard, two bytes encoding.

if enc=ng, URL contains UTF8 encoding. ASCII-8 characters are encoded as 1 byte. Characters above ASCII-8 are encoded in unicode format (%u0161 = "á")

callback – only for JS output, name of callback function. The service calls in JavaScript output this function and gives it object with result.

JSON output of Routing function

Output JSON object in comparison with SOAP has defined object request and object response. Object response is in according to SOAP result. There is only one difference. Names of properties are in low case.

```
{
  "request": {
    "cmd": command (function: routing/deletesession/destroysession)
  },
  "response": {
        Object response is in according to SOAP result. There is only one difference. Names of properties are in low case.
  }
}
```

XML output of Routing function

Output XML object in comparison with SOAP has defined object request and object response. Object response is in according to SOAP result. There is only one difference. Names of properties are in low case.
XML structure:

```
<routingresult>
  <request cmd="routing"/>
  <response>
    <!--
        Elements with similar structure with SOAP version follow here. There is one exception for array. Each item has to be placed into an element. Each itinerary item is in <item> element.
      ...<response>
         <itineraire>
          <items>
            <item>...</item>
          </items>
         </itineraire>
         ...
      </response>
    -->
  </response>
</routingresult>
```

Parameters of DeleteSession parameters:

cmd=d
reqid
SID
rid

auser
apass
callback
enc

JSON output of Delete session function:

```
{
 "request": {
  "cmd": "deletesession"
 },
 "response": {
  "result": true,
  "sid": "naqpfostf8874v9u611drf9ud5",
  "routingid": "myrtid",
  "reqid": "myreqid"
 }
}
```

XML output of DeleteSession function:

```
<routingresult>
 <request cmd="deletesession"/>
 <response>
  <result>1</result>
  <sid>naqpfostf8874v9u611drf9ud5</sid>
  <routingid>as</routingid>
  <reqid>mojereqid</reqid>
 </response>
</routingresult>
```

Parameters of DestroySession parameters:

cmd=ds
reqid
SID - session id
auser
apass
callback
enc

JSON output of DestroySession function:

```
{
 "request": {
  "cmd": "destroysession"
 },
 "response": {
  "result": true,
```


"sid": "naqpfostf8874v9u611drf9ud5",
"reqid": "myreqid"
}
}

XML output of DestroySession function:

<routingresult>
<request cmd="destroysession"/>
<response>
<result>1</result>
<sid>naqpfostf8874v9u611drf9ud5</sid>
<reqid>myreqid</reqid>
</response>
</routingresult>

The result is stored in “result” element/property in both formats (XML/JSON). If the result is successful value is set true. In case of error, the value is set false. An error code is not sent.

Position s.r.o.
Londýnská 665/45, 120 00 Praha 2
http://www.position.cz

LIST OF ERROR CODES

The service returns error number in Error property and error text in ErrorMessage property.
If result is successful, Error property is set 0 and Error message contains an empty string.

0	OK
-1	unknown command in cmd parameter
-2	Any place has not been found. If Places parameter is equal true than this place has value searchresult=0 in output array Places. Value of searchresult property can be greater than one more places have been found in according to input. The input must be better specified. The service does not return an error. It uses the first of found places.
-3	wrong coordinates, If Places parameter is equal true than this place has coords property set false in output array.
-4	Authentication failed. User name or password is incorrect. Or user has insufficient permissions.
-5	wrong name of projection (input)
-6	Neither input locations are set or they cannot be read from session.
-10	internal service error

Position s.r.o.
Londýnská 665/45, 120 00 Praha 2
http://www.position.cz

EXAMPLE

Route: Rozehnaly – Lipec

https://routing.position.cz/?cmd=r&output=js&pt0=Rozehnaly&pt1=Lipec&i=1&rt=f&auser=user&apass=password

Output in JSON:


```
{
 "request":{
  "cmd":"routing"
 },
 "response":{
  "session":false,
  "routingmode":"fastest",
  "itinerairetype":"Basic view",
  "itineraire":{
   "totaltime":"00:07:50",
   "totallength":"4.4 km",
   "items":[{
    "descr":"Rozehnaly (okres Kol\u00edn)",
    "km":"",
    "length":"",
    "duration":"",
    "routenum":"",
    "coords":{
     "proj":"WGS-84",
     "lat":50.11022,
     "lon":15.38747},
    "proj":"WGS-84"},{
    "descr":"CZ, Radovesnice II",
    "km":"0,0 km",
    "length":"1,9 km",
    "duration":" 00:00:00",
    "routenum":"32718",
    "coords":{
     "proj":"WGS-84",
     "lat":50.1102198,
     "lon":15.3874704},
    "proj":"WGS-84"},{
    "descr":"CZ, Radovesnice II",
    "km":"1,9 km",
    "length":"2,5 km",
    "duration":" 00:03:20",
    "routenum":"32711",
    "coords":{
     "proj":"WGS-84",
     "lat":50.1050761,
     "lon":15.3681563},
    "proj":"WGS-84"},{
    "descr":"Lipec (okres Kol\u00edn)",
    "km":"4,4 km",
```



```
    "length":"",
    "duration":"00:07:50",
    "routenum":"",
    "coords":{
     "proj":"WGS-84",
     "lat":50.08447,
     "lon":15.36454},
    "proj":"WGS-84"}
   ]},
  "error":0,
  "errormessage":""
 }
}
```


https://routing.position.cz/?cmd=r&output=XML&pt0=Rozehnaly&pt1=Lipec&i=1&rt=f&auser=someuser&apass=somepassword

Output in XML:


```
<routingresult>
	<request cmd="routing"/>
	<response>
		<session>0</session>
		<routingmode>fastest</routingmode>
		<itinerairetype>Basic view</itinerairetype>
		<itineraire>
			<totaltime>00:07:50</totaltime>
			<totallength>4.4 km</totallength>
			<items>
				<item>
					<descr>Rozehnaly (okres Kolín)</descr>
					<km/>
					<length/>
					<duration/>
					<routenum/>
					<coords>
						<proj>WGS-84</proj>
						<lat>50.11022</lat>
						<lon>15.38747</lon>
					</coords>
					<proj>WGS-84</proj>
				</item>
				<item>
					<descr>CZ, Radovesnice II</descr>
					<km>0,0 km</km>
					<length>1,9 km</length>
					<duration> 00:00:00</duration>
					<routenum>32718</routenum>
					<coords>
```



```
                                        <proj>WGS-84</proj>
                                        <lat>50.1102197766</lat>
                                        <lon>15.3874703981</lon>
                                </coords>
                                <proj>WGS-84</proj>
                        </item>
                        <item>
                                <descr>CZ, Radovesnice II</descr>
                                <km>1,9 km</km>
                                <length>2,5 km</length>
                                <duration> 00:03:20</duration>
                                <routenum>32711</routenum>
                                <coords>
                                        <proj>WGS-84</proj>
                                        <lat>50.1050761053</lat>
                                        <lon>15.3681562773</lon>
                                </coords>
                                <proj>WGS-84</proj>
                        </item>
                        <item>
                                <descr>Lipec (okres Kolín)</descr>
                                <km>4,4 km</km>
                                <length/>
                                <duration>00:07:50</duration>
                                <routenum/>
                                <coords>
                                <proj>WGS-84</proj>
                                        <lat>50.08447</lat>
                                        <lon>15.36454</lon>
                                </coords>
                                <proj>WGS-84</proj>
                        </item>
                </items>
        </itineraire>
        <error>0</error>
        <errormessage/>
    </response>
</routingresult>
```


Position s.r.o.
Londýnská 665/45, 120 00 Praha 2
http://www.position.cz

INFORMATION ABOUT POSITION´S SERVICES

WMS Position - MAPS – street level Europe coverage (according NAVTEQ data actual coverage)
WMS Position - SEARCH – finding addresses level Europe coverage (according NAVTEQ data actual coverage)
WMS Position - REVERSE GEOCODING – street level geocoding, address level geocoding possibility, special data possible
WMS Position - ROUTING – street level routing
WMS Position - ROUTE OPTIMIZATION – street level optimization