Geoptimization API

GEOCONCEPT SAS

Version : WINTER-23-1

Date : 1/30/23


Geoptimization API

Introduction

Geoconcept Geoptimization JavaScript API and Geoptimization web services are aimed at developers and integrators, providing a JavaScript library and a complete series of hosted web services that are easily integrated so the geographic dimension can be rapidly added to your applications.

Example

- Search demonstrator [https://en.geoconcept.com/geocoding-demonstrator] showing search web services (autocomplete, geocoding, address standardization) and WMTS map tiling.
- Routing demonstrator [https://en.geoconcept.com/route-api-demonstrator] showing routing (including prohibited transit zones), reverse geocoding web services and WMTS map tiling.

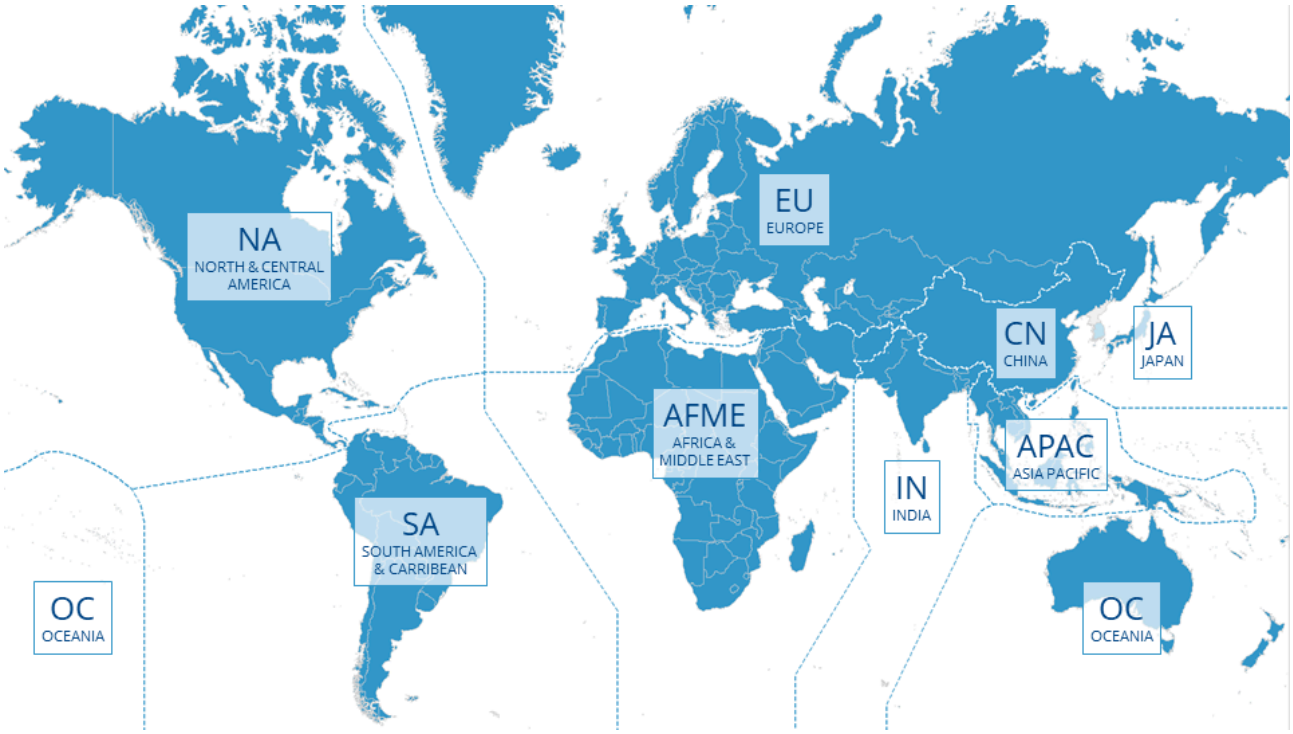
How can I get started?

To use our Geoptimization JavaScript API and Geoptimization web services, simply fill in this form [https://en.geoconcept.com/test-geoptimization-api] and you will receive an email with login identifiers valid for 30 days. You will be able to trial both documentation and components to integrate within your applications.

It goes without saying, during the trial period you will be able to use our services – as described in the documentation - free of charge, without any obligation to provide us with information on your side.

Which coverage?

Coverage may vary depending on the web service, but most are available on any chosen continental server (see below) for the following geographic area:



To request a specific continental server, replace the XX string, inside the url, with the chosen continent:

`https://api.geoconcept.com/XX/GCW/geoconcept-web/api/lbs/`

What new features are available?

Winter 2023 release

- Data updated to 2022 Q4
- isochrones/isodistances updated to v5 with a new parameter: resolution
- compatcMatrix updated to v5 with a new parameter: maxTargets

Fall 2022 release

- geocode new parameter: excludePlaces
- ZFE reject flag added to all routing services (can be used only for France)

Spring 2022 release

- Data updated to 2022 Q2

Fall 2021 release

- Upgrade full data release version 2021
- New web service : Multi isochrone to compute several isochrones and allow/avoid overlapping between geometries

Summer 2021 release

- New web services
  - Weather: forecasts and reports on current conditions / severe weather alerts
  - Autosuggest: automatic completion service which return places (points of interest, restaurants, monuments, etc.) in addition to addresses
- Features added on the following services:
  - Route calculation service (v7) including more attributs on route sheet and fuel consumption or kilowatt hours for electric vehicles
  - Geocoding (v4) including an improved mode and the capability to use district/neighbourhood

Fall 2020 release

- Upgrade full data release version 2020
- One new map layer: Charging stations for electric vehicles (France)
- Add exclusions for *Hazardous material* to all routing services (Route calculation, Route calculation (batch), Waypoint sequence, Isochrones, Matrix calculation, Search around, Search along, Pickup and delivery)
- Features added on the followings services:

- Add *formatItems* and *fields* the Route calculation service (v6)
- Improved formated results for Waypoint sequence (v4) and Compact matrix calculation (v4)

Spring 2020 release

- Add Estimated Time of Arrival (ETA) service to provide time to arrive to destination accounting real-time traffic
- Add batch Route calculation service to calculates a set of itineraries in one request
- Add Geoconcept Territory Manager API, this API allow to segment a territory or optimize existing ones
- Features added on others services:
  - Addresses coordinates for reverse geocoding (v3)
  - Add 2 news formats *completegeometry* and *compresscompletegeometry* to provide legs geometries on the Route calculation service

Winter 2020 release

- The autocompletion offers global coverage and additional settings to allow geographic filtering
- Add the capability to compute toll cost on route calculation varying according to the time of passage and the type of vehicle
- Add sections on the Geoconcept JavaScript API:
  - grey scale and night mode,
  - heat map layer,
  - search around,
  - tracking animation.

Summer 2019 release

- The access url has changed to: https://api.geoconcept.com/. The previous one is kept for compatibility.
- Add news parameters on the optimization service.
- Add sections on the Geoconcept JavaScript API:
  - customized icons for points,
  - external Popover,
  - using HERE maps layer,
  - add a GeoJSON,
  - locate yourself.

Spring 2019 release

- Complete update of the Geoconcept JavaScript API, based over the OpenLayers JavaScript library release 4.6.4.
- Add the pickup and delivery service can be used to list the best solutions for pickup/collection and drop-off/delivery, whether for the transport of people or of goods.

Fall 2018 release

Of the most significant new features, we will cite the following:

- data has been fully updated, and provides even more exhaustive coverage
- there is now an option for advanced specification of route calculations and route matrices:
  - refined speeds can now take traffic conditions into account
  - traffic restrictions for heavy goods vehicles (Truck Attributes)
  - new vehicle profiles (bicycle, bus, emergency truck, emergency vehicle, taxi, …)

Geoptimization JavaScript API

The Geoconcept JavaScript API lets you embed maps in your own web page. Designed especially for web sites with a heavy traffic, this DHTML/JavaScript client can be used to develop very reactive user interfaces and is compatible with all browsers (no ActiveX, Java…).

The Geoconcept JavaScript API is based over the OpenLayers JavaScript library (Release 4.6.4, see: openlayers.org). This API uses a subset of OpenLayers classes and adds a set of Geoconcept classes.

Load the JavaScript API

In your HTML document, include the CSS and JavaScript API files.

HTML

```
<script src="https://api.geoconcept.com/EU/GCW/geoconcept-web/combo?babel-external-helpers-min.js&yui-
min.js&gcui-loader-min.js&webjars-requires.js&loader-init-min.js"></script>
<script src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/build/ol.js"></script>
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<script src="https://api.geoconcept.com/EU/GCW/geoconcept-web/combo?gcui-htc6/htc.js"></script>
```

Specify your credentials

To access to Geoconcept web services, you need to specify your application key and token to use the API.

To get yours credentials simply fill the form [http://en.geoconcept.com/test-geoptimization-api] and you will receive an email with login identifiers valid for 90 days.

Before creating your map, specify your credentials using this JavaScript code :

JavaScript

```
GCUI.Settings = {appkey:"REPLACE_WITH_YOUR_APP_KEY", apptoken:"REPLACE_WITH_YOUR_APP_TOKEN"};
```

Map

Display a map

HTML

```
<div id="mapDiv" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};

var map = new GCUI.Map('mapDiv', options);
```

Map options

Option	Type	Description
server	String	Url of the LBS Platform tile server
layer	String	Name of the layer
x	Float	X-coordinate of the map center
y	Float	Y-coordinate of the map center
scale	Integer	Logical scale of the map
showSlider	Boolean	Display or not the zoom slider (default is true)

The following code example creates a map centered on Paris (using x, y coordinates and scale options) :

HTML

```
<div id="mapDiv1" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 1
};

var map = new GCUI.Map('mapDiv1', options);
```

Get the map

After creating a map, you can get the map object from its id :

JavaScript

```
function init() {
    var map = new GCUI.Map('map', options);
}
function doSomethingWithMap() {
    var map = GCUI.getMap('map');
}
```

Load the map

The GCUI.Map constructor with **server** and **layer** options arguments gets asynchronously map informations (extent, resolutions, …) from the Geoptimization server. The map events system trigger a *load* event to notify that the map is loaded :

HTML

```
<div id="map1" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map1', options);
map.onEvent("load", onMapLoaded);

function onMapLoaded() {
        var map = GCUI.getMap('map1');
        alert("Map is loaded. Map extent is : " + map.getExtent());
}
```

The following code registers a callback function that will be called when map is ready :

```
map.onEvent("load",onMapLoaded);
```

In this example, the callback function display the map extent :

```
function onMapLoaded() {
    var map = GCUI.getMap('map1');
    alert("Map is loaded. Map extent is : "+ map.getExtent());
}
```

Set the map zoom

To get the current zoom level of the map, use :

```
var zoomLevel = map.getZoom();
```

Note : The OpenLayers zoom level 0 corresponds to a map fully zoomed out and the zoom level 21 (depending on the map the scale levels may vary) corresponds to a map fully zoomed in. It's the reverse of the Geoconcept logical scales (in Geoconcept, logical scale 22 corresponds to a map fully zoomed out and logical scale 0 corresponds to a map fully zoomed in).

To get the current Geoconcept logical scale of the map, use :

```
var logicalScale = map.getLogicalScale();
```

So, to convert Geoconcept logical scale to OpenLayers zoom level :

```
var zoomLevel = map.getNumZoomLevels() - logicalScale;
```

You can specify the minimum (default value is 0) and maximum (default value is 22) logical scales of the map :

```
map.minLogicalScale = 3;
map.maxLogicalScale = 10;
```

Several API methods (from OpenLayers API) allow to modify the zoom level of the map :

- `zoomTo` : Zoom to a specific zoom level

- `zoomIn` : Zoom in
- `zoomOut` : Zoom out
- `zoomToExtent` : Zoom to a specific bounding-box
- `zoomToMaxExtent` : Zoom to the map extent

HTML

```
<div style="position: absolute; z-index: 5000; left: 65px;">
        <button type="button" onclick="GCUI.examples.zoomIn()">zoom in</button>
        <button type="button" onclick="GCUI.examples.zoomOut()">zoom out</button>
        <button type="button" onclick="GCUI.examples.zoomTo(0)">zoom to 0</button>
        <button type="button" onclick="GCUI.examples.zoomTo(22)">zoom to 22</button>
        <button type="button" onclick="GCUI.examples.zoomExtent()">zoom to bounds</button>
        <button type="button" onclick="GCUI.examples.zoomMapExtent()">map extent</button>
</div>
<div id="map2" style="height: 300px;"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 8
};
var map = new GCUI.Map('map2', options);

GCUI.examples = {
        zoomIn : function() {
                map.getView().setZoom(map.getView().getZoom() + 1);
        },
        zoomOut : function() {
                map.getView().setZoom(map.getView().getZoom() - 1);
        },
        zoomTo : function(z) {
                 map.getView().setZoom(z);
        },
        zoomExtent : function() {
                map.zoomToExtent([251363, 6247962, 270242, 6253695]);
        },
        zoomMapExtent : function() {
                map.getView().setZoom(map.getExtent());
        }
};
```

Set the map center

To get the current center of the map, use :

```
var center = map.getCenter();
```

To change the center of the map, use the `setCenter` method :

```
map.getView().setCenter([260803,6250829]);
```

Set the map size

To get the current size (in pixels) of the map div, use :

```
var size = map.getSize();
```

To modify the map size, use the `setSize` method :

```
map.setSize('80%','80%');
```

or

```
map.setSize('400px','400px');
```

HTML

```
<div id="map3" style="height: 300px;"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 8
};
var map = new GCUI.Map('map3', options);
map.setSize('400px', '250px');
```

Map Control

Add a control to the map

To add a control to the map, use the `addControl` method :

```
map.addControl(mycontrol);
```

Graphical scale

Create a **`ol.control.ScaleLine`** control and add it to the map to display a graphical scale.

HTML

```
<div id="map1" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 263860,
    y : 6242153,
```

```
    scale : 8
};

var map = new GCUI.Map('map1', options);
map.addControl(new ol.control.ScaleLine());
```

Zoom slider

Create a `ol.control.Zoom` control and add it to the map to display a scale slider.

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<style>
  .ol-rotate {
        top: 170px;
        right: 0;
      }
</style>
<div id="map2" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 8
};

var map = new GCUI.Map('map2', options);
map.addControl(new ol.control.Zoom());
```

Overview map

Create a new instance of `ol.control.OverviewMap` class and add it to the existing map to display an Overview map.

HTML

```
 <style>
      .ol-custom-overviewmap,
      .ol-custom-overviewmap.ol-uncollapsible {
        bottom: auto;
        left: auto;
        right: 0;
        top: 0;
      }

      .ol-custom-overviewmap:not(.ol-collapsed)  {
        border: 1px solid black;
      }

      .ol-custom-overviewmap .ol-overviewmap-map {
        border: none;
        width: 300px;
      }
```

```
      .ol-custom-overviewmap .ol-overviewmap-box {
        border: 2px solid red;
      }

      .ol-custom-overviewmap:not(.ol-collapsed) button{
        bottom: auto;
        left: auto;
        right: 1px;
        top: 1px;
      }

      .ol-rotate {
        top: 170px;
        right: 0;
      }
    </style>
<div id="map3" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 8
};

var map = new GCUI.Map('map3', options);
var overviewmap = new ol.control.OverviewMap({
                                 className: 'ol-overviewmap ol-custom-overviewmap'
                                  });
map.addControl(overviewmap);
```

Map Overlay

Basic Popup

To get a basic popup to be overlayed on the clicked coordinate, follow the below steps:

- Add a map click handler to initiate an event to render a popup

```
map.on('singleclick', function(evt) {
// Create a block to handle the map click event to render a pop up
};
```

- Create a new instance of the **`GCUI.Control.Popup`** class and pass the map, clicked coordinate and the information to be displayed on the popup.

```
var popup = new GCUI.Control.Popup(map, coordinate, {content : text});
```

- Then add the control to the existing map using **`map.addOverlay()`** function and pass the popup.

```
map.addOverlay(popup);
```

- To initialize the popup on the clicked coordinate use **`popup.initialize()`** function and pass the coordinate.

```
popup.initialize(coordinate);
```

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<style>
.anchorright#ol-popup {
    right: -30px;
}

.anchorright#ol-popup:after {
    right: 20px;
}

.anchorright#ol-popup:before {
    right: 20px;
}

.anchorbottom#ol-popup {
    bottom: 11px;
}

.anchorbottom#ol-popup:after, .anchorbottom#ol-popup:before {
    top: 100%;
}

#ol-popup-content.content {
    max-height: 25em;
    overflow-y: auto;
}

.#ol-popup{
    padding-top: 2em;
    padding-right: 0.35em;
}

.anchorbottom#ol-popup:after {
    border-top-color: white;
}

.anchorbottom#ol-popup:before {
    border-top-color: #cccccc;
}

.anchorleft#ol-popup {
    left: -25px;
}

.anchorleft#ol-popup:after {
    left: 25px;
}

.anchorleft#ol-popup:before {
    left: 25px;
}

.anchortop#ol-popup {
    top: 11px;
}

.anchortop#ol-popup:after, .anchortop#ol-popup:before {
    bottom: 100%;
```

```
}

.anchortop#ol-popup:after {
    border-bottom-color: white;
}

.anchortop#ol-popup:before {
    border-bottom-color: #cccccc;
}

#ol-popup {
    position: absolute;
    background-color: white;
    -webkit-filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    padding: 15px;
    border-radius: 5px;
    border: 1px solid #cccccc;
    min-width: 280px;
}

#ol-popup:after, #ol-popup:before {
    border: solid transparent;
    content: " ";
    height: 0;
    width: 0;
    position: absolute;
    pointer-events: none;
}

#ol-popup:after {
    border-width: 10px;
    margin-left: -10px;
}

#ol-popup:before {
    border-width: 11px;
    margin-left: -11px;
}

#ol-popup-closer {
    text-decoration: none;
    position: absolute;
    top: 2px;
    right: 8px;
    font-family: Icons;
}

#ol-popup-closer:hover {
    cursor: pointer;
    color: #004687;
}

#ol-popup-closer:after {
  content: "#";
}

#ol-popup .pages {
        text-align: right;
    position: relative;
    right: -1em;
    bottom: -0.8em;
}
```

```
#ol-popup .pages i {
    cursor: pointer;
}

#ol-popup .pages > * {
    display: inline-block;
}

#ol-popup .pages i.disabled {
    cursor: default;
}

.gcuiAnchoredContentData {
    height: 100%;
    width: 100%;
    overflow-y: auto;
    overflow-x: hidden;
}

.gcuiAnchoredContentData .table-width {
    width: 50%;
}

.gcuiAnchoredContentData .ui.table  .right.aligned .content {
    text-align: right;
    margin-right: .2em;
    color: #808080;
}

.gcuiAnchoredContentData .ui.table .content {
    color: #000000;
    border: none !important;
}

.gcuiAnchoredContentData .ui.table td {
    padding: .25em;
    border: none !important;
}
.gcuiAnchoredContentData .ui.table, th, td {
         border: none !important;
}
.ol-custom-overviewmap,
.ol-custom-overviewmap.ol-uncollapsible {
        bottom: auto;
        left: auto;
        right: 0;
        top: 0;
}
.ol-custom-overviewmap:not(.ol-collapsed)  {
        border: 1px solid black;
}
.ol-custom-overviewmap .ol-overviewmap-map {
        border: none;
        width: 300px;
}
.ol-custom-overviewmap .ol-overviewmap-box {
        border: 2px solid red;
}
.ol-custom-overviewmap:not(.ol-collapsed) button{
        bottom: auto;
        left: auto;
        right: 1px;
```

```
        top: 1px;
}
.ol-rotate {
        top: 170px;
        right: 0;
}
</style>
<div id="mapDiv" style="height:300px"></div>
```

JavaScript

```
var options = {
           server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
           layer : 'STANDARD',
           x : 260803,
           y : 6250829,
           scale : 8
                };
var map = new GCUI.Map('mapDiv', options);
// Creating a new instance of the zoomslider
var zoomslider = new ol.control.Zoom();
// Adding the zoom control to the existing map
map.addControl(zoomslider);
// Creating a new instance of the overview map
var overviewmap = new ol.control.OverviewMap({
className: 'ol-overviewmap ol-custom-overviewmap'
                                 });
// Adding the overviewmap control to the existing map
map.addControl(overviewmap);
// Creating a new instance of the scale
var scaleline = new ol.control.ScaleLine();
// Adding the scale control to the existing map
map.addControl(scaleline);
/**
* Add a click handler to the map to render the popup.
*/
map.on('singleclick', function(evt) {
        map.getOverlays().clear();
        var coordinate = evt.coordinate;
        var map_precision = map.getLayers().getArray();
        var x = coordinate[0]/map_precision[0].precision;
        var y = coordinate[1]/map_precision[0].precision;
        var epsg = this.getProjection();
        var opt_options = "<div class='gcuiAnchoredContentData'><table class='ui very basic table
 unstackable' cellspacing='0' cellpadding='0'><tbody><tr><td class='right aligned'><div class='content'>X</
div></td><td><div class='content'>" + x + "</div></td></tr><tr><td class='right aligned'><div
 class='content'>Y</div></td><td><div class='content'>" + y + "</div></td></tr><tr><td class='right
 aligned'><div class='content'>EPSG</div></td><td><div class='content'>" + epsg + "</div></td></tr></
tbody></table></div>";
        var popup = new GCUI.Control.Popup(map, coordinate, {content : opt_options});
        map.addOverlay(popup);
        popup.initialize(coordinate);
});
```

Geoconcept Infobox

To get a Geoconcept Infobox to be overlayed on the clicked coordinate, follow the below steps:

- Create an infobox webservice request and pass the clicked coordinate to get back a JSON file

```
const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/gcis/json/object/info?mapName=HE-
ENT-M21-EURO_DOM-map-[20211123-150500]-v[6]/HE-ENT-M21-EURO_DOM_cua&x=' + x + '&y=' + y +
 '&infoMode=infoboxfields&tabName=STANDARD&appkey=' + apikey + '&apptoken=' + apptoken;
```

```
Note: The mapName and the tabName would change based on the continent of interest. To get it    please
 contact Geoconcept sales support team.
```

- Add a map click handler to initiate an event to render a popup

```
map.on('singleclick', function(evt) {
// Create a block to handle the map click event to render a pop up
};
```

- Create a new instance of the **`GCUI.Control.Popup`** class and pass the map, clicked coordinate and the information to be displayed on the infobox.

```
var popup = new GCUI.Control.Popup(map, coordinate, {content : text});
```

- Then add the control to the existing map using **`map.addOverlay()`** function and pass the popup.

```
map.addOverlay(popup);
```

- To initialize the popup on the clicked coordinate use **`popup.initialize()`** function and pass the coordinate.

```
popup.initialize(coordinate);
```

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<style>
.anchorright#ol-popup {
    right: -30px;
}

.anchorright#ol-popup:after {
    right: 20px;
}

.anchorright#ol-popup:before {
    right: 20px;
}

.anchorbottom#ol-popup {
    bottom: 11px;
}

.anchorbottom#ol-popup:after, .anchorbottom#ol-popup:before {
    top: 100%;
}

#ol-popup-content.content {
    max-height: 25em;
    overflow-y: auto;
}

.#ol-popup{
    padding-top: 2em;
```

```
    padding-right: 0.35em;
}

.anchorbottom#ol-popup:after {
    border-top-color: white;
}

.anchorbottom#ol-popup:before {
    border-top-color: #cccccc;
}

.anchorleft#ol-popup {
    left: -25px;
}

.anchorleft#ol-popup:after {
    left: 25px;
}

.anchorleft#ol-popup:before {
    left: 25px;
}

.anchortop#ol-popup {
    top: 11px;
}

.anchortop#ol-popup:after, .anchortop#ol-popup:before {
    bottom: 100%;
}

.anchortop#ol-popup:after {
    border-bottom-color: white;
}

.anchortop#ol-popup:before {
    border-bottom-color: #cccccc;
}

#ol-popup {
    position: absolute;
    background-color: white;
    -webkit-filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    padding: 15px;
    border-radius: 5px;
    border: 1px solid #cccccc;
    min-width: 280px;
}

#ol-popup:after, #ol-popup:before {
    border: solid transparent;
    content: " ";
    height: 0;
    width: 0;
    position: absolute;
    pointer-events: none;
}

#ol-popup:after {
    border-width: 10px;
    margin-left: -10px;
}
```

```
#ol-popup:before {
    border-width: 11px;
    margin-left: -11px;
}

#ol-popup-closer {
    text-decoration: none;
    position: absolute;
    top: 2px;
    right: 8px;
    font-family: Icons;
}

#ol-popup-closer:hover {
    cursor: pointer;
    color: #004687;
}

#ol-popup-closer:after {
  content: "#";
}

#ol-popup .pages {
        text-align: right;
    position: relative;
    right: -1em;
    bottom: -0.8em;
}

#ol-popup .pages i {
    cursor: pointer;
}

#ol-popup .pages > * {
    display: inline-block;
}

#ol-popup .pages i.disabled {
    cursor: default;
}

.gcuiAnchoredContentData {
    height: 100%;
    width: 100%;
    overflow-y: auto;
    overflow-x: hidden;
}

.gcuiAnchoredContentData .table-width {
    width: 50%;
}

.gcuiAnchoredContentData .ui.table  .right.aligned .content {
    text-align: right;
    margin-right: .2em;
    color: #808080;
}

.gcuiAnchoredContentData .ui.table .content {
    color: #000000;
    border: none !important;
}
```

```
.gcuiAnchoredContentData .ui.table td {
    padding: .25em;
    border: none !important;
}
.gcuiAnchoredContentData .ui.table, th, td {
         border: none !important;
}
.ol-custom-overviewmap,
.ol-custom-overviewmap.ol-uncollapsible {
        bottom: auto;
        left: auto;
        right: 0;
        top: 0;
}
.ol-custom-overviewmap:not(.ol-collapsed)  {
        border: 1px solid black;
}
.ol-custom-overviewmap .ol-overviewmap-map {
        border: none;
        width: 300px;
}
.ol-custom-overviewmap .ol-overviewmap-box {
        border: 2px solid red;
}
.ol-custom-overviewmap:not(.ol-collapsed) button{
        bottom: auto;
        left: auto;
        right: 1px;
        top: 1px;
}
.ol-rotate {
        top: 170px;
        right: 0;
}
</style>
<div id="mapDiv1" style="height:300px"></div>
```

JavaScript

```
var options = {
           server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
           layer : 'STANDARD',
           x : 260803,
           y : 6250829,
           scale : 8
                };
var map = new GCUI.Map('mapDiv1', options);
// Creating a new instance of the zoomslider
var zoomslider = new ol.control.Zoom();
// Adding the zoom control to the existing map
map.addControl(zoomslider);
// Creating a new instance of the overview map
var overviewmap = new ol.control.OverviewMap({
className: 'ol-overviewmap ol-custom-overviewmap'
                                });
// Adding the overviewmap control to the existing map
map.addControl(overviewmap);
// Creating a new instance of the scale
var scaleline = new ol.control.ScaleLine();
// Adding the scale control to the existing map
map.addControl(scaleline);
/**
```

```
* Add a click handler to the map to render the popup.
*/
map.on('singleclick', function(evt)
        {
                var coordinate = evt.coordinate;
                // Getting the precision of the map
                var map_precision = map.getLayers().getArray();
                var x = coordinate[0]/map_precision[0].precision;
                var y = coordinate[1]/map_precision[0].precision;
                // Creating a HTTP request for getting the underlying information
                const request = new XMLHttpRequest();
                const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/gcis/json/object/info?
mapName=HE-ENT-M21-EURO_DOM-map-[20211123-150500]-v[6]/HE-ENT-M21-EURO_DOM_cua&x=' + x + '&y=' + y +
 '&infoMode=infoboxfields&tabName=STANDARD&appkey=' + $("input[name=apikey]").val() + '&apptoken=' +
 $("input[name=apitoken]").val();
                request.open("GET", url);
                request.send();
// Treatment of the output JSON to display the information on the popup
request.onreadystatechange=(e)=>{
           if(request.readyState == 4 && request.status == 200)
                {
                        var responseJson = request.responseText;
                        var infobox = JSON.parse(responseJson);
                        var opt_options = "<div class='gcuiAnchoredContentData'><table class='ui very
 basic table unstackable' cellspacing='0' cellpadding='0'><tbody><tr><td class='right aligned'><div
 class='content'>Class</div></td><td><div class='content'>" + infobox.result.fields.Class + "</div></td></
tr><tr><td class='right aligned'><div class='content'>SubClass</div></td><td><div class='content'>" +
 infobox.result.fields.Subclass + "</div></td></tr><tr><td class='right aligned'><div class='content'>Name</
div></td><td><div class='content'>" + infobox.result.fields.Name + "</div></td></tr></tbody></table></div>";
                        var infobox = new GCUI.Control.Popup(map, coordinate, {content : opt_options});
                        map.addOverlay(infobox);
                        infobox.initialize(coordinate);
                }
        }
});
```

Map Mode

Grey Scale

To get a grey scale map , follow the below steps:

- Create a new instance of the class **`GCUI.Map`** and push a new instance of the class **`ol.View`** in it.

```
var map = new GCUI.Map('mapDiv', {
        layers: [],
        view: new ol.View({
            center: center,
            zoom: zoom,
            projection: projection
        }),
        zoom: true
    });
```

- Create a new instance of the class **`ol.source.Raster`** and pass the map source and play around the pixel operation to convert it into grey scale.

```
var rasterSource = new ol.source.Raster({
```

```
sources: [
                  // original source here, e.g. ol.source.WMTS
                  gcLayer.getSource()
],
operation: (pixels, data) => {
        var pixel = pixels[0];
        var lightness = (pixel[0] * 0.3 + pixel[1] * 0.59 + pixel[2] * 0.11);
        return [lightness, lightness, lightness, pixel[3]];
}
});
```

HTML

```
<div id="mapDiv1" style="height:300px"></div>
```

JavaScript

```
function createMap(center, zoom, projection) {
         return new GCUI.Map('mapDiv1', {
        layers: [],
        view: new ol.View({
            center: center,
            zoom: zoom,
            projection: projection
        }),
        zoom: true
    });
}
var map = createMap([261108, 6251020], 6);

if (map) {
var gcLayer = new GCUI.Layer.GeoConcept('GC layer',
        'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
        {
                layer:'STANDARD',
                crossDomain: "Anonymous"
        });
gcLayer.on('change:initialized', function() {
        var rasterSource = new ol.source.Raster({
                sources: [
                                  // original source here, e.g. ol.source.WMTS
                                  gcLayer.getSource()
                ],
                operation: (pixels, data) => {
                        var pixel = pixels[0];
                        var lightness = (pixel[0] * 0.3 + pixel[1] * 0.59 + pixel[2] * 0.11);
                        return [lightness, lightness, lightness, pixel[3]];
                }
        });
        var raster = new ol.layer.Image({
                source: rasterSource,
                name: "Source Raster (Greyscale)"
        });
        raster.setMap(map);
});
        }
```

Night mode

To get a night mode map , follow the below steps:

- Create a new instance of the class `GCUI.Map` and push a new instance of the class `ol.View` in it.

```
var map = new GCUI.Map('mapDiv', {
        layers: [],
        view: new ol.View({
            center: center,
            zoom: zoom,
            projection: projection
        }),
        zoom: true
    });
```

- Create a new instance of the class `ol.source.Raster` and pass the map source and play around the pixel operation to convert it into night mode.

```
var rasterSource = new ol.source.Raster({
sources: [
                  // original source here, e.g. ol.source.WMTS
                  gcLayer.getSource()
],
operation: (pixels, data) => {
        var pixel = pixels[0];
        var lightness = (255 - pixel[0], 255 - pixel[1], 255 - pixel[2]);
        return [lightness, lightness, lightness, pixel[3]];
}
});
```

HTML

```
<div id="mapDiv2" style="height:300px"></div>
```

JavaScript

```
function createMap(center, zoom, projection) {
         return new GCUI.Map('mapDiv2', {
        layers: [],
        view: new ol.View({
            center: center,
            zoom: zoom,
            projection: projection
        }),
        zoom: true
    });
}
var map = createMap([261108, 6251020], 6);

if (map) {
var gcLayer = new GCUI.Layer.GeoConcept('GC layer',
        'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
        {
                layer:'STANDARD',
                crossDomain: "Anonymous"
        });
gcLayer.on('change:initialized', function() {
        var rasterSource = new ol.source.Raster({
                sources: [
                                  // original source here, e.g. ol.source.WMTS
                                  gcLayer.getSource()
                ],
```

```
                operation: (pixels, data) => {
                        var pixel = pixels[0];
                        var lightness = (255 - pixel[0], 255 - pixel[1], 255 - pixel[2]);
                        return [lightness, lightness, lightness, pixel[3]];
                }
        });
        var raster = new ol.layer.Image({
                source: rasterSource,
                name: "Source Raster (Night mode)"
        });
        raster.setMap(map);
});
        }
```

Layer

Add a vector layer

To add a vector layer and to display it on the map, follow the below steps:

- Create a new instance of **`ol.Feature`** class

```
var point_feature = new ol.Feature({});
```

- Create a new instance of **`ol.geom.Point`** class and push the X and Y

```
var point_geom = new ol.geom.Point([260803,6250829]);
```

- Set the geometry of the ol feature as the **`ol.geom.Point`** class

```
point_feature.setGeometry(point_geom);
```

- Create a new instance of **`GCUI.Layer.StandardVector`** class and map the source of the vector to the ol feature

```
var vector_layer = new GCUI.Layer.StandardVector({
                    source: new ol.source.Vector({
                    features: [point_feature]
             })
        });
```

- Add the layer to the existing map using `map.addLayer` method and push the vector layer created

```
map.addLayer(vector_layer);
```

- Define the style of the point using **`ol.style.Fill`** and **`ol.style.Stroke`** classes

```
var fill = new ol.style.Fill({
              color: [0, 70, 135, 0.9]
        });
var stroke = new ol.style.Stroke({
              color: [255, 255, 255, 1],
              width: 2
       });
```

- Create a new instance of the **`ol.style.Style`** class and set the style of the layer using `vector_layer.setStyle()`

```
var style = new ol.style.Style({
               image: new ol.style.Circle({
               fill: fill,
               stroke: stroke,
               radius: 5
         }),
               fill: fill,
               stroke: stroke
         });
vector_layer.setStyle(style);
```

- Zoom to the added vector layer using `map.zoomToExtent()` method

```
map.zoomToExtent(point_feature.getGeometry().getExtent());
```

HTML

```
<div id="map" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
                        var point_geom = new ol.geom.Point([260803,6250829]);
                                point_feature = new ol.Feature({});
                                point_feature.setGeometry(point_geom);
                                vector_layer = new GCUI.Layer.StandardVector({
                                source: new ol.source.Vector({
                                        features: [point_feature]
                                                })
                                        });
                                map.addLayer(vector_layer);
//Adding a style to the vector layer
                                var fill = new ol.style.Fill({
                                        color: [0, 70, 135, 0.9]
                                });
                                var stroke = new ol.style.Stroke({
                                color: [255, 255, 255, 1],
                                width: 2
                                });
                                var style = new ol.style.Style({
                                image: new ol.style.Circle({
                                fill: fill,
                                stroke: stroke,
                                radius: 5
                                }),
                                fill: fill,
                                stroke: stroke
                                });
                                vector_layer.setStyle(style);
                                map.zoomToExtent(point_feature.getGeometry().getExtent());
```

```
                                map.getView().setZoom(8);
}
```

Add a vector layer with custom icon

To add a vector layer and to display it on the map, follow the below steps:

- Create a new instance of **`ol.Feature`** class

```
var point_feature = new ol.Feature({});
```

- Create a new instance of **`ol.geom.Point`** class and push the X and Y

```
var point_geom = new ol.geom.Point([258353.06,6240408.91]);
```

- Set the geometry of the ol feature as the **`ol.geom.Point`**

```
point_feature.setGeometry(point_geom);
```

- Create a new instance of **`GCUI.Layer.StandardVector`** class and map the source of the vector to the ol feature

```
var vector_layer = new GCUI.Layer.StandardVector({
                   source: new ol.source.Vector({
                   features: [point_feature]
             })
        });
```

- Add the layer to the existing map using `map.addLayer` method and push the vector layer created

```
map.addLayer(vector_layer);
```

- Define the style of the point using **`ol.style.Fill`** and **`ol.style.Stroke`** classes

```
var fill = new ol.style.Fill({
              color: [0, 70, 135, 0.9]
        });
var stroke = new ol.style.Stroke({
              color: [255, 255, 255, 1],
              width: 2
       });
```

- Create a new instance of the **`ol.style.Style`** class and set the source of the **`ol.style.Icon`** an image URL and set the style using `vector_layer.setStyle()`

```
        var point_style = new ol.style.Style({
                                    image: new ol.style.Icon({
                                    crossOrigin: 'anonymous',
                                    src: 'https://i.ibb.co/SBRmrF5/show-Image.png'
                                    })
                            });
                            vector_layer.setStyle(point_style);
```

- Zoom to the added vector layer using `map.zoomToExtent()` method

```
map.zoomToExtent(point_feature.getGeometry().getExtent());
```

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<style>
.anchorright#ol-popup {
    right: -30px;
}

.anchorright#ol-popup:after {
    right: 20px;
}

.anchorright#ol-popup:before {
    right: 20px;
}

.anchorbottom#ol-popup {
    bottom: 11px;
}

.anchorbottom#ol-popup:after, .anchorbottom#ol-popup:before {
    top: 100%;
}

#ol-popup-content.content {
    max-height: 25em;
    overflow-y: auto;
}

.#ol-popup{
    padding-top: 2em;
    padding-right: 0.35em;
}

.anchorbottom#ol-popup:after {
    border-top-color: white;
}

.anchorbottom#ol-popup:before {
    border-top-color: #cccccc;
}

.anchorleft#ol-popup {
    left: -25px;
}

.anchorleft#ol-popup:after {
    left: 25px;
}

.anchorleft#ol-popup:before {
    left: 25px;
}

.anchortop#ol-popup {
    top: 11px;
}

.anchortop#ol-popup:after, .anchortop#ol-popup:before {
    bottom: 100%;
```

```
}

.anchortop#ol-popup:after {
    border-bottom-color: white;
}

.anchortop#ol-popup:before {
    border-bottom-color: #cccccc;
}

#ol-popup {
    position: absolute;
    background-color: white;
    -webkit-filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    padding: 15px;
    border-radius: 5px;
    border: 1px solid #cccccc;
    min-width: 280px;
}

#ol-popup:after, #ol-popup:before {
    border: solid transparent;
    content: " ";
    height: 0;
    width: 0;
    position: absolute;
    pointer-events: none;
}

#ol-popup:after {
    border-width: 10px;
    margin-left: -10px;
}

#ol-popup:before {
    border-width: 11px;
    margin-left: -11px;
}

#ol-popup-closer {
    text-decoration: none;
    position: absolute;
    top: 2px;
    right: 8px;
    font-family: Icons;
}

#ol-popup-closer:hover {
    cursor: pointer;
    color: #004687;
}

#ol-popup-closer:after {
  content: "#";
}

#ol-popup .pages {
        text-align: right;
    position: relative;
    right: -1em;
    bottom: -0.8em;
}
```

```
#ol-popup .pages i {
    cursor: pointer;
}

#ol-popup .pages > * {
    display: inline-block;
}

#ol-popup .pages i.disabled {
    cursor: default;
}

.gcuiAnchoredContentData {
    height: 100%;
    width: 100%;
    overflow-y: auto;
    overflow-x: hidden;
}

.gcuiAnchoredContentData .table-width {
    width: 50%;
}

.gcuiAnchoredContentData .ui.table  .right.aligned .content {
    text-align: right;
    margin-right: .2em;
    color: #808080;
}

.gcuiAnchoredContentData .ui.table .content {
    color: #000000;
    border: none !important;
}

.gcuiAnchoredContentData .ui.table td {
    padding: .25em;
    border: none !important;
}
.gcuiAnchoredContentData .ui.table, th, td {
         border: none !important;
}
.ol-custom-overviewmap,
.ol-custom-overviewmap.ol-uncollapsible {
        bottom: auto;
        left: auto;
        right: 0;
        top: 0;
}
.ol-custom-overviewmap:not(.ol-collapsed)  {
        border: 1px solid black;
}
.ol-custom-overviewmap .ol-overviewmap-map {
        border: none;
        width: 300px;
}
.ol-custom-overviewmap .ol-overviewmap-box {
        border: 2px solid red;
}
.ol-custom-overviewmap:not(.ol-collapsed) button{
        bottom: auto;
        left: auto;
        right: 1px;
```

```
        top: 1px;
}
.ol-rotate {
        top: 170px;
        right: 0;
}
</style>
<div id="map1" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map1', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
                                var vectorSource = new ol.source.Vector({});
                                var point_feature_1 = new ol.Feature({
                                geometry: new ol.geom.Point([258353.06,6240408.91]),
                                Name: "Groupe Geoconcept",
                                Address: "152 Avenue Aristide Briand, 92220 Bagneux, France"
                                });
                        vectorSource.addFeature(point_feature_1);
                        vector_layer = GCUI.Layer.StandardVector({
                                source: vectorSource
                                        });
                                map.addLayer(vector_layer);
//Set the source of the point style to a custom image
                                var point_style = new ol.style.Style({
                                        image: new ol.style.Icon({
                                        crossOrigin: 'anonymous',
                                        src: 'https://i.ibb.co/SBRmrF5/show-Image.png'
                                        })
                                });
                                vector_layer.setStyle(point_style);
                                extent = vectorSource.getExtent();
                                map.getView().fit(extent);
                                map.getView().setZoom(12);
                                //select interaction working on "click"
                                var selectClick = new ol.interaction.Select({
                                  condition: ol.events.condition.click
                                });
                                 if (selectClick !== null) {
                                        map.addInteraction(selectClick);
                                        selectClick.on('select', function(e) {
                                          var feature = e.target.getFeatures().getArray();
                                          if (feature.length >= 1) {
                                          var opt_options = "<div class='gcuiAnchoredContentData'><table
 class='ui very basic table unstackable' cellspacing='0' cellpadding='0'><tbody><tr><td class='right
 aligned'><div class='content'>Name</div></td><td><div class='content'>" + feature[0].values_.Name
 + "</div></td></tr><tr><td class='right aligned'><div class='content'>Address</div></td><td><div
 class='content'>" + feature[0].values_.Address + "</div></td></tr></tbody></table></div>";
                                          var popup = new GCUI.Control.Popup(map,
 feature[0].values_.geometry.flatCoordinates, {content : opt_options});
                                          map.addOverlay(popup);
                                          popup.initialize(feature[0].values_.geometry.flatCoordinates);
                                          }
                                        });
         }
}
```

Embed a video or an image

To embed a video or an image to the vector layer, follow the below steps:

- Follow all the steps mentioned in the above section (Add a vector layer with custom icon)
- Define the popup with video URL and image URL as below

```
var opt_options = "<div class='gcuiAnchoredContentData'><table class='ui very basic table unstackable'
 cellspacing='0' cellpadding='0'><tbody><tr><td class='right aligned'><div class='content'>Name</
div></td><td><div class='content'>" + feature[0].values_.Name + "</div></td></tr><tr><td class='right
 aligned'><div class='content'>Address</div></td><td><div class='content'>" + feature[0].values_.Address
 + "</div></td></tr><tr><td class='right aligned'><div class='content'>Company logo</div></td><td><div
 class='content'><img src='https://en.geoconcept.com/sites/all/themes/geoevo/assets/images/logo.png'
 alt='Italian Trulli'></div></td></tr><tr><td class='right aligned'><div class='content'>Corporate Video</
div></td><td><div class='content'><iframe width='300' height='200' src='https://www.youtube.com/embed/
zYZHuYipOjM'></iframe></div></td></tr></tbody></table></div>";
```

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<style>
.anchorright#ol-popup {
    right: -30px;
}

.anchorright#ol-popup:after {
    right: 20px;
}

.anchorright#ol-popup:before {
    right: 20px;
}

.anchorbottom#ol-popup {
    bottom: 11px;
}

.anchorbottom#ol-popup:after, .anchorbottom#ol-popup:before {
    top: 100%;
}

#ol-popup-content.content {
    max-height: 25em;
    overflow-y: auto;
}

.#ol-popup{
    padding-top: 2em;
    padding-right: 0.35em;
}

.anchorbottom#ol-popup:after {
    border-top-color: white;
}

.anchorbottom#ol-popup:before {
    border-top-color: #cccccc;
}

.anchorleft#ol-popup {
    left: -25px;
```

```
}

.anchorleft#ol-popup:after {
    left: 25px;
}

.anchorleft#ol-popup:before {
    left: 25px;
}

.anchortop#ol-popup {
    top: 11px;
}

.anchortop#ol-popup:after, .anchortop#ol-popup:before {
    bottom: 100%;
}

.anchortop#ol-popup:after {
    border-bottom-color: white;
}

.anchortop#ol-popup:before {
    border-bottom-color: #cccccc;
}

#ol-popup {
    position: absolute;
    background-color: white;
    -webkit-filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    padding: 15px;
    border-radius: 5px;
    border: 1px solid #cccccc;
    min-width: 280px;
}

#ol-popup:after, #ol-popup:before {
    border: solid transparent;
    content: " ";
    height: 0;
    width: 0;
    position: absolute;
    pointer-events: none;
}

#ol-popup:after {
    border-width: 10px;
    margin-left: -10px;
}

#ol-popup:before {
    border-width: 11px;
    margin-left: -11px;
}

#ol-popup-closer {
    text-decoration: none;
    position: absolute;
    top: 2px;
    right: 8px;
    font-family: Icons;
}
```

```
#ol-popup-closer:hover {
    cursor: pointer;
    color: #004687;
}

#ol-popup-closer:after {
  content: "#";
}

#ol-popup .pages {
        text-align: right;
    position: relative;
    right: -1em;
    bottom: -0.8em;
}

#ol-popup .pages i {
    cursor: pointer;
}

#ol-popup .pages > * {
    display: inline-block;
}

#ol-popup .pages i.disabled {
    cursor: default;
}

.gcuiAnchoredContentData {
    height: 100%;
    width: 100%;
    overflow-y: auto;
    overflow-x: hidden;
}

.gcuiAnchoredContentData .table-width {
    width: 50%;
}

.gcuiAnchoredContentData .ui.table  .right.aligned .content {
    text-align: right;
    margin-right: .2em;
    color: #808080;
}

.gcuiAnchoredContentData .ui.table .content {
    color: #000000;
    border: none !important;
}

.gcuiAnchoredContentData .ui.table td {
    padding: .25em;
    border: none !important;
}
.gcuiAnchoredContentData .ui.table, th, td {
         border: none !important;
}
.ol-custom-overviewmap,
.ol-custom-overviewmap.ol-uncollapsible {
        bottom: auto;
        left: auto;
        right: 0;
```

```
        top: 0;
}
.ol-custom-overviewmap:not(.ol-collapsed)  {
        border: 1px solid black;
}
.ol-custom-overviewmap .ol-overviewmap-map {
        border: none;
        width: 300px;
}
.ol-custom-overviewmap .ol-overviewmap-box {
        border: 2px solid red;
}
.ol-custom-overviewmap:not(.ol-collapsed) button{
        bottom: auto;
        left: auto;
        right: 1px;
        top: 1px;
}
.ol-rotate {
        top: 170px;
        right: 0;
}
</style>
<div id="map2" style="height:600px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map2', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
                                var vectorSource = new ol.source.Vector({});
                                var point_feature_1 = new ol.Feature({
                                geometry: new ol.geom.Point([258353.06,6240408.91]),
                                Name: "Groupe Geoconcept",
                                Address: "152 Avenue Aristide Briand, 92220 Bagneux, France"
                                });
                        vectorSource.addFeature(point_feature_1);
                        vector_layer = new GCUI.Layer.StandardVector({
                                source: vectorSource
                                        });
                                map.addLayer(vector_layer);
//Set the source of the point style to a custom image
                                var point_style = new ol.style.Style({
                                        image: new ol.style.Icon({
                                        crossOrigin: 'anonymous',
                                        src: 'https://i.ibb.co/SBRmrF5/show-Image.png'
                                        })
                                });
                                vector_layer.setStyle(point_style);
                                extent = vectorSource.getExtent();
                                map.getView().fit(extent);
                                map.getView().setZoom(12);
                                //select interaction working on "click"
                                var selectClick = new ol.interaction.Select({
                                  condition: ol.events.condition.click
                                });
                                 if (selectClick !== null) {
                                        map.addInteraction(selectClick);
                                        selectClick.on('select', function(e) {
```

```
                                        var feature = e.target.getFeatures().getArray();
                                        if (feature.length >= 1) {
                                        var opt_options = "<div class='gcuiAnchoredContentData'><table
 class='ui very basic table unstackable' cellspacing='0' cellpadding='0'><tbody><tr><td class='right
 aligned'><div class='content'>Name</div></td><td><div class='content'>" + feature[0].values_.Name
 + "</div></td></tr><tr><td class='right aligned'><div class='content'>Address</div></td><td><div
 class='content'>" + feature[0].values_.Address + "</div></td></tr><tr><td class='right aligned'><div
 class='content'>Company logo</div></td><td><div class='content'><img src='https://en.geoconcept.com/
sites/all/themes/geoevo/assets/images/logo.png' alt='Italian Trulli'></div></td></tr><tr><td class='right
 aligned'><div class='content'>Corporate Video</div></td><td><div class='content'><iframe width='300'
 height='200' src='https://www.youtube.com/embed/zYZHuYipOjM'></iframe></div></td></tr></tbody></table></
div>";
                                        var popup = new GCUI.Control.Popup(map,
 feature[0].values_.geometry.flatCoordinates, {content : opt_options});
                                        map.addOverlay(popup);
                                        popup.initialize(feature[0].values_.geometry.flatCoordinates);
                                        }
                                      });
         }
}
```

Add an external popover

To add an external popover to the map, follow the below steps:

- Create a new instance of **`ol.Overlay`** and push the position and HTML element to it and add the overlay object to the existing map.

```
 var pos = ol.proj.fromLonLat([2.320825, 48.7965412]);
                                                        var geoconcept = new ol.Overlay({
                                                        position: pos,
                                                        element: document.getElementById('geoconcept')
                                                        });
                                                        map.addOverlay(geoconcept);
```

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<script src="https://code.jquery.com/jquery-2.2.3.min.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
<style>
#geoconcept {
                font-family: "Open Sans";
        background-color:white;
        color: #104685;
        font-size: 12pt;
        font-weight: bold;
      }
.popover-content {
        min-width: 180px;
      }
</style>
<div id="map3" style="height:300px"></div>
<div style="display: none;" >
      <!-- Clickable label for Geoconcept -->
      <a class="overlay" id="geoconcept" target="_blank" href="https://en.geoconcept.com/">Groupe
 Geoconcept</a>
    </div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map3', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
                                var vectorSource = new ol.source.Vector({});
                                var point_feature_1 = new ol.Feature({
//Create a new point by pushing the X & Y
                                geometry: new ol.geom.Point([258353.06,6240408.91]),
                                Name: "Groupe Geoconcept",
                                Address: "152 Avenue Aristide Briand, 92220 Bagneux, France"
                                });
                                vectorSource.addFeature(point_feature_1);
                                vector_layer = new GCUI.Layer.StandardVector({
                                source: vectorSource
                                });
                                map.addLayer(vector_layer);
//Set the source of the point style to a custom image
                                var point_style = new ol.style.Style({
                                        image: new ol.style.Icon({
                                        crossOrigin: 'anonymous',
                                        src: 'https://i.ibb.co/SBRmrF5/show-Image.png'
                                        })
                                });
                                vector_layer.setStyle(point_style);
                                extent = vectorSource.getExtent();
                                map.getView().fit(extent);
                                map.getView().setZoom(12);
//Add a label overlay (with hyperlink) on the map
                                var pos = ol.proj.fromLonLat([2.320825, 48.7965412]);
                                var geoconcept = new ol.Overlay({
                                position: pos,
                                element: document.getElementById('geoconcept')
                                });
                                map.addOverlay(geoconcept);
}
```

HERE maps tiles API

To add a HERE maps tile to the existing Geoconcept map, follow the below steps:

- Generate the HERE maps tile URL and append it with HERE maps appId and appCode
- Create a new instance of `ol.layer.Tile` and push the HERE maps tile URL, required tile scheme (normal, transit, satellite, pedestrian, terrain or hybrid) and its attributions and add the layer to the existing map.

```
var urlTpl = 'https://{1-4}.{base}.maps.cit.api.here.com' +
        '/{type}/2.1/maptile/newest/{scheme}/{z}/{x}/{y}/256/png' +
        '?app_id={app_id}&app_code={app_code}';
var layers = [];
var layerDesc = hereLayers[0];
      layers.push(new ol.layer.Tile({
                  name: 'HERE-NORMAL',
          visible: false,
          preload: Infinity,
          source: new ol.source.XYZ({
```

```
            url: createUrl(urlTpl, layerDesc),
            attributions: 'Map Tiles &copy; ' + new Date().getFullYear() + ' ' +
              '<a href="http://developer.here.com">HERE</a>'
          })
        }));
map.addLayer(layers[0]);
```

HTML

```
 <link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css">
<style>
.checkbox {
margin-left: 25px;
margin-top: 0px;
}
.checkbox, .radio {
    position: relative;
    display: block;
    margin-top: 10px;
    margin-bottom: -20px;
}
.layerSwitcher .item.layer:hover {
background-color: #000000;
}
.layerSwitcher .layer > .label,
.layerSwitcher .group > .label {
    display: inline-block;
    margin-right: 0em;
}
.layman {
  position: absolute;
  width: 15%;
  background:rgba(0,0,0,0.5);
  z-index: 50;
}
</style>
<div>
<div id="layers" class="layman" ></div>
<div id="map4" style="height:300px;"></div>
</div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 1
};
var map = new GCUI.Map('map4', options);
if(document.getElementById('layers').innerHTML == "") {
var layerSwitcher  = new GCUI.Control.LayerSwitcher({
        target: document.getElementById('layers'),
        noOptions: true,
        noDrag: true,
        noMenu: true
});
map.addControl(layerSwitcher);
  var appId = 'kDm0Jq1K4Ak7Bwtn8uvk';
  var appCode = 'xnmvc4dKZrDfGlvQHXSvwQ';
  var hereLayers = [
```

```
    {
      base: 'base',
      type: 'maptile',
      scheme: 'normal.day',
      app_id: appId,
      app_code: appCode
    },
    {
      base: 'base',
      type: 'maptile',
      scheme: 'normal.day.transit',
      app_id: appId,
      app_code: appCode
    },
    {
      base: 'base',
      type: 'maptile',
      scheme: 'pedestrian.day',
      app_id: appId,
      app_code: appCode
    },
    {
      base: 'aerial',
      type: 'maptile',
      scheme: 'terrain.day',
      app_id: appId,
      app_code: appCode
    },
    {
      base: 'aerial',
      type: 'maptile',
      scheme: 'satellite.day',
      app_id: appId,
      app_code: appCode
    },
    {
      base: 'aerial',
      type: 'maptile',
      scheme: 'hybrid.day',
      app_id: appId,
      app_code: appCode
    }
  ];
  var urlTpl = 'https://{1-4}.{base}.maps.cit.api.here.com' +
    '/{type}/2.1/maptile/newest/{scheme}/{z}/{x}/{y}/256/png' +
    '?app_id={app_id}&app_code={app_code}';
  var layers = [];
  var layerDesc = hereLayers[0];
  layers.push(new ol.layer.Tile({
          name: 'HERE-NORMAL',
      visible: false,
      preload: Infinity,
      source: new ol.source.XYZ({
        url: createUrl(urlTpl, layerDesc),
        attributions: 'Map Tiles &copy; ' + new Date().getFullYear() + ' ' +
          '<a href="http://developer.here.com">HERE</a>'
      })
    }));
  map.addLayer(layers[0]);
  var layerDesc = hereLayers[1];
  layers.push(new ol.layer.Tile({
          name: 'HERE-TRANSIT',
      visible: false,
```

```
      preload: Infinity,
      source: new ol.source.XYZ({
        url: createUrl(urlTpl, layerDesc),
        attributions: 'Map Tiles &copy; ' + new Date().getFullYear() + ' ' +
          '<a href="http://developer.here.com">HERE</a>'
      })
    }));
  map.addLayer(layers[1]);
  var layerDesc = hereLayers[2];
  layers.push(new ol.layer.Tile({
          name: 'HERE-PEDASTRIAN',
      visible: false,
      preload: Infinity,
      source: new ol.source.XYZ({
        url: createUrl(urlTpl, layerDesc),
        attributions: 'Map Tiles &copy; ' + new Date().getFullYear() + ' ' +
          '<a href="http://developer.here.com">HERE</a>'
      })
    }));
  map.addLayer(layers[2]);
  var layerDesc = hereLayers[3];
  layers.push(new ol.layer.Tile({
          name: 'HERE-TERRAIN',
      visible: false,
      preload: Infinity,
      source: new ol.source.XYZ({
        url: createUrl(urlTpl, layerDesc),
        attributions: 'Map Tiles &copy; ' + new Date().getFullYear() + ' ' +
          '<a href="http://developer.here.com">HERE</a>'
      })
    }));
  map.addLayer(layers[3]);
  var layerDesc = hereLayers[4];
  layers.push(new ol.layer.Tile({
          name: 'HERE-SATELLITE',
      visible: false,
      preload: Infinity,
      source: new ol.source.XYZ({
        url: createUrl(urlTpl, layerDesc),
        attributions: 'Map Tiles &copy; ' + new Date().getFullYear() + ' ' +
          '<a href="http://developer.here.com">HERE</a>'
      })
    }));
  map.addLayer(layers[4]);
  var layerDesc = hereLayers[5];
  layers.push(new ol.layer.Tile({
          name: 'HERE-HYBRID',
      visible: false,
      preload: Infinity,
      source: new ol.source.XYZ({
        url: createUrl(urlTpl, layerDesc),
        attributions: 'Map Tiles &copy; ' + new Date().getFullYear() + ' ' +
          '<a href="http://developer.here.com">HERE</a>'
      })
    }));
  map.addLayer(layers[5]);
}
else{
map.refresh();
}
      function createUrl(tpl, layerDesc) {
        return tpl
          .replace('{base}', layerDesc.base)
```

```
            .replace('{type}', layerDesc.type)
            .replace('{scheme}', layerDesc.scheme)
            .replace('{app_id}', layerDesc.app_id)
            .replace('{app_code}', layerDesc.app_code);
      }
```

Add a GeoJSON

To add a GeoJSON file to the existing Geoconcept map, follow the below steps:

- Define the look and feel of the GeoJSON object for different types of features like point, line and polygon

```
var fill = new ol.style.Fill({
              color: [0, 70, 135, 0.9]
       });
var stroke = new ol.style.Stroke({
              color: [255, 255, 255, 1],
              width: 2
      });
 var image = new ol.style.Circle({
       radius: 5,
       fill: fill,
       stroke: stroke
     });

 var styles = {
    'Point': new ol.style.Style({
      image: image
    }),
    'LineString': new ol.style.Style({
      stroke: new ol.style.Stroke({
        color: 'red',
        width: 1
      })
    })
  };
```

- Define the GeoJSON object with different geographical types

```
var geojsonObject = {
        'type': 'FeatureCollection',
        'crs': {
          'type': 'name',
          'properties': {
            'name': 'EPSG:3857'
          }
        },
        'features': [{
          'type': 'Feature',
          'geometry': {
            'type': 'Point',
            'coordinates': [260803, 6250829]
          }
        }, {
          'type': 'Feature',
          'geometry': {
            'type': 'Point',
            'coordinates': [8937009.2400000002, 1469287.27]
          }
        },
```

```
                {
          'type': 'Feature',
          'geometry': {
            'type': 'Point',
            'coordinates': [13522429.61, 3664134.81]
          }
        },
                {
          'type': 'Feature',
          'geometry': {
            'type': 'Point',
            'coordinates': [-8927129.20, 2969626.93]
          }
        },
                {
          'type': 'Feature',
          'geometry': {
            'type': 'LineString',
            'coordinates': [[-8927129.20, 2969626.93], [260803, 6250829], [8937009.2400000002, 1469287.27],
 [13522429.61, 3664134.81]]
          }
        }]
};
```

- Create a vector source and set the features of the vector source to the GeoJSON object created

```
var vectorSource = new ol.source.Vector({
    features: (new ol.format.GeoJSON()).readFeatures(geojsonObject)
  });
```

- Create a vector layer and set the source of the layer to the vector source created and add it to the existing Geoconcept map

```
var vectorLayer = new GCUI.Layer.StandardVector({
    source: vectorSource,
    style: styleFunction
  });
  map.addLayer(vectorLayer);
```

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css">
<div id="map5" style="height:300px;"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829
};
var map = new GCUI.Map('map5', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
var map = GCUI.getMap('map5');
var fill = new ol.style.Fill({
        color: [0, 70, 135, 0.9]
        });
var stroke = new ol.style.Stroke({
```

```
          color: [255, 255, 255, 1],
      width: 2
         });
  var image = new ol.style.Circle({
          radius: 5,
          fill: fill,
          stroke: stroke
        });

        var styles = {
          'Point': new ol.style.Style({
            image: image
          }),
          'LineString': new ol.style.Style({
            stroke: new ol.style.Stroke({
              color: 'red',
              width: 1
            })
          })
        };

        var styleFunction = function(feature) {
          return styles[feature.getGeometry().getType()];
        };

  var geojsonObject = {
          'type': 'FeatureCollection',
          'crs': {
            'type': 'name',
            'properties': {
              'name': 'EPSG:3857'
            }
          },
          'features': [{
            'type': 'Feature',
            'geometry': {
              'type': 'Point',
              'coordinates': [260803, 6250829]
            }
          }, {
            'type': 'Feature',
            'geometry': {
              'type': 'Point',
              'coordinates': [8937009.2400000002, 1469287.27]
            }
          },
                  {
            'type': 'Feature',
            'geometry': {
              'type': 'Point',
              'coordinates': [13522429.61, 3664134.81]
            }
          },
                  {
            'type': 'Feature',
            'geometry': {
              'type': 'Point',
              'coordinates': [-8927129.20, 2969626.93]
            }
          },
                  {
            'type': 'Feature',
            'geometry': {
```

```
            'type': 'LineString',
            'coordinates': [[-8927129.20, 2969626.93], [260803, 6250829], [8937009.2400000002, 1469287.27],
 [13522429.61, 3664134.81]]
          }
        }]
      };
var vectorSource = new ol.source.Vector({
    features: (new ol.format.GeoJSON()).readFeatures(geojsonObject)
  });

var vectorLayer = new GCUI.Layer.StandardVector({
    source: vectorSource,
    style: styleFunction
  });
map.addLayer(vectorLayer);
map.zoomToExtent(vectorSource.getExtent());
}
```

Add a Heatmap layer

To add a Heatmap layer to the existing Geoconcept map, follow the below steps:

- Create a new instance of **`ol.layer.Heatmap`** and push the GIS data as its vector source and set the blur and the radius parameters according to your requirement.

```
var vector = new ol.layer.Heatmap({
  source: new ol.source.Vector({
        url: 'https://raw.githubusercontent.com/openlayers/openlayers/master/examples/data/
kml/2012_Earthquakes_Mag5.kml',
        format: new ol.format.KML({
          extractStyles: false
        })
  }),
  blur: parseInt(15, 10),
  radius: parseInt(7, 10)
});
```

- Add the Heatmap vector layer to the existing Geoconcept map

```
vector.getSource().on('addfeature', function(event) {
  var name = event.feature.get('name');
  var magnitude = parseFloat(name.substr(2));
  event.feature.set('weight', magnitude - 5);
});
map.addLayer(vector);
```

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css">
<div id="map6" style="height:300px;"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 10
};
```

```
var map = new GCUI.Map('map6', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
var vector = new ol.layer.Heatmap({
  source: new ol.source.Vector({
        url: 'https://raw.githubusercontent.com/openlayers/openlayers/master/examples/data/
kml/2012_Earthquakes_Mag5.kml',
        format: new ol.format.KML({
          extractStyles: false
        })
  }),
  blur: parseInt(15, 10),
  radius: parseInt(7, 10)
});

vector.getSource().on('addfeature', function(event) {
  // 2012_Earthquakes_Mag5.kml stores the magnitude of each earthquake in a
  // standards-violating <magnitude> tag in each Placemark.  We extract it from
  // the Placemark's name instead.
  var name = event.feature.get('name');
  var magnitude = parseFloat(name.substr(2));
  event.feature.set('weight', magnitude - 5);
});
map.addLayer(vector);
}
```

Geolocation

Locate yourself

To know to your current position in the map, follow the below steps:

- Create a new instance of **`ol.Geolocation`** class and set the trackingOptions enableHighAccuracy to true

```
var geolocation = new ol.Geolocation({
        trackingOptions: {
        enableHighAccuracy: true
        },
        projection: map.getView().getProjection()
});
```

- To enable your browser to detect your location set tracking to true

```
geolocation.setTracking(true);
```

- To get an approximate buffer area around your location create a new instance of **`ol.Feature`** class and set the geometry of the feature to geolocation accuracy geometry

```
var accuracyFeature = new ol.Feature();
geolocation.on('change:accuracyGeometry', function() {
accuracyFeature.setGeometry(geolocation.getAccuracyGeometry());
});
```

- Listen to the change position event on geolocation object and get the coordinates and push features to the new instance of **`GCUI.Layer.StandardVector`** class

```
geolocation.on('change:position', function() {
```

```
var coordinates = geolocation.getPosition();
positionFeature.setGeometry(coordinates ? new ol.geom.Point(coordinates) : null);
var vector_layer = new GCUI.Layer.StandardVector({
source: new ol.source.Vector({
features: [accuracyFeature,positionFeature]
})
});
```

HTML

```
<div style="position: absolute; z-index: 5000; left: 65px;">
        <button type="button" onclick="GCUI.locate.geoLocate()">Geolocate</button>
</div>
<div id="maploc" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD',
    x : 260803,
    y : 6250829,
    scale : 1
};
var map = new GCUI.Map('maploc', options);
GCUI.locate = {
geoLocate : function() {
        var geolocation = new ol.Geolocation({
// enableHighAccuracy must be set to true to have the heading value.
        trackingOptions: {
        enableHighAccuracy: true
        },
        projection: map.getView().getProjection()
        });
        geolocation.setTracking(true);
        var positionFeature = new ol.Feature();
        positionFeature.setStyle(new ol.style.Style({
        image: new ol.style.Circle({
        radius: 6,
        fill: new ol.style.Fill({
        color: '#3399CC'}),
        stroke: new ol.style.Stroke({
        color: '#fff',
        width: 2
        })
        })
        }));
        var accuracyFeature = new ol.Feature();
        geolocation.on('change:accuracyGeometry', function() {
        accuracyFeature.setGeometry(geolocation.getAccuracyGeometry());
        });
        geolocation.on('change:position', function() {
        var coordinates = geolocation.getPosition();
        positionFeature.setGeometry(coordinates ? new ol.geom.Point(coordinates) : null);
        var vector_layer = new GCUI.Layer.StandardVector({
        source: new ol.source.Vector({
        features: [accuracyFeature,positionFeature]
        })
        });
        map.addLayer(vector_layer);
        map.zoomToExtent(positionFeature.getGeometry().getExtent());
        map.getView().setZoom(10);
```

```
        });
}
};
```

Create a buffer around a point

To generate a buffer around a point, follow the below steps:

- Create a new instance of the `ol.Feature` class and set the geometry of the feature to a circle geometry and push the buffer area which needs to be created around that point

```
var circleBuffer = new GCUI.ol.Feature({
        geometry: new GCUI.ol.geom.Circle([258353.06,6240408.91], 4000)
});
```

- Set the style of the buffer according to your interest

```
var buffer_style = [new ol.style.Style({
        fill: new ol.style.Fill({
        color: 'rgba(255,255,255,0.03)'
}),
        stroke: new ol.style.Stroke({
        width: 2
})})
];
```

- Set the style of the `ol.Feature` object to the style created and add the feature to the vector source.

```
circleBuffer.setStyle(buffer_style);
vectorSource.addFeature(circleBuffer);
```

HTML

```
<div id="map1" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map1', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
                                var vectorSource = new ol.source.Vector({});
                                var point_feature_1 = new ol.Feature({
                                geometry: new ol.geom.Point([258353.06,6240408.91]),
                                Name: "Groupe Geoconcept",
                                Address: "152 Avenue Aristide Briand, 92220 Bagneux, France"
                                });
                        vectorSource.addFeature(point_feature_1);
                        vector_layer = new GCUI.Layer.StandardVector({
                                source: vectorSource
                                        });
                                map.addLayer(vector_layer);
//Set the source of the point style to a custom image
                                var point_style = new ol.style.Style({
                                        image: new ol.style.Icon({
                                        crossOrigin: 'anonymous',
```

```
                                        src: 'https://i.ibb.co/SBRmrF5/show-Image.png'
                                        })
                                });
                                var buffer_style = [new ol.style.Style({
                                        fill: new ol.style.Fill({
                                        color: 'rgba(255,255,255,0.03)'
                                        }),
                                        stroke: new ol.style.Stroke({
                                        color: 'rgba(30,144,255,1)',
                                        width: 2
                                        })})
                                        ];
                                        var circleBuffer = new GCUI.ol.Feature({
                                        geometry: new GCUI.ol.geom.Circle([258353.06,6240408.91], 4000)
                                        });
                                circleBuffer.setStyle(buffer_style);
                                vectorSource.addFeature(circleBuffer);
                                vector_layer.setStyle(point_style);
                                extent = vectorSource.getExtent();
                                map.getView().fit(extent);
                                map.getView().setZoom(12);
}
```

Generate an Isochrone/Isodistance

To generate an Isochrone/Isodistance from a location, follow the below steps:

- Create a isochrone/isodistance webservice request and pass the start coordinates, distance/time for which the isonchrone/isodistance needs to be generated and method to get back a isochrone/isodistance JSON file

```
const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/isochrone/v4.json?location=' +
 location + '&distance=' + distance + '&time=' + time + '&method=' + method + '&appkey=' + apikey +
 '&apptoken=' + apptoken';
```

- Read the Isochrone/Isodistance WKT geometry from the response received. Reproject it to the map projection if required.

```
isochrone_feature = ol.format.WKT.prototype.readFeature(parsedjson.wktGeometry,{
        dataProjection: 'EPSG:4326',
        featureProjection: 'EPSG:3857'
        });
```

- Create a vector layer and push the isochrone/isodistance feature created.

```
vector_layer = new GCUI.Layer.StandardVector({
        source: new ol.source.Vector({
        features: [isochrone_feature]
        })
});
```

- Add the vector layer to the existing map

```
map.addLayer(vector_layer);
```

- Define the style of the point using `ol.style.Fill` and `ol.style.Stroke` classes

```
var fill = new ol.style.Fill({
             color: [158, 100, 213, 0.5]
```

```
        });
var stroke = new ol.style.Stroke({
              color: [158, 100, 213, 0.5],
              width: 2
       });
```

- Create a new instance of the `ol.style.Style` class and set the style of the layer using `vector_layer.setStyle()`

```
var style = new ol.style.Style({
               image: new ol.style.Circle({
               fill: fill,
               stroke: stroke,
               radius: 5
         }),
               fill: fill,
               stroke: stroke
         });
vector_layer.setStyle(style);
```

HTML

```
<div style="position: absolute; z-index: 5000; left: 65px;">
        <input type="text" id="location" placeholder="Location" value="2.321134,48.796575">
        <input type="text" id="distance" placeholder="Distance" value="10000">
        <input type="text" id ="time" placeholder="Time" value="1800">
        <input type="text" id ="method" placeholder="Method" value="time">
        <button type="button" onclick="GCUI.isoexamples.getIsochrone()">Generate Isochrone</button>
</div>
<div id="mapDiv1" style="height:300px"></div>
```

JavaScript

```
var svg = '<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"><path
 d="M12 0c-4.198 0-8 3.403-8 7.602 0 4.198 3.469 9.21 8 16.398 4.531-7.188 8-12.2 8-16.398
 0-4.199-3.801-7.602-8-7.602zm0 11c-1.657 0-3-1.343-3-3s1.343-3 3-3 3 1.343 3 3-1.343 3-3 3z"/></svg>';
var options = {
           server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
           layer : 'STANDARD',
           x : 260803,
           y : 6250829,
           scale : 8
              };
var map = new GCUI.Map('mapDiv1', options);
GCUI.isoexamples = {
getIsochrone    : function () {
                              var location = document.getElementById("location").value;
                              var distance = document.getElementById("distance").value;
                              var time = document.getElementById("time").value;
                              var method = document.getElementById("method").value;
                              const request = new XMLHttpRequest();
                              const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/
isochrone/v4.json?location=' + location + '&distance=' + distance + '&time=' + time + '&method=' + method +
 '&appkey=' + $("input[name=apikey]").val() + '&apptoken=' + $("input[name=apitoken]").val();
                              request.open("GET", url);
                              request.send();
// Treatment of the output JSON to display the information on the popup
                              request.onreadystatechange=(e)=>{
                        if(request.readyState == 4 && request.status == 200)
                        {
                        var responseJson = request.responseText;
```

```
                        parsedjson = JSON.parse(responseJson);
                        if (map.getLayers().array_.length > 1){
                                location_pointlayer.getSource().removeFeature(location_feature);
                                vector_layer.getSource().removeFeature(isochrone_feature);
                                                                }
// Adding a vector layer to the map
                                var location_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(location.split(',').shift()),parseFloat(location.split(',').pop())]));
                                location_feature = new ol.Feature({});
                                location_feature.setGeometry(location_geom);
                                location_pointlayer = new GCUI.Layer.StandardVector({
                                        source: new ol.source.Vector({
                                                features: [location_feature]
                                                })
                                                });
                                map.addLayer(location_pointlayer);
                                isochrone_feature =
 ol.format.WKT.prototype.readFeature(parsedjson.wktGeometry,{
                                dataProjection: 'EPSG:4326',
                                featureProjection: 'EPSG:3857'
                                });
                                vector_layer = new GCUI.Layer.StandardVector({
                                        source: new ol.source.Vector({
                                                features: [isochrone_feature]
                                                })
                                                });
                                map.addLayer(vector_layer);
//Adding a style to the vector layer
                                var location_style = new ol.style.Style({
                                  image: new ol.style.Icon({
                                  crossOrigin: 'anonymous',
                                  src: 'https://i.ibb.co/cCJjRvj/Finish.png'
                                        })
                                });
                                var fill = new ol.style.Fill({
                                  color: [158, 100, 213, 0.5]
                                });
                                var stroke = new ol.style.Stroke({
                                  color: [158, 100, 213, 0.5],
                                  width: 5
                                });
                                var style = new ol.style.Style({
                                  image: new ol.style.Circle({
                                        fill: fill,
                                        stroke: stroke,
                                        radius: 5
                                  }),
                                  fill: fill,
                                  stroke: stroke
                                });
                                location_pointlayer.setStyle(location_style);
                                vector_layer.setStyle(style);
                                extent = isochrone_feature.getGeometry().getExtent();
                                map.getView().fit(extent);
                }
                }
}
}
```

Search Around

To search around and find the closest resource , follow the below steps:

- Define the start location

```
var location_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(location.split(',').shift()),parseFloat(location.split(',').pop())]));
location_feature = new ol.Feature({Name: "Start Location",id:"0",distanceMeters:0});
location_feature.setGeometry(location_geom);
location_pointlayer = new GCUI.Layer.StandardVector({
        source: new ol.source.Vector({
                features: [location_feature]
                })
                });
```

- Define the list of finish locations whose distance needs to be calculated

```
var resource1_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource1.split(',').shift()),parseFloat(resource1.split(',').pop())]));
                resource1_feature = new ol.Feature({Name: "Resource1",id:"1",distanceMeters:0});
                resource1_feature.setGeometry(resource1_geom);
var resource2_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource2.split(',').shift()),parseFloat(resource2.split(',').pop())]));
                resource2_feature = new ol.Feature({Name: "Resource2",id:"2",distanceMeters:0});
                resource2_feature.setGeometry(resource2_geom);
var resource3_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource3.split(',').shift()),parseFloat(resource3.split(',').pop())]));
                resource3_feature = new ol.Feature({Name: "Resource3",id:"3",distanceMeters:0});
                resource3_feature.setGeometry(resource3_geom);
var resource4_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource4.split(',').shift()),parseFloat(resource4.split(',').pop())]));
                resource4_feature = new ol.Feature({Name: "Resource4",id:"4",distanceMeters:0});
                resource4_feature.setGeometry(resource4_geom);
var resource5_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource5.split(',').shift()),parseFloat(resource5.split(',').pop())]));
                resource5_feature = new ol.Feature({Name: "Resource5",id:"5",distanceMeters:0});
                resource5_feature.setGeometry(resource5_geom);
                resource_pointlayer = new GCUI.Layer.StandardVector({
                        source: new ol.source.Vector({
                                features:
 [resource1_feature,resource2_feature,resource3_feature,resource4_feature,resource5_feature]
                                })
                                });
```

- Construct the BODY of the request with the start location and the list of resources and POST it to searchAround webservices to get the response.

```
const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/searchAround/v4.json?' + '&appkey=' +
 appkey + '&apptoken=' + apptoken;

data = '{"location": {"x":' + parseFloat(location.split(',').shift()) + ',"y":' +
 parseFloat(location.split(',').pop()) + '},"resources": [{"id":' + 1 + ',"x":' +
 parseFloat(resource1.split(',').shift()) + ',"y":' +  parseFloat(resource1.split(',').pop())
 + '},{"id": ' + 2 + ',"x":' + parseFloat(resource2.split(',').shift()) +
 ',"y":' +  parseFloat(resource2.split(',').pop()) + '},{"id": ' + 3 + ',"x":' +
 parseFloat(resource3.split(',').shift()) + ',"y":' +  parseFloat(resource3.split(',').pop())
 + '},{"id": ' + 4 + ',"x":' + parseFloat(resource4.split(',').shift()) +
 ',"y":' +  parseFloat(resource4.split(',').pop()) + '},{"id": ' + 5 + ',"x":' +
 parseFloat(resource5.split(',').shift()) + ',"y":' +  parseFloat(resource5.split(',').pop()) +
 '}],"method":"time","reverse": "false"}';
```

- Treat the response and display the results on to the existing map.

```
vector_layer = new GCUI.Layer.StandardVector({
```

```
        source: new ol.source.Vector({
                features: search_around_points
                }),
        style: searchStyle
                });
map.addLayer(vector_layer);
```

HTML

```
<style>
.anchorright#ol-popup {
    right: -30px;
}

.anchorright#ol-popup:after {
    right: 20px;
}

.anchorright#ol-popup:before {
    right: 20px;
}

.anchorbottom#ol-popup {
    bottom: 11px;
}

.anchorbottom#ol-popup:after, .anchorbottom#ol-popup:before {
    top: 100%;
}

#ol-popup-content.content {
    max-height: 25em;
    overflow-y: auto;
}

.#ol-popup{
    padding-top: 2em;
    padding-right: 0.35em;
}

.anchorbottom#ol-popup:after {
    border-top-color: white;
}

.anchorbottom#ol-popup:before {
    border-top-color: #cccccc;
}

.anchorleft#ol-popup {
    left: -25px;
}

.anchorleft#ol-popup:after {
    left: 25px;
}

.anchorleft#ol-popup:before {
    left: 25px;
}

.anchortop#ol-popup {
    top: 11px;
}
```

```
.anchortop#ol-popup:after, .anchortop#ol-popup:before {
    bottom: 100%;
}

.anchortop#ol-popup:after {
    border-bottom-color: white;
}

.anchortop#ol-popup:before {
    border-bottom-color: #cccccc;
}

#ol-popup {
    position: absolute;
    background-color: white;
    -webkit-filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    padding: 15px;
    border-radius: 5px;
    border: 1px solid #cccccc;
    min-width: 280px;
}

#ol-popup:after, #ol-popup:before {
    border: solid transparent;
    content: " ";
    height: 0;
    width: 0;
    position: absolute;
    pointer-events: none;
}

#ol-popup:after {
    border-width: 10px;
    margin-left: -10px;
}

#ol-popup:before {
    border-width: 11px;
    margin-left: -11px;
}

#ol-popup-closer {
    text-decoration: none;
    position: absolute;
    top: 2px;
    right: 8px;
    font-family: Icons;
}

#ol-popup-closer:hover {
    cursor: pointer;
    color: #004687;
}

#ol-popup-closer:after {
  content: "#";
}

#ol-popup .pages {
        text-align: right;
    position: relative;
```

```
    right: -1em;
    bottom: -0.8em;
}

#ol-popup .pages i {
    cursor: pointer;
}

#ol-popup .pages > * {
    display: inline-block;
}

#ol-popup .pages i.disabled {
    cursor: default;
}

.gcuiAnchoredContentData {
    height: 100%;
    width: 100%;
    overflow-y: auto;
    overflow-x: hidden;
}

.gcuiAnchoredContentData .table-width {
    width: 50%;
}

.gcuiAnchoredContentData .ui.table  .right.aligned .content {
    text-align: right;
    margin-right: .2em;
    color: #808080;
}

.gcuiAnchoredContentData .ui.table .content {
    color: #000000;
    border: none !important;
}

.gcuiAnchoredContentData .ui.table td {
    padding: .25em;
    border: none !important;
}
.gcuiAnchoredContentData .ui.table, th, td {
         border: none !important;
}
.ol-custom-overviewmap,
.ol-custom-overviewmap.ol-uncollapsible {
        bottom: auto;
        left: auto;
        right: 0;
        top: 0;
}
.ol-custom-overviewmap:not(.ol-collapsed)  {
        border: 1px solid black;
}
.ol-custom-overviewmap .ol-overviewmap-map {
        border: none;
        width: 300px;
}
.ol-custom-overviewmap .ol-overviewmap-box {
        border: 2px solid red;
}
.ol-custom-overviewmap:not(.ol-collapsed) button{
```

```
        bottom: auto;
        left: auto;
        right: 1px;
        top: 1px;
}
.ol-rotate {
        top: 170px;
        right: 0;
}
</style>
<div style="position: absolute; z-index: 5000; left: 15px;">
        <input type="text" id="slocation" placeholder="Start Location" value="-1.553927,47.21858" size="15">
        <input type="text" id="Resource1" placeholder="Resource1" value="-1.576829,47.224742" size="15">
        <input type="text" id ="Resource2" placeholder="Resource2" value="-1.544900,47.220649" size="15">
        <input type="text" id ="Resource3" placeholder="Resource3" value="-1.534977,47.210036" size="15">
        <input type="text" id ="Resource4" placeholder="Resource4" value="-1.550059,47.224586" size="15">
        <input type="text" id ="Resource5" placeholder="Resource5" value="-1.577197,47.202243"
 size="15"><br>
        <button type="button" onclick="GCUI.examples.searchAround()" >Search Around</button>
</div>
<div id="mapDiv2" style="height:300px"></div>
```

JavaScript

```
var map;
var resource_fill;
var resource_stroke;
var resource_text;
var data;
var final_data;
var input_data = null;
var search_around_points = [];
var vector_layer;
var options = {
           server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
           layer : 'STANDARD',
           x : 260803,
           y : 6250829,
           scale : 8
              };
var map = new GCUI.Map('mapDiv2', options);
map.onEvent("load", onMapLoaded);
function onMapLoaded() {
        var location = document.getElementById("slocation").value;
        var resource1 = document.getElementById("Resource1").value;
        var resource2 = document.getElementById("Resource2").value;
        var resource3 = document.getElementById("Resource3").value;
        var resource4 = document.getElementById("Resource4").value;
        var resource5 = document.getElementById("Resource5").value;
        var location_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(location.split(',').shift()),parseFloat(location.split(',').pop())]));
                    location_feature = new ol.Feature({Name: "Start Location",id:"0",distanceMeters:0});
                    location_feature.setGeometry(location_geom);
                    location_pointlayer = new GCUI.Layer.StandardVector({
                            source: new ol.source.Vector({
                                    features: [location_feature]
                                    })
                                    });
        var resource1_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource1.split(',').shift()),parseFloat(resource1.split(',').pop())]));
                    resource1_feature = new ol.Feature({Name: "Resource1",id:"1",distanceMeters:0});
                    resource1_feature.setGeometry(resource1_geom);
```

```
        var resource2_geom = new
ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource2.split(',').shift()),parseFloat(resource2.split(',').pop())]));
                    resource2_feature = new ol.Feature({Name: "Resource2",id:"2",distanceMeters:0});
                    resource2_feature.setGeometry(resource2_geom);
        var resource3_geom = new
ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource3.split(',').shift()),parseFloat(resource3.split(',').pop())]));
                    resource3_feature = new ol.Feature({Name: "Resource3",id:"3",distanceMeters:0});
                    resource3_feature.setGeometry(resource3_geom);
        var resource4_geom = new
ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource4.split(',').shift()),parseFloat(resource4.split(',').pop())]));
                    resource4_feature = new ol.Feature({Name: "Resource4",id:"4",distanceMeters:0});
                    resource4_feature.setGeometry(resource4_geom);
        var resource5_geom = new
ol.geom.Point(ol.proj.fromLonLat([parseFloat(resource5.split(',').shift()),parseFloat(resource5.split(',').pop())]));
                    resource5_feature = new ol.Feature({Name: "Resource5",id:"5",distanceMeters:0});
                    resource5_feature.setGeometry(resource5_geom);
                    resource_pointlayer = new GCUI.Layer.StandardVector({
                            source: new ol.source.Vector({
                                    features:
[resource1_feature,resource2_feature,resource3_feature,resource4_feature,resource5_feature]
                                    })
                                    });
        var location_fill = new ol.style.Fill({
                  color: [71, 176, 67, 0.9]
        });
        resource_fill = new ol.style.Fill({
                  color: 'red'
        });
        var location_stroke = new ol.style.Stroke({
                  color: [255, 255, 255, 1],
                  width: 2
    });
    resource_stroke = new ol.style.Stroke({
                  color: [255, 255, 255, 1],
                  width: 2
    });
    var location_text = new ol.style.Text({
                font: '10px Open Sans',
                text: 'S',
                fill: new ol.style.Fill({
                        color: [255, 255, 255, 1]
        })
        });
        resource_text = new ol.style.Text({
                font: '10px Open Sans',
                text: 'F',
                fill: new ol.style.Fill({
                        color: [255, 255, 255, 1]
        })
        });
        var location_style = new ol.style.Style({
                    image: new ol.style.Circle({
                    fill: location_fill,
                    stroke: location_stroke,
                    text:location_text,
                    radius: 10
         }),
                    fill: location_fill,
                    stroke: location_stroke,
                    text:location_text
         });
         var resource_style = new ol.style.Style({
                    image: new ol.style.Circle({
```

```
                    fill: resource_fill,
                    stroke: resource_stroke,
                    text:resource_text,
                    radius: 10
         }),
                    fill: resource_fill,
                    stroke: resource_stroke,
                    text:resource_text
         });
     location_pointlayer.setStyle(location_style);
         resource_pointlayer.setStyle(resource_style);
     map.addLayer(location_pointlayer);
         map.addLayer(resource_pointlayer);
         map.zoomToExtent(resource_pointlayer.getSource().getExtent());
         //select interaction working on "click"
        var selectClick = new ol.interaction.Select({
          condition: ol.events.condition.click
        });
         if (selectClick !== null) {
                        map.addInteraction(selectClick);
                        selectClick.on('select', function(e) {
                          var feature = e.target.getFeatures().getArray();
                          if (feature.length >= 1) {
                          var opt_options = "<div class='gcuiAnchoredContentData'><table class='ui very
 basic table unstackable' cellspacing='0' cellpadding='0'><tbody><tr><td class='right aligned'><div
 class='content'>Name</div></td><td><div class='content'>" + feature[0].values_.Name + "</div></td></
tr><tr><td class='right aligned'><div class='content'>Identifier</div></td><td><div class='content'>"
 + feature[0].values_.id + "</div></td></tr>" + "</div></td></tr><tr><td class='right aligned'><div
 class='content'>DistanceInMeters</div></td><td><div class='content'>" + feature[0].values_.distanceMeters +
 "</div></td></tr></tbody></table></div>";
                          var popup = new GCUI.Control.Popup(map,
 feature[0].values_.geometry.flatCoordinates, {content : opt_options});
                          map.addOverlay(popup);
                          popup.initialize(feature[0].values_.geometry.flatCoordinates);
                          }
                        });
  }
}
GCUI.examples = {
searchAround    : function () {
if (search_around_points.length > 1){
                        search_around_points = [];
                        vector_layer.getSource().clear();
                        }
                        var location = document.getElementById("slocation").value;
                        var resource1 = document.getElementById("Resource1").value;
                        var resource2 = document.getElementById("Resource2").value;
                        var resource3 = document.getElementById("Resource3").value;
                        var resource4 = document.getElementById("Resource4").value;
                        var resource5 = document.getElementById("Resource5").value;
                        const search_url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/
api/lbs/searchAround/v4.json?' + '&appkey=' + $("input[name=apikey]").val() + '&apptoken=' +
 $("input[name=apitoken]").val();
                        post_body = '{"location": {"x":' + parseFloat(location.split(',').shift())
 + ',"y":' + parseFloat(location.split(',').pop()) + '},"resources": [{"id":' + 1 + ',"x":' +
 parseFloat(resource1.split(',').shift()) + ',"y":' +  parseFloat(resource1.split(',').pop())
 + '},{"id": ' + 2 + ',"x":' + parseFloat(resource2.split(',').shift()) +
 ',"y":' +  parseFloat(resource2.split(',').pop()) + '},{"id": ' + 3 + ',"x":' +
 parseFloat(resource3.split(',').shift()) + ',"y":' +  parseFloat(resource3.split(',').pop())
 + '},{"id": ' + 4 + ',"x":' + parseFloat(resource4.split(',').shift()) +
 ',"y":' +  parseFloat(resource4.split(',').pop()) + '},{"id": ' + 5 + ',"x":' +
 parseFloat(resource5.split(',').shift()) + ',"y":' +  parseFloat(resource5.split(',').pop()) +
 '}],"method":"time","reverse": "false"}';
```

```
                        $.ajax({
                        url : search_url,
                        type: 'POST',
                        data: post_body,
                        dataType: "json",
                        contentType:"application/json;charset\x3dutf-8",
                        success: function(result){
                        parsedjson = result;
                        resource_pointlayer.getSource().clear();
                        for(i = 0; i<parsedjson.searchAroundResult.length; i++){
                        parsedjson.searchAroundResult[i].id = parseInt(parsedjson.searchAroundResult[i].id)
                        }
                        final_data =
$.extend(true,JSON.parse(post_body).resources,parsedjson.searchAroundResult.sort(function(a, b){return a.id
- b.id}));
                        final_data.sort(function(a, b){return a.distanceMeters - b.distanceMeters})
                        for (i=0;i<final_data.length;i++){
                        var xy = ol.proj.fromLonLat([final_data[i].x,final_data[i].y]);
                        search_around_points.push(new ol.Feature({
                        geometry: new ol.geom.Point(xy),
                        Name: 'Resource' + (i+1),
                        id: parseInt(final_data[i].id),
                        distanceMeters: final_data[i].distanceMeters,
                        Rank: (i+1)
                        }));
                        }

                        function searchStyle(feature) {
                        var resource_icon = new ol.style.Text({
                                font: '10px Open Sans',
                                text: feature.get('Rank').toString(),
                                fill: new ol.style.Fill({
                                        color: 'white'
                        })
                        });
                          var style = new ol.style.Style({
                                image: new ol.style.Circle({
                                  radius: 10,
                                  text: resource_icon,
                                  stroke: new ol.style.Stroke({
                                        color: 'white',
                                        width: 2
                                  }),
                                  fill: new ol.style.Fill({
                                        color: 'red'
                                  })
                                }),
                                text: resource_icon
                          });
                          return [style];
                        }
                        vector_layer = new GCUI.Layer.StandardVector({
                                source: new ol.source.Vector({
                                        features: search_around_points
                                        }),
                                style: searchStyle
                                        });
                        map.addLayer(vector_layer);

                        },
                        error: function(error){
                        alert(error);
                        }
```

```
                        })
}
}
```

Tracking Animation

To animate a GPS tracking , follow the below steps:

- Define the route in WKT geometry which needs to be tracked and read the WKT feature using the function `ol.format.WKT.prototype.readFeature`
- Reproject it to the map projection if necessary

```
var polyline = "LINESTRING (2.364632 48.864657,2.364579 48.861966,2.364587 48.861336,2.364537
 48.859457,2.364438 48.857838,2.364233 48.856426,2.362948 48.854603,2.366468 48.853520,2.368516
 48.853212,2.367298 48.850843,2.365799 48.847646,2.367620 48.845831,2.364568 48.844123,2.361985
 48.840291,2.358277 48.835208,2.356520 48.832845,2.359491 48.832237,2.365139 48.829188,2.366523
 48.826991,2.369328 48.821730,2.376556 48.816573,2.383493 48.810967,2.396966 48.817444,2.406021
 48.824962,2.419579 48.818368,2.435368 48.808749)";

var line_feature = ol.format.WKT.prototype.readFeature(polyline,{
dataProjection: 'EPSG:4326',
featureProjection: 'EPSG:3857'
});
var routeFeature = new ol.Feature({
        type: 'route',
        geometry: line_feature.getGeometry()
  });
```

- Define the start point, start marker and end point by creating a new instance of the class **`ol.geom.Point`**
- Push the created **`ol.geom.Point`** to **`ol.Feature`**

```
var firstPoint = new ol.geom.Point([263229,6251927]);
geoMarker = new ol.Feature({
type:'geoMarker'});
geoMarker.setGeometry(firstPoint);

startMarker = new ol.Feature({
type:'startMarker'});
startMarker.setGeometry(firstPoint);

var lastPoint = new ol.geom.Point([271103,6242472]);
endMarker = new ol.Feature({
type:'endMarker'});
endMarker.setGeometry(lastPoint);
```

- Push all the features to the vector layer and add the layer to the exiting Geoconcept map.

```
vectorlayer = new GCUI.Layer.StandardVector({
source: new ol.source.Vector({
        features: [geoMarker,routeFeature,startMarker,endMarker]
                }),
                style: function(feature) {
                          // hide geoMarker if animation is active
                          if (animating && feature.get('type') === 'geoMarker') {
                                return null;
                          }
                          return styles[feature.get('type')];
                        }
```

```
        });
map.addLayer(vectorlayer);
```

- Get all the coordinates of the route and the total route length using the below statements.

```
var routeCoords = line_feature.getGeometry().flatCoordinates;
var routeLength = line_feature.getGeometry().flatCoordinates.length;
```

- While animating calculate the elapsed time and create an index.
- Use the index value to move the current point from one vertices to other to show the movement of the location on the map.

```
if (animating) {
var elapsedTime = frameState.time - now;
 var index = Math.round(10 * elapsedTime / 1000);
      if (index >= routeLength) {
    stopAnimation(true);
    return;
   }
  var currentPoint = new ol.geom.Point([routeCoords[index],routeCoords[index+1]]);
  var feature = new ol.Feature(currentPoint);
  vectorContext.drawFeature(feature, styles.geoMarker);
 }
```

HTML

```
<div style="position: absolute; z-index: 5000; left: 65px;">
        <button type="button" id="start-animation">Start Animation</button>
</div>
<div id="mapDiv3" style="height:300px"></div>
```

JavaScript

```
var center =[260803, 6250829];
var options = {
           server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
           layer : 'STANDARD',
           x : 260803,
           y : 6250829,
           scale : 1
              };
var map = new GCUI.Map('mapDiv3', options);
var polyline = "LINESTRING (2.364632 48.864657,2.364579 48.861966,2.364587 48.861336,2.364537
 48.859457,2.364438 48.857838,2.364233 48.856426,2.362948 48.854603,2.366468 48.853520,2.368516
 48.853212,2.367298 48.850843,2.365799 48.847646,2.367620 48.845831,2.364568 48.844123,2.361985
 48.840291,2.358277 48.835208,2.356520 48.832845,2.359491 48.832237,2.365139 48.829188,2.366523
 48.826991,2.369328 48.821730,2.376556 48.816573,2.383493 48.810967,2.396966 48.817444,2.406021
 48.824962,2.419579 48.818368,2.435368 48.808749)";

var line_feature = ol.format.WKT.prototype.readFeature(polyline,{
dataProjection: 'EPSG:4326',
featureProjection: 'EPSG:3857'
});
var routeFeature = new ol.Feature({
        type: 'route',
        geometry: line_feature.getGeometry()
  });
var routeCoords = line_feature.getGeometry().flatCoordinates;
var routeLength = line_feature.getGeometry().flatCoordinates.length;
```

```
var firstPoint = new ol.geom.Point([263229,6251927]);
geoMarker = new ol.Feature({
type:'geoMarker'});
geoMarker.setGeometry(firstPoint);

startMarker = new ol.Feature({
type:'startMarker'});
startMarker.setGeometry(firstPoint);

var lastPoint = new ol.geom.Point([271103,6242472]);
endMarker = new ol.Feature({
type:'endMarker'});
endMarker.setGeometry(lastPoint);

var styles = {
'route': new ol.style.Style({
   stroke: new ol.style.Stroke({
        color: [0, 70, 135, 1],
         width: 6
          })
          }),
'startMarker': new ol.style.Style({
  image: new ol.style.Icon({
   crossOrigin: 'anonymous',
       src: 'https://i.ibb.co/cCJjRvj/Finish.png'
   })
}),
'geoMarker': new ol.style.Style({
    image: new ol.style.Circle({
       fill: new ol.style.Fill({
        color: 'white'
          }),
       stroke: new ol.style.Stroke({
        color: [0, 70, 135, 1],
         width: 6
          }),
         radius: 7
       })
}),
'endMarker': new ol.style.Style({
  image: new ol.style.Icon({
   crossOrigin: 'anonymous',
       src: 'https://i.ibb.co/VjGzprJ/Start.png'
   })
})
};
vectorlayer = new GCUI.Layer.StandardVector({
    source: new ol.source.Vector({
            features: [geoMarker,routeFeature,startMarker,endMarker]
                    }),
                        style: function(feature) {
                                  // hide geoMarker if animation is active
                                  if (animating && feature.get('type') === 'geoMarker') {
                                        return null;
                                  }
                                  return styles[feature.get('type')];
                                }
            });
map.addLayer(vectorlayer);
map.zoomToExtent(vectorlayer.getSource().getExtent());
var animating = false;
var now;
```

```
var startButton = document.getElementById('start-animation');
var moveFeature = function(event) {
var vectorContext = event.vectorContext;
var frameState = event.frameState;

if (animating) {
var elapsedTime = frameState.time - now;
 var index = Math.round(10 * elapsedTime / 1000);
      if (index >= routeLength) {
    stopAnimation(true);
    return;
   }
  var currentPoint = new ol.geom.Point([routeCoords[index],routeCoords[index+1]]);
  var feature = new ol.Feature(currentPoint);
  vectorContext.drawFeature(feature, styles.geoMarker);
  }
map.render();
};
function startAnimation() {
          if (animating) {
      stopAnimation(false);
    } else {
      animating = true;
      now = new Date().getTime();
          startButton.textContent = 'Cancel Animation';
          geoMarker.setStyle(null);
          map.getView().setCenter(center);
      map.on('postcompose', moveFeature);
      map.render();
    }
   }
function stopAnimation(ended) {
    animating = false;
         startButton.textContent = 'Start Animation';
    var coord = ended ? routeCoords.slice(0,2) : routeCoords.slice(0,2);
    /** @type {module:ol/geom/Point~Point} */ (geoMarker.getGeometry())
      .setCoordinates(coord);
    //remove listener
    map.on('postcompose', moveFeature);
        map.render();
  }
startButton.addEventListener('click', startAnimation, false);
```

Geocoding

To geocode an address and to display it on the map, follow the below steps:

- Create a geocoding webservice request and pass the address field to get back a geocoded JSON file

```
const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/geocode/
v2.json?addressLine=' + document.getElementById("address").value + '&postCode=' +
 document.getElementById("postcode").value + '&city=' + document.getElementById("city").value +
 '&countryCode=' + document.getElementById("country").value + '&maxResponses=5&srs=epsg:3857&appkey=' +
 apikey + '&apptoken=' + apptoken;
```

- Create a new instance of **`ol.Feature`** class

```
var point_feature = new ol.Feature({});
```

- Create a new instance of **`ol.geom.Point`** class and push the X and Y

```
var point_geom = new ol.geom.Point([parsedjson.geocodedAddresses[0].x,parsedjson.geocodedAddresses[0].y]);
```

- Set the geometry of the ol feature as the **`ol.geom.Point`** class

```
point_feature.setGeometry(point_geom);
```

- Create a new instance of **`GCUI.Layer.StandardVector`** class and map the source of the vector to the ol feature

```
var vector_layer = new GCUI.Layer.StandardVector({
                    source: new ol.source.Vector({
                    features: [point_feature]
                })
            });
```

- Add the layer to the existing map using `map.addLayer` method and push the vector layer created

```
map.addLayer(vector_layer);
```

- Define the style of the point using **`ol.style.Fill`** and **`ol.style.Stroke`** classes

```
var fill = new ol.style.Fill({
                color: [0, 70, 135, 0.9]
            });
var stroke = new ol.style.Stroke({
                color: [255, 255, 255, 1],
                width: 2
           });
```

- Create a new instance of the **`ol.style.Style`** class and set the style of the layer using `vector_layer.setStyle()`

```
var style = new ol.style.Style({
                 image: new ol.style.Circle({
                 fill: fill,
                 stroke: stroke,
                 radius: 5
             }),
                 fill: fill,
                 stroke: stroke
             });
vector_layer.setStyle(style);
```

HTML

```
<div style="position: absolute; z-index: 5000; left: 65px;">
        <input type="text" id="address" placeholder="Address" value="152-160 avenue Aristide Briand">
        <input type="text" id="city" placeholder="City" value="BAGNEUX">
        <input type="text" id ="postcode" placeholder="Postcode" value="92220">
        <input type="text" id ="country" placeholder="Country" value="FR">
        <button type="button" onclick="GCUI.examples.getGeocode()">Geocode</button>
</div>
<div id="map1" style="height:300px"></div>
```

JavaScript

```
var options = {
           server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
           layer : 'STANDARD',
           x : 260803,
           y : 6250829,
           scale : 8
              };
var map = new GCUI.Map('map1', options);
GCUI.examples = {
getGeocode      : function () {
                        const request = new XMLHttpRequest();
                        const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/
lbs/geocode/v2.json?addressLine=' + document.getElementById("address").value + '&postCode=' +
 document.getElementById("postcode").value + '&city=' + document.getElementById("city").value +
 '&countryCode=' + document.getElementById("country").value + '&maxResponses=5&srs=epsg:3857&appkey=' +
 $("input[name=apikey]").val() + '&apptoken=' + $("input[name=apitoken]").val();
                        request.open("GET", url);
                        request.send();
// Treatment of the output JSON to display the information on the popup
                        request.onreadystatechange=(e)=>{
                        if(request.readyState == 4 && request.status == 200)
                        {
                                var responseJson = request.responseText;
                                parsedjson = JSON.parse(request.responseText);
// Adding a vector layer to the map
                                if (parsedjson.geocodedAddresses.length >= 1) {
                                if (map.getLayers().array_.length > 1){
                                vector_layer.getSource().removeFeature(point_feature);
                                }
                                var point_geom = new
 ol.geom.Point([parsedjson.geocodedAddresses[0].x,parsedjson.geocodedAddresses[0].y]);
                                point_feature = new ol.Feature({});
                                point_feature.setGeometry(point_geom);
                                vector_layer = new GCUI.Layer.StandardVector({
                                        source: new ol.source.Vector({
                                                features: [point_feature]
                                                })
                                                });
                                map.addLayer(vector_layer);
//Adding a style to the vector layer
                                var fill = new ol.style.Fill({
                                  color: [0, 70, 135, 0.9]
                                });
                                var stroke = new ol.style.Stroke({
                                  color: [255, 255, 255, 1],
                                  width: 2
                                });
                                var style = new ol.style.Style({
                                  image: new ol.style.Circle({
                                        fill: fill,
                                        stroke: stroke,
                                        radius: 5
                                  }),
                                  fill: fill,
                                  stroke: stroke
                                });
                                vector_layer.setStyle(style);
                                extent = point_feature.getGeometry().getExtent();
                                if (parsedjson.geocodedAddresses[0].geocodeType > 1)
                                {
                                map.getView().fit(extent);
                                map.getView().setZoom(18);
```

```
                                }
                                else
                                {
                                map.getView().fit(extent);
                                map.getView().setZoom(12);
                        }
                                }
                        }
                }
        }
}
```

Routing

To create a route and to display it on the map, follow the below steps:

- Create a routing webservice request and pass the start, stop and finish coordinates to get back a route JSON file

```
const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/route/v5.json?
origin=' + start + '&destination=' + finish + '&waypoints=' + stop + '&countryCode=' +
 document.getElementById("country").value + '&maxResponses=5&srs=epsg:4326&appkey=' + apikey + '&apptoken='
 + apptoken';
```

- Create a new instance of **`ol.Feature`** class

```
var point_feature = new ol.Feature({});
```

- Create a new instance of **`ol.geom.Point`** class and push the X and Y

```
var point_geom = new ol.geom.Point([parsedjson.geocodedAddresses[0].x,parsedjson.geocodedAddresses[0].y]);
```

- Set the geometry of the ol feature as the **`ol.geom.Point`** class

```
point_feature.setGeometry(point_geom);
```

- Create a new instance of **`GCUI.Layer.StandardVector`** class and map the source of the vector to the ol feature

```
var vector_layer = new GCUI.Layer.StandardVector({
                    source: new ol.source.Vector({
                    features: [point_feature]
                })
        });
```

- Add the layer to the existing map using `map.addLayer` method and push the vector layer created

```
map.addLayer(vector_layer);
```

- Define the style of the point using **`ol.style.Fill`** and **`ol.style.Stroke`** classes

```
var fill = new ol.style.Fill({
                color: [0, 70, 135, 0.9]
        });
var stroke = new ol.style.Stroke({
                color: [255, 255, 255, 1],
                width: 2
```

```
        });
```

- Create a new instance of the `ol.style.Style` class and set the style of the layer using `vector_layer.setStyle()`

```
var style = new ol.style.Style({
               image: new ol.style.Circle({
               fill: fill,
               stroke: stroke,
               radius: 5
         }),
               fill: fill,
               stroke: stroke
         });
vector_layer.setStyle(style);
```

HTML

```
<div style="position: absolute; z-index: 5000; left: 65px;">
        <input type="text" id="start" placeholder="Start" value="2.295048,48.873907">
        <input type="text" id="stop" placeholder="Stop" value="2.435500, 48.842714">
        <input type="text" id ="finish" placeholder="Finish" value="2.346406,48.846230">
        <input type="text" id ="country" placeholder="Country" value="FR">
        <button type="button" onclick="GCUI.examples.getRoute()">Calculate Route</button>
</div>
<div id="mapDiv" style="height:300px"></div>
```

JavaScript

```
var svg = '<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"><path
 d="M12 0c-4.198 0-8 3.403-8 7.602 0 4.198 3.469 9.21 8 16.398 4.531-7.188 8-12.2 8-16.398
 0-4.199-3.801-7.602-8-7.602zm0 11c-1.657 0-3-1.343-3-3s1.343-3 3-3 3 1.343 3 3-1.343 3-3 3z"/></svg>';
var options = {
           server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
           layer : 'STANDARD',
           x : 260803,
           y : 6250829,
           scale : 8
              };
var map = new GCUI.Map('mapDiv', options);
GCUI.examples = {
getRoute        : function () {
                                var start = document.getElementById("start").value;
                                var finish = document.getElementById("finish").value;
                                var stop = document.getElementById("stop").value;
                                const request = new XMLHttpRequest();
                                const url = 'https://api.geoconcept.com/EU/GCW/geoconcept-web/
api/lbs/route/v5.json?origin=' + start + '&destination=' + finish + '&waypoints=' + stop +
 '&countryCode=' + document.getElementById("country").value + '&maxResponses=5&srs=epsg:4326&appkey=' +
 $("input[name=apikey]").val() + '&apptoken=' + $("input[name=apitoken]").val();
                                request.open("GET", url);
                                request.send();
// Treatment of the output JSON to display the information on the popup
                                request.onreadystatechange=(e)=>{
                        if(request.readyState == 4 && request.status == 200)
                        {
                        var responseJson = request.responseText;
                        parsedjson = JSON.parse(responseJson);
                        if (map.getLayers().array_.length > 1){
                                origin_pointlayer.getSource().removeFeature(start_feature);
                                destn_pointlayer.getSource().removeFeature(end_feature);
```

```
                                vector_layer.getSource().removeFeature(line_feature);
                                                                }
// Adding a vector layer to the map
                                var start_geom = new
 ol.geom.Point(ol.proj.fromLonLat([parseFloat(start.split(',').shift()),parseFloat(start.split(',').pop())]));
                                var end_geom = new
 ol.geom.Point((ol.proj.fromLonLat([parseFloat(finish.split(',').shift()),parseFloat(finish.split(',').pop())])));
                                var way_geom = new
 ol.geom.Point((ol.proj.fromLonLat([parseFloat(stop.split(',').shift()),parseFloat(stop.split(',').pop())])));
                                start_feature = new ol.Feature({});
                                start_feature.setGeometry(start_geom);
                                origin_pointlayer = new GCUI.Layer.StandardVector({
                                        source: new ol.source.Vector({
                                                features: [start_feature]
                                                })
                                                });
                                end_feature = new ol.Feature({});
                                end_feature.setGeometry(end_geom);
                                destn_pointlayer = new GCUI.Layer.StandardVector({
                                        source: new ol.source.Vector({
                                                features: [end_feature]
                                                })
                                                });
                                stop_feature = new ol.Feature({});
                                stop_feature.setGeometry(way_geom);
                                stop_pointlayer = new GCUI.Layer.StandardVector({
                                        source: new ol.source.Vector({
                                                features: [stop_feature]
                                                })
                                                });
                                map.addLayer(origin_pointlayer);
                                map.addLayer(stop_pointlayer);
                                map.addLayer(destn_pointlayer);
                                line_feature = ol.format.WKT.prototype.readFeature(parsedjson.wktGeometry,{
                                dataProjection: 'EPSG:4326',
                                featureProjection: 'EPSG:3857'
                                });
                                vector_layer = new GCUI.Layer.StandardVector({
                                        source: new ol.source.Vector({
                                                features: [line_feature]
                                                })
                                                });
                                map.addLayer(vector_layer);
//Adding a style to the vector layer
                                var start_style = new ol.style.Style({
                                  image: new ol.style.Icon({
                                  crossOrigin: 'anonymous',
                                  src: 'https://i.ibb.co/VjGzprJ/Start.png'
                                        })
                                });
                                var end_style = new ol.style.Style({
                                  image: new ol.style.Icon(/** @type {module:ol/style/Icon~Options} */ ({
                                  crossOrigin: 'anonymous',
                                  src: 'https://i.ibb.co/cCJjRvj/Finish.png'
                                        }))
                                });
                                var stop_style = new ol.style.Style({
                                  image: new ol.style.Icon(/** @type {module:ol/style/Icon~Options} */ ({
                                  crossOrigin: 'anonymous',
                                  src: 'https://i.ibb.co/Qm1tbNy/Stop.png'
                                        }))
                                });
                                var fill = new ol.style.Fill({
```

```
                                color: [0, 70, 135, 0.9]
                              });
                              var stroke = new ol.style.Stroke({
                                color: [0, 70, 135, 1],
                                width: 5
                              });
                              var style = new ol.style.Style({
                                image: new ol.style.Circle({
                                      fill: fill,
                                      stroke: stroke,
                                      radius: 5
                                }),
                                fill: fill,
                                stroke: stroke
                              });
                              origin_pointlayer.setStyle(start_style);
                              destn_pointlayer.setStyle(end_style);
                              stop_pointlayer.setStyle(stop_style);
                              vector_layer.setStyle(style);
                              extent = line_feature.getGeometry().getExtent();
                              map.getView().fit(extent);
                }
                }
}
}
```

Event

Map loading event

The GCUI.Map constructor with **`server`** and **`layer`** options arguments gets asynchronously map informations (extent, resolutions, …) from the Geoptimization server. The map events system trigger a *load* event to notify that the map is loaded :

HTML

```
<div id="map1" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map1', options);
map.onEvent("load", onMapLoaded);

function onMapLoaded() {
    var map = GCUI.getMap('map1');
    alert("Map is loaded. Map extent is : " + map.getExtent());
}
```

The following code registers a callback function that will be called when map is ready :

```
map.onEvent("load",onMapLoaded);
```

In this example, the callback function display the map extent :

```
function onMapLoaded() {
    var map = GCUI.getMap('map1');
    alert("Map is loaded. Map extent is : "+ map.getExtent());
}
```

Map moveend event

This example shows how to listen to the map `moveend` event :

HTML

```
<div id="map2" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map2', options);
map.on('moveend', onMoveEnd);
function onMoveEnd(evt) {
        alert("End move, map extent is : "+ map.getExtent()+ " and zoom level is " + map.getZoom());
      }
```

Map click event

You can add a control to the map to register and define an action on the click event :

HTML

```
<div id="map4" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map4', options);
map.on('singleclick', onClick);
function onClick(evt) {
                var coordinate = evt.coordinate;
          alert("You clicked near x: " + coordinate[0] + " , y: " + coordinate[1]);
      }
```

Feature select event

You can add a control to the map to register and define an action on the feature selected event :

HTML

```
<link rel="stylesheet" href="https://openlayers.org/en/v5.3.0/css/ol.css" type="text/css" />
<style>
.anchorright#ol-popup {
    right: -30px;
```

```
}

.anchorright#ol-popup:after {
    right: 20px;
}

.anchorright#ol-popup:before {
    right: 20px;
}

.anchorbottom#ol-popup {
    bottom: 11px;
}

.anchorbottom#ol-popup:after, .anchorbottom#ol-popup:before {
    top: 100%;
}

#ol-popup-content.content {
    max-height: 25em;
    overflow-y: auto;
}

.#ol-popup{
    padding-top: 2em;
    padding-right: 0.35em;
}

.anchorbottom#ol-popup:after {
    border-top-color: white;
}

.anchorbottom#ol-popup:before {
    border-top-color: #cccccc;
}

.anchorleft#ol-popup {
    left: -25px;
}

.anchorleft#ol-popup:after {
    left: 25px;
}

.anchorleft#ol-popup:before {
    left: 25px;
}

.anchortop#ol-popup {
    top: 11px;
}

.anchortop#ol-popup:after, .anchortop#ol-popup:before {
    bottom: 100%;
}

.anchortop#ol-popup:after {
    border-bottom-color: white;
}

.anchortop#ol-popup:before {
    border-bottom-color: #cccccc;
}
```

```
#ol-popup {
    position: absolute;
    background-color: white;
    -webkit-filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));
    padding: 15px;
    border-radius: 5px;
    border: 1px solid #cccccc;
    min-width: 280px;
}

#ol-popup:after, #ol-popup:before {
    border: solid transparent;
    content: " ";
    height: 0;
    width: 0;
    position: absolute;
    pointer-events: none;
}

#ol-popup:after {
    border-width: 10px;
    margin-left: -10px;
}

#ol-popup:before {
    border-width: 11px;
    margin-left: -11px;
}

#ol-popup-closer {
    text-decoration: none;
    position: absolute;
    top: 2px;
    right: 8px;
    font-family: Icons;
}

#ol-popup-closer:hover {
    cursor: pointer;
    color: #004687;
}

#ol-popup-closer:after {
  content: "#";
}

#ol-popup .pages {
        text-align: right;
    position: relative;
    right: -1em;
    bottom: -0.8em;
}

#ol-popup .pages i {
    cursor: pointer;
}

#ol-popup .pages > * {
    display: inline-block;
}
```

```
#ol-popup .pages i.disabled {
    cursor: default;
}

.gcuiAnchoredContentData {
    height: 100%;
    width: 100%;
    overflow-y: auto;
    overflow-x: hidden;
}

.gcuiAnchoredContentData .table-width {
    width: 50%;
}

.gcuiAnchoredContentData .ui.table  .right.aligned .content {
    text-align: right;
    margin-right: .2em;
    color: #808080;
}

.gcuiAnchoredContentData .ui.table .content {
    color: #000000;
    border: none !important;
}

.gcuiAnchoredContentData .ui.table td {
    padding: .25em;
    border: none !important;
}
.gcuiAnchoredContentData .ui.table, th, td {
         border: none !important;
}
.ol-custom-overviewmap,
.ol-custom-overviewmap.ol-uncollapsible {
        bottom: auto;
        left: auto;
        right: 0;
        top: 0;
}
.ol-custom-overviewmap:not(.ol-collapsed)  {
        border: 1px solid black;
}
.ol-custom-overviewmap .ol-overviewmap-map {
        border: none;
        width: 300px;
}
.ol-custom-overviewmap .ol-overviewmap-box {
        border: 2px solid red;
}
.ol-custom-overviewmap:not(.ol-collapsed) button{
        bottom: auto;
        left: auto;
        right: 1px;
        top: 1px;
}
.ol-rotate {
        top: 170px;
        right: 0;
}
</style>
<div id="map5" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map5', options);
map.onEvent("load", onMapLoaded);

function onMapLoaded() {
        var vectorSource = new ol.source.Vector({});
        var point_feature_1 = new ol.Feature({
        geometry: new ol.geom.Point([255490, 6249610]),
        Name: "My Data 1"
        });
        var point_feature_2 = new ol.Feature({
        geometry: new ol.geom.Point([264433, 6252668]),
        Name: "My Data 2"
        });
        vectorSource.addFeature(point_feature_1);
        vectorSource.addFeature(point_feature_2);
        vector_layer = new GCUI.Layer.StandardVector({
        source: vectorSource
                });
        map.addLayer(vector_layer);
//Adding a style to the vector layer
        var fill = new ol.style.Fill({
          color: [0, 70, 135, 0.9]
        });
        var stroke = new ol.style.Stroke({
          color: [255, 255, 255, 1],
          width: 2
        });
        var style = new ol.style.Style({
          image: new ol.style.Circle({
                fill: fill,
                stroke: stroke,
                radius: 10
          }),
          fill: fill,
          stroke: stroke
        });
        vector_layer.setStyle(style);
        extent = vectorSource.getExtent();
        map.getView().fit(extent);
        //select interaction working on "click"
        var selectClick = new ol.interaction.Select({
          condition: ol.events.condition.click
        });
         if (selectClick !== null) {
                map.addInteraction(selectClick);
                selectClick.on('select', function(e) {
                  var feature = e.target.getFeatures().getArray();
                  if (feature.length >= 1) {
                  var opt_options = "<div class='gcuiAnchoredContentData'><table class='ui very basic table
 unstackable' cellspacing='0' cellpadding='0'><tbody><tr><td class='right aligned'><div class='content'>X</
div></td><td><div class='content'>" + feature[0].values_.geometry.flatCoordinates[0] + "</div></
td></tr><tr><td class='right aligned'><div class='content'>Y</div></td><td><div class='content'>" +
 feature[0].values_.geometry.flatCoordinates[1] + "</div></td></tr><tr><td class='right aligned'><div
 class='content'>Name</div></td><td><div class='content'>" + feature[0].values_.Name + "</div></td></tr></
tbody></table></div>";
                  var popup = new GCUI.Control.Popup(map, feature[0].values_.geometry.flatCoordinates,
 {content : opt_options});
```

```
                  map.addOverlay(popup);
                  popup.initialize(feature[0].values_.geometry.flatCoordinates);
                  }
                });
          }
}
```

Feature modify event

You can add a control to the map to register and define an action on the feature modified event :

HTML

```
<div id="map6" style="height:300px"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map6', options);
map.onEvent("load", onMapLoaded);

function onMapLoaded() {
        var vectorSource = new ol.source.Vector({});
        var point_feature_1 = new ol.Feature({
        geometry: new ol.geom.Point([255490, 6249610]),
        Name: "My Data 1"
        });
        var point_feature_2 = new ol.Feature({
        geometry: new ol.geom.Point([264433, 6252668]),
        Name: "My Data 2"
        });
        vectorSource.addFeature(point_feature_1);
        vectorSource.addFeature(point_feature_2);
        vector_layer = new GCUI.Layer.StandardVector({
        source: vectorSource
                });
        map.addLayer(vector_layer);
//Adding a style to the vector layer
        var fill = new ol.style.Fill({
          color: [0, 70, 135, 0.9]
        });
        var stroke = new ol.style.Stroke({
          color: [255, 255, 255, 1],
          width: 2
        });
        var style = new ol.style.Style({
          image: new ol.style.Circle({
                fill: fill,
                stroke: stroke,
                radius: 10
          }),
          fill: fill,
          stroke: stroke
        });
        vector_layer.setStyle(style);
        extent = vectorSource.getExtent();
        map.getView().fit(extent);
```

```
        var modify = new ol.interaction.Modify({source: vectorSource});
        map.addInteraction(modify);
}
```

Projection

Center the map on WGS84 coordinates

HTML

```
<div style="position: absolute; z-index: 5000; left: 65px;">
        <input type="text" id="Long" placeholder="Longitude" value="2.328450697115128">
        <input type="text" id="Lat" placeholder="Latitude" value="48.80814035316188">
        <button type="button" onclick="GCUI.examples.moveTo()">Move To</button>
</div>
<div id="map" style="height: 300px;"></div>
```

JavaScript

```
var options = {
    server : 'https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts',
    layer : 'STANDARD'
};
var map = new GCUI.Map('map', options);

GCUI.examples = {
        moveTo : function() {
    var map = GCUI.getMap('map');
    var coords =
 [parseFloat(document.getElementById("Long").value),parseFloat(document.getElementById("Lat").value)];
    var xy = ol.proj.transform(coords, 'EPSG:4326','EPSG:3857');
    map.moveTo(xy, 11);
}
}
```

Geoptimization REST API

Geoptimization web services are accessible in REST or standard protocol, they provide map display, geocoding, address standardization, route and matrix calculation, search around, search along, route sequencing and much more besides.

The quality of our web services reflect our long experience of handling geographic data, and offer the powerful functionality that comes with highly evolved engines tested over time.

What are the web services availables?

Please find the list below:

Mapping a Weather

- Map display (WMTS)
- Coordinates Transformation
- Weather

Search/Geocoding

- Autocompletion
- Autosuggest
- Geocoding / Batch geocoding
- Normalization
- Reverse geocoding
- Find an object

Territory management

- Territory management

Routing/Optimizing

- Route calculation / Batch route calculation
- ETA (Estimated Time of Arrival)
- Waypoint sequence
- Isochrone / Multi-isochrone
- Matrix calculation
- Search Around
- Search Along
- Pickup and Delivery
- Optimization

WMTS

Basic principles

This web service Web Map Tiles Service (WMTS) is respecting the standards of the Open Geospatial Consortium (OGC), to publish cartographic data in the form of predefined tiles.

Geoptimization-api is using the same projection (Spherical Mercator EPSG:3857) and tiling profile than the mains others maps providers to ensure the same world coverage and cross sharing compatibility.

Parameters / properties

KVP and REST protocols are supported for WMTS by the application. WMTS 1.0.0 OGC is also supported (http://www.opengeospatial.org/standards/wmts).

> While retrieving tiles in itself will not require authentication, you will need to call the LayerInfo service on the layer regularly to check whether the *Version*, *tab* (used for STYLE) *tileWidth* and *tileHeight* parameters need updating at the time of the call. The *layerName* parameter itself remains identical at all times.

GetCapabilities

- KVP encoding: https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts?request=GetCapabilities&version=1.0.0&service=WMTS
- REST encoding: https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts/1.0.0/WmtsCapabilities.xml

GetTile

- KVP encoding example: https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts?LAYER=STANDARD&layerVersion=1&SERVICE=WMTS&REQUEST=GetTile&VERSION=1.0.0&STYLE=STANDAR%3A3857&TILEMATRIX=11&TILEROW=707&TILECOL=1036&FORMAT=image%2Fpng
- REST encoding example: https://api.geoconcept.com/EU/GCW/geoconcept-web/wmts/STANDARD/STANDARD/epsg:3857/11/707/1036.png

FAQ

1. What is the WMTS coverage available?

   Most of the world is available.

Layer informations

Basic principles

This web service provides metadatas of the mapping layers available. These informations are required in order to use Web Map Tiles Service (WMTS).

Layerinfo example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/layer/json",
        "type":"GET",
        "params": {
                "name": {
                        "sample":"STANDARD"
```

```
                }
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
map	Map file name	yes *	
tab	Tab name	yes *	
name	Layer name	yes *	

(*) At least one of the three parameters map, tab and name must be assigned a value.

Output

parameter	type	min/max	description
ratios	string	0/1	Ratios for each map scale (12 or 24 scales)
extent	string	0/1	Map bounding box
precision	string	0/1	Map precision (ie: 0.01)
projection	string	0/1	Map projection (EPSG code such as EPSG:3857)
tab	string	0/1	Tab name
name	string	0/1	Layer name
version	string	0/1	Layer version
format	string	0/1	Picture format (png, jpg)
map	string	0/1	Map file name
tileWidth	string	0/1	Tile width
tileHeight	string	0/1	Tile height
metadataUrl	string	0/1	Metadata url
legendUrl	string	0/1	Legend url

Possible returns

Case of an layerInfo that has been found (status is OK)


```
{
   "message":"Layer informations",
   "status":"OK",
   "result":{
      "extent":{
         "minX":-2003750660,
         "minY":-2003750660,
         "maxX":2003750660,
         "maxY":2003750660
      },
      "precision":0.01,
      "projection":"EPSG:3857",
      "tab":"STANDARD",
      "name":"STANDARD",
```

```
      "version":"210225001",
      "format":"png",
      "map":"EURO/HE-ENT-M20-EURO_cua",
      "tileWidth":256,
      "tileHeight":256,
      "isWebmap":false,
      "ratios":[
         0.13396648497664457,
         -0.13396648497664457,
         0.06698324248832228,
         -0.06698324248832228,
         0.03349162124416114,
         -0.03349162124416114,
         0.01674581062208057,
         -0.01674581062208057,
         0.008372905311040285,
         -0.008372905311040285,
         0.004186452655520143,
         -0.004186452655520143,
         0.0020932263277600714,
         -0.0020932263277600714,
         0.0010466131638800357,
         -0.0010466131638800357,
         0.0005233065819400178,
         -0.0005233065819400178,
         0.0002616532909700089,
         -0.0002616532909700089,
         0.00013082664548500446,
         -0.00013082664548500446,
         0.00006541332274250223,
         -0.00006541332274250223,
         0.00003270661371251115,
         -0.00003270661371251115,
         0.00001635330685625558,
         -0.00001635330685625558,
         0.00000817666534281279,
         -0.00000817666534281279,
         0.00000408833267140639,
         -0.00000408833267140639,
         0.00000204416633570319,
         -0.00000204416633570319,
         0.00000102208316785159,
         -0.00000102208316785159,
         5.110415839257987e-7,
         -5.110415839257987e-7,
         2.5552079196289934e-7,
         -2.5552079196289934e-7,
         1.2776039598144967e-7,
         -1.2776039598144967e-7,
         6.388019799072483e-8,
         -6.388019799072483e-8
      ]
   }
}
```


Case of a wrong or empty parameter (status is ERROR)


```
{
   "message":"Layer  doesn't exist",
   "status":"ERROR",
   "result":{
      "extent":null,
      "precision":0.0,
```

```
      "projection":null,
      "tab":null,
      "name":null,
      "version":null,
      "format":null,
      "map":null,
      "tileWidth":0,
      "tileHeight":0,
      "metadataUrl":null,
      "legendUrl":null
   }
}
```

FAQ

1. What are the *layerName* availables?
   - STANDARD Global basemap - Plain - World
   - ORDER1 The highest administrative level in which a country can be subdivided - Transparency - World (availability may vary following country)
   - ORDER2 An intermediate administrative level of a country and is a sub-division of an Order-1 Area - Transparency - World (availability may vary following country)
   - ORDER8 The lowest level of the country's administrative hierarchy that is present country-wide - Transparency - World (availability may vary following country)
   - POSTAL_BOUNDARIES Postal areas - Transparency - World (availability may vary following country)
   - MAINROADS Main road network - Transparency - Europe, North America
   - CHARGING_STATIONS Charging stations for electric vehicles - Transparency - France

Coordinates transformation

Basic principles

This web service transforms pairs of coordinates from one projection to another. For example, from wgs84 to Sphere Mercator.

Coordinates transformation example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/srsTransform.json",
        "method":"POST",
        "body": {
        "inSrs": "epsg:4326",
        "outSrs": "epsg:3857",
        "points": [
        {
          "x": "31.137736",
          "y": "29.975256"
        },
        {
          "x": "103.866986",
          "y": "13.412490"
        },
```

```
        {
          "x": "-72.54583",
          "y": "-13.1639"
        }
      ]
    }
}
```

Parameters / properties

Input

parameter	description	optional	default
inSrs	input projection (EPSG code such as epsg:4326 or wgs84)	no	
outSrs	output projection (EPSG code such as epsg:4326 or wgs84)	no	
points	Coordinates for points separated by “,”. Pair of XY coordinates are separated by “;”.	no	

Output

Transformed points (srsTransformResult)

parameter	type	min/max	description
points	array geographicPoint	0/ unlimited	transformed points

Points(geographicPoint)

parameter	type	min/max	description
x	double	1/1	first coordinate or longitude
y	double	1/1	second coordonnée or latitude

Possible returns

Case of an itinerary found (status is OK)

```
{
  "message": null,
  "status": "OK",
  "points": [
    {
      "x": 3466236.915975383,
      "y": 3500369.6293757656
    },
    {
      "x": 11562419.991752075,
      "y": 1506897.9005721502
    },
    {
      "x": -8075764.854775389,
      "y": -1478463.6733575095
    }
  ]
}
```

Case of an empty start projection

```
An error append : (500) Internal Server Error
InvalidParameterException: inSrs can't be null or empty
```

Case of an empty destination projection

```
An error append : (500) Internal Server Error
InvalidParameterException: outSrs can't be null or empty
```

Case of an incorrect projection

```
An error append : (500) Internal Server Error
CoordinateTransformEngineException: inSrs or outSrs do not exist
```

Case of a coordinate for an empty point

```
An error append : (500) Internal Server Error
InvalidParameterException: points can't be null or empty
```

Case of a faulty typing entry on the coordinates for a series of points

```
An error append : (500) Internal Server Error
NumberFormatException: For input string: "48.8degree"
```

Weather

Basic principles

The Weather API is a RESTful API that provides for a specific location:

- weather forecasts and reports on current weather conditions
- information on severe weather alerts
- information about when the sun and moon rise and set, and the phase of the moon

Weather observation example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/weather.json",
        "type":"GET",
        "params": {
                "product": {
                        "sample":"observation"},
                "latitude": {
                        "sample":"47.26682"},
                "longitude": {
                        "sample":"-2.33794"}
        }
}
```

Parameters / Properties

Input

Input documentation [https://developer.here.com/documentation/destination-weather/dev_guide/topics/resource-report.html], can be used too (without the credentials).

parameter	description	optional	default
product	A parameter identifying the type of report to obtain. The possible values are as follows: - observation – current weather conditions from the eight closest locations to the specified location - forecast_7days – morning, afternoon, evening and night weather forecasts for the next seven days. - forecast_7days_simple – daily weather forecasts for the next seven days - forecast_hourly – hourly weather forecasts for the next seven days - forecast_astronomy – information on when the sun and moon rise and set, and on the phase of the moon for the next seven days - alerts – forecasted weather alerts for the next 24 hours - nws_alerts – all active watches and warnings for the US and Canada	no	
latitude and longitude	Geographical coordinates in WGS-84-compliant format, specifying the area covered by the weather report. For example, *latitude=41.83&longitude=-87.68* requests a report for these coordinates. The response is for the closest reporting stations to these coordinates.	yes *	
name	Name of a city. If there is more than one match for the name, then the most populous location is in the response. Country, state and street name can be added to this parameter. For example, the response for *name=Berlin, USA* is a report for the city of Berlin, New Hampshire, the response for *name=Berlin, IL* is a report for the city of Berlin, Illinois, and the response for *name=Berlin* is a report for the city of Berlin, Germany.	yes *	
zipcode	ZIP code of the location. This parameter is supported only for locations in the United States of America.	yes *	
hourlydate	Date for which hourly forecasts are to be retrieved. The format is YYYY-MM-DD or YYYY-MM-DDThh:mm:ss Available only when the product parameter is set to forecast_hourly If you set hourlydate for a date and time in the past or more than seven days in the future, the response does not contain any information. If you set hourlydate to the current date, the response contains hourly forecasts for the current date, including some forecasts before when you sent the query depending on the data in the system. If you set hourlydate to a future date, the response contains hourly forecasts for the entire day.	no	
oneobservation	Boolean, if set to true, the response only includes the closest location. Only available when the product parameter is set to observation.	no	false
language	Defines the language used in the descriptions in the response. For example, the response for a query with language=french has descriptions in French. The following languages are supported (danish, dutch, english, french, german, greek, hindi, italian, japanese, polish, portuguese, romanian, russian, spanish). If the language specified is not supported, the response contains descriptions in English.	no	english
metric	Boolean, defines whether metric or imperial units are used in the response. If set to false, the response contains imperial units (feet, inch, Fahrenheit, miles).	no	true

(*) At least one of these latitude and longitude OR name OR zipcode must be assigned a value.

Output

See Output documentation [https://developer.here.com/documentation/destination-weather/dev_guide/topics/resource-response-type-report.html].

Possible returns

Case of a correct answer (status is 200)


```
{
    "observations": {
        "location": [
            {
                "observation": [
                    {
                        "daylight": "D",
                        "description": "Partly sunny. Mild.",
                        "skyInfo": "14",
                        "skyDescription": "Partly sunny",
                        "temperature": "19.00",
                        "temperatureDesc": "Mild",
                        "comfort": "19.00",
                        "highTemperature": "18.10",
                        "lowTemperature": "15.60",
                        "humidity": "64",
                        "dewPoint": "12.00",
                        "precipitation1H": "*",
                        "precipitation3H": "*",
                        "precipitation6H": "*",
                        "precipitation12H": "*",
                        "precipitation24H": "*",
                        "precipitationDesc": "",
                        "airInfo": "*",
                        "airDescription": "",
                        "windSpeed": "16.68",
                        "windDirection": "230",
                        "windDesc": "Southwest",
                        "windDescShort": "SW",
                        "barometerPressure": "1011.18",
                        "barometerTrend": "",
                        "visibility": "*",
                        "snowCover": "*",
                        "icon": "6",
                        "iconName": "mostly_cloudy",
                        "iconLink": "http://app-test.geoconcept.com/zEU/GCW/geoconcept-web/api/lbs/weather/
icon/17.png",
                        "ageMinutes": "27",
                        "activeAlerts": "0",
                        "country": "France",
                        "state": "Pays de la Loire",
                        "city": "Pornichet",
                        "latitude": 47.2668,
                        "longitude": -2.3379,
                        "distance": 56.43,
                        "elevation": 0,
                        "utcTime": "2021-07-06T14:30:00.000+02:00"
                    }
                ],
                "country": "France",
```

```
                "state": "Pays de la Loire",
                "city": "Pornichet",
                "latitude": 47.26682,
                "longitude": -2.33794,
                "distance": 0,
                "timezone": 1
            },
            {
                "observation": [
                    {
                        "daylight": "D",
                        "description": "Overcast. Mild.",
                        "skyInfo": "18",
                        "skyDescription": "Overcast",
                        "temperature": "19.00",
                        "temperatureDesc": "Mild",
                        "comfort": "19.00",
                        "highTemperature": "*",
                        "lowTemperature": "*",
                        "humidity": "63",
                        "dewPoint": "11.78",
                        "precipitation1H": "*",
                        "precipitation3H": "0.03",
                        "precipitation6H": "0.03",
                        "precipitation12H": "*",
                        "precipitation24H": "*",
                        "precipitationDesc": "",
                        "airInfo": "*",
                        "airDescription": "",
                        "windSpeed": "25.95",
                        "windDirection": "220",
                        "windDesc": "Southwest",
                        "windDescShort": "SW",
                        "barometerPressure": "1011.10",
                        "barometerTrend": "Rising",
                        "visibility": "19.15",
                        "snowCover": "0.00",
                        "icon": "7",
                        "iconName": "cloudy",
                        "iconLink": "http://app-test.geoconcept.com/zEU/GCW/geoconcept-web/api/lbs/weather/
icon/17.png",
                        "ageMinutes": "57",
                        "activeAlerts": "0",
                        "country": "France",
                        "state": "Pays de la Loire",
                        "city": "Nantes",
                        "latitude": 47.1667,
                        "longitude": -1.6,
                        "distance": 56.9,
                        "elevation": 23,
                        "utcTime": "2021-07-06T14:00:00.000+02:00"
                    }
                ],
                "country": "France",
                "state": "Pays de la Loire",
                "city": "Nantes",
                "latitude": 47.1667,
                "longitude": -1.6,
                "distance": 56.9,
                "timezone": 1
            },
            {
                "observation": [
```

```
                    {
                        "daylight": "D",
                        "description": "Partly sunny. Mild.",
                        "skyInfo": "14",
                        "skyDescription": "Partly sunny",
                        "temperature": "19.50",
                        "temperatureDesc": "Mild",
                        "comfort": "19.50",
                        "highTemperature": "*",
                        "lowTemperature": "*",
                        "humidity": "72",
                        "dewPoint": "14.28",
                        "precipitation1H": "*",
                        "precipitation3H": "0.00",
                        "precipitation6H": "0.00",
                        "precipitation12H": "*",
                        "precipitation24H": "*",
                        "precipitationDesc": "",
                        "airInfo": "*",
                        "airDescription": "",
                        "windSpeed": "22.24",
                        "windDirection": "220",
                        "windDesc": "Southwest",
                        "windDescShort": "SW",
                        "barometerPressure": "1011.70",
                        "barometerTrend": "Rising",
                        "visibility": "15.13",
                        "snowCover": "*",
                        "icon": "6",
                        "iconName": "mostly_cloudy",
                        "iconLink": "http://app-test.geoconcept.com/zEU/GCW/geoconcept-web/api/lbs/weather/
icon/17.png",
                        "ageMinutes": "57",
                        "activeAlerts": "0",
                        "country": "France",
                        "state": "Pays de la Loire",
                        "city": "St-Sauveur",
                        "latitude": 46.7,
                        "longitude": -2.3333,
                        "distance": 63.1,
                        "elevation": 24,
                        "utcTime": "2021-07-06T14:00:00.000+02:00"
                    }
                ],
                "country": "France",
                "state": "Pays de la Loire",
                "city": "St-Sauveur",
                "latitude": 46.7,
                "longitude": -2.3333,
                "distance": 63.1,
                "timezone": 1
            },
            {
                "observation": [
                    {
                        "daylight": "D",
                        "description": "Partly sunny. Mild.",
                        "skyInfo": "14",
                        "skyDescription": "Partly sunny",
                        "temperature": "16.89",
                        "temperatureDesc": "Mild",
                        "comfort": "16.89",
                        "highTemperature": "*",
```

```
                        "lowTemperature": "*",
                        "humidity": "77",
                        "dewPoint": "12.78",
                        "precipitation1H": "*",
                        "precipitation3H": "0.00",
                        "precipitation6H": "0.00",
                        "precipitation12H": "*",
                        "precipitation24H": "*",
                        "precipitationDesc": "",
                        "airInfo": "*",
                        "airDescription": "",
                        "windSpeed": "35.21",
                        "windDirection": "210",
                        "windDesc": "Southwest",
                        "windDescShort": "SW",
                        "barometerPressure": "1010.20",
                        "barometerTrend": "Rising",
                        "visibility": "15.13",
                        "snowCover": "*",
                        "icon": "6",
                        "iconName": "mostly_cloudy",
                        "iconLink": "http://app-test.geoconcept.com/zEU/GCW/geoconcept-web/api/lbs/weather/
icon/17.png",
                        "ageMinutes": "57",
                        "activeAlerts": "0",
                        "country": "France",
                        "state": "Brittany",
                        "city": "Le Talut",
                        "latitude": 47.3,
                        "longitude": -3.2167,
                        "distance": 66.46,
                        "elevation": 34,
                        "utcTime": "2021-07-06T14:00:00.000+02:00"
                    }
                ],
                "country": "France",
                "state": "Brittany",
                "city": "Le Talut",
                "latitude": 47.3,
                "longitude": -3.2167,
                "distance": 66.46,
                "timezone": 1
            },
            {
                "observation": [
                    {
                        "daylight": "D",
                        "description": "Overcast. Mild.",
                        "skyInfo": "18",
                        "skyDescription": "Overcast",
                        "temperature": "18.72",
                        "temperatureDesc": "Mild",
                        "comfort": "18.72",
                        "highTemperature": "*",
                        "lowTemperature": "*",
                        "humidity": "66",
                        "dewPoint": "12.22",
                        "precipitation1H": "*",
                        "precipitation3H": "0.00",
                        "precipitation6H": "0.00",
                        "precipitation12H": "*",
                        "precipitation24H": "*",
                        "precipitationDesc": "",
```

```
                        "airInfo": "*",
                        "airDescription": "",
                        "windSpeed": "20.39",
                        "windDirection": "230",
                        "windDesc": "Southwest",
                        "windDescShort": "SW",
                        "barometerPressure": "1011.90",
                        "barometerTrend": "Rising",
                        "visibility": "18.02",
                        "snowCover": "*",
                        "icon": "7",
                        "iconName": "cloudy",
                        "iconLink": "http://app-test.geoconcept.com/zEU/GCW/geoconcept-web/api/lbs/weather/
icon/17.png",
                        "ageMinutes": "57",
                        "activeAlerts": "0",
                        "country": "France",
                        "state": "Pays de la Loire",
                        "city": "La Roche-Sur-Yon",
                        "latitude": 46.7,
                        "longitude": -1.3833,
                        "distance": 96.11,
                        "elevation": 85,
                        "utcTime": "2021-07-06T14:00:00.000+02:00"
                    }
                ],
                "country": "France",
                "state": "Pays de la Loire",
                "city": "La Roche-Sur-Yon",
                "latitude": 46.7,
                "longitude": -1.3833,
                "distance": 96.11,
                "timezone": 1
            },
            {
                "observation": [
                    {
                        "daylight": "D",
                        "description": "Partly sunny. Mild.",
                        "skyInfo": "14",
                        "skyDescription": "Partly sunny",
                        "temperature": "17.00",
                        "temperatureDesc": "Mild",
                        "comfort": "17.00",
                        "highTemperature": "*",
                        "lowTemperature": "*",
                        "humidity": "77",
                        "dewPoint": "13.00",
                        "precipitation1H": "*",
                        "precipitation3H": "*",
                        "precipitation6H": "*",
                        "precipitation12H": "*",
                        "precipitation24H": "*",
                        "precipitationDesc": "",
                        "airInfo": "*",
                        "airDescription": "",
                        "windSpeed": "20.39",
                        "windDirection": "240",
                        "windDesc": "Southwest",
                        "windDescShort": "SW",
                        "barometerPressure": "1012.19",
                        "barometerTrend": "",
                        "visibility": "*",
```

```
                        "snowCover": "*",
                        "icon": "6",
                        "iconName": "mostly_cloudy",
                        "iconLink": "http://app-test.geoconcept.com/zEU/GCW/geoconcept-web/api/lbs/weather/
icon/17.png",
                        "ageMinutes": "27",
                        "activeAlerts": "0",
                        "country": "France",
                        "state": "Pays de la Loire",
                        "city": "La Roche",
                        "latitude": 46.7,
                        "longitude": -1.38,
                        "distance": 96.3,
                        "elevation": 87,
                        "utcTime": "2021-07-06T14:30:00.000+02:00"
                    }
                ],
                "country": "France",
                "state": "Pays de la Loire",
                "city": "La Roche",
                "latitude": 46.7,
                "longitude": -1.38,
                "distance": 96.3,
                "timezone": 1
            }
        ]
    },
    "feedCreation": "2021-07-06T12:57:23.235Z",
    "metric": true
}
```


Case of a missing parameter (status is 400)


```
{
    "Type": "Invalid Request",
    "Message": [
        "Mandatory parameter latitude or longitude is missing"
    ]
}
```


Autocompletion

Basic principles

This automatic completion service, or autocomplete, saves internauts time when entering location or address. The function suggests likely locations as the letters of an address are typed in, or even as they are deleted, in the localisation input field.

example: the internaut enters "avenue des cha», and the service returns «Avenue des Champs-Élysées, Paris», as well as other addresses starting with the same string.

The location names are ranked or graded according to a weighting that allows the most probable solutions to be presented first, taking into account the size of the town.

This web service is not part of the geocoding operation. The geocoding web service can be used with the data received from the autocomplete service, taking the address found by the autocomplete module as an entry parameter of the geocoder.

To ensure optimised performances, we advise building the geocoding from the result of the autocomplete operation with separate fields, and not basing the geocoding on fulltext fields (both of these are provided in the autocomplete result).

See too Autosuggest web service.

Autocompletion example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/autocomplete/v3.json",
        "type":"GET",
        "params": {
                "text": {
                        "sample":"Avenue des cha"
                },
                "maximumResponses": {
                        "sample":"3"
                },
                "countryCode": {
                        "sample":"FR"
                },
                "matchLevels": {
                        "sample":"fromLabel"
                }
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
text	Location address to enter	yes	
maximumResponses	maximum number of results for addresses in the response. if the value is left empty, the parameter defined by default will apply	yes	10
bounds	Sets a focus on a geographic area represented by the top-left and the bottom-right corners (so *longitude-min,latitude-min;longitude-max,latitude-max*) of a bounding box so the results within this area are more important than results outside of this area. ex: &bounds=2.3222,48.8111;2.4222,48.9111 Can be combined with the countryCode.	yes	
center	Sets a focus on a geographic point represented by a single geo-coordinate pair, comma separated, so the results nearby this point are more important than results far from this point. ex: &center=2.3222,48.8111 Can be combined with the countryCode.	yes	
countryCode	The countryCode parameter limits suggestions to a country (2-letter ISO country codes) Can be combined with the bound or center parameters.	yes	

parameter	description	optional	default
language	The preferred language of address elements in the result. The language parameter must be provided as 2-letter ISO language code.	yes	
matchLevels	Filter match precision, one of: houseNumber, intersection, street, postalCode, district, city, county, state, country or fromLabel. To use geocoding web service with the data received from the autocomplete service it is recommended to use the parameter matchLevels=fromLabel.	yes	

Output

property	type	min/max	description
fulltext	string	0/1	Location found with a specific nomenclature including the post code and the name of the town.
language	string	0/1	Location result language (2-letter ISO language code).
countryCode	string	0/1	Country code of the location (2-letter ISO country code).
matchLevel	string	0/1	Match precision. One of: houseNumber, intersection, street, postalCode, district, city, county, state, country.
distance	string	0/1	Distance in meters from the search center. Only available for queries with bound or center parameter.
country	string	0/1	Country of the address found
state	string	0/1	State of the address found
county	string	0/1	County of the address found
city	string	0/1	City of the address found
postCode	string	0/1	Postal code of the address found
district	string	0/1	District of the address found
street	string	0/1	Street of the address found
streetNumber	string	0/1	Street number of the address found

Possible returns

The case of addresses that are found (status est OK)


```
{
   "message":null,
   "status":"OK",
   "results":[
      {
         "fulltext":"Avenue des Champs-Élysées, 75008 Paris",
         "language":"fr",
         "countryCode":"FR",
         "matchLevel":"street",
         "distance":0.0,
         "country":"France",
         "state":"Île-de-France",
         "county":"Paris",
         "city":"Paris",
         "postCode":"75008",
         "district":"8e Arrondissement",
         "street":"Avenue des Champs-Élysées",
         "streetNumber":null
      },
      {
```

```
            "fulltext":"Avenue des Champs Lasniers, 91940 Les Ulis",
            "language":"fr",
            "countryCode":"FR",
            "matchLevel":"street",
            "distance":0.0,
            "country":"France",
            "state":"Île-de-France",
            "county":"Essonne",
            "city":"Les Ulis",
            "postCode":"91940",
            "district":null,
            "street":"Avenue des Champs Lasniers",
            "streetNumber":null
         },
         {
            "fulltext":"Avenue des Charmes, 94520 Périgny",
            "language":"fr",
            "countryCode":"FR",
            "matchLevel":"street",
            "distance":0.0,
            "country":"France",
            "state":"Île-de-France",
            "county":"Val-de-Marne",
            "city":"Périgny",
            "postCode":"94520",
            "district":null,
            "street":"Avenue des Charmes",
            "streetNumber":null
         }
      ]
   }
```


The case of an address that does not exist (status is OK)


```
{
   "message":null,
   "status":"OK",
   "results":[
   ]
}
```


Autosuggest

Basic principles

This automatic completion service, or autocomplete, saves internauts time when entering place names or addresses. The function suggests likely locations as the letters of an address are typed in, or even as they are deleted, in the localisation input field.

Unlike autocompletion service, it can return places (points of interest, restaurants, monuments, etc.) in addition to addresses.

example: the internaut enters "louvre", and the service returns either:

- Louvre museum "Musée du Louvre (Pyramide du Louvre)",
- Underground stations "Palais Royal Musée du Louvre" and "Louvre Rivoli",
- Town "Louvres, Île-de-France, France",

- Car park "Le Louvre".

Autosuggest example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/autosuggest.json",
        "type":"GET",
        "params": {
                "query": {
                        "sample":"Louvre"
                },
                "maxItems": {
                        "sample":"5"
                },
                "location": {
                        "sample":"2.30432,48.79859"
                },
                "language": {
                        "sample":"fr"
                }
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
query	A free-text query	no	
location	The center of the search area. A coordinates couple as always *easting,northing* (*X,Y* or *longitude,latitude*, depending on the coordinate system given in srs parameter). Since the default coordinate system is *longitude,latitude* WGS84 (EPSG:4326), the default for location is *longitude,latitude* WGS84.	yes *	
bounds	Search within a geographic area provided as a rectangle. A bounding box, provided as always *easting-min,northing-min,easting-max,northing-max* (*easting-min,northing-min;easting-max,northing-max* is tolerated). The coordinates depend on the coordinate system given in srs parameter. So, with the default coordinate system the format is *longitude-min,latitude-min,longitude-max,latitude-max*. Example : *-284651.51,5922035.22;-99368.12,6080412.70*, with srs=*EPSG:3857*. This is a hard filter. Items will be returned if they are located within the specified area.	yes *	
circle	Search within a geographic circular area, provided as center and radius. So always *easting,northing;radius* (*easting,northing;r=radius* is tolerated). Radius is always in meters. The coordinates depend on the coordinate system given in srs parameter. So, with the default coordinate system the format is *longitude,latitude;radius*. Example: *-260370.793,5984549.94;10000*. This is a hard filter. Items will be returned if they are located within the specified area.	yes *	
countryCode	A country (or multiple countries provided as comma-separated) alpha2 or alpha-3 country codes.	yes	

parameter	description	optional	default
srs	Coordinate system used for all coordinates (given in location parameter as input, or in AutosuggestResults's item location as output). The format is an EPSG coordinate system identifier. Defaults to longitude,latitude' WGS84 (EPSG:4326). Example : *EPSG:3857* for pseudo-Mercator.	yes	
language	The language to be used for result rendering as a BCP 47 compliant language code.	yes	
maxItems	Maximum number of items to be retrieved.	no	5
extended	When set to true, may return resultType items that are not strictly considered as an address (such as administrativeArea, or other).	no	false

(*) At least one of the three parameters location, bounds or circle must be assigned a value.

Output

property	type	min/max	description
items	array of AutosuggestItem	0/ unlimited	List of proposed items resulting of the search.

Items (AutosuggestItem)

property	type	min/max	description
name	string	0/1	Name of the item. Depending on the item's resultType, it contains : place : the place name. locality, street, houseNumber : an address label (same as addressLabel)
resultType	string	0/1	Item's result type. Possible values : - place - locality - street - houseNumber - addressBlock - intersection - postalCodePoint Only if extended = true: - administrativeArea - *other*
localityType	string	0/1	When resultType is locality, contains the locality type. Possible values : - city - district - subdistrict - postalCode Empty when resultType is not locality.
categories	array of string	0/ unlimited	When resultType is place, contains a the types of the place. The first element is always the main type. Empty when resultType is not place.
addressLabel	string	0/1	Contains an address label, that is a single string representing the item's address.

property	type	min/max	description
			The form of the address and the elements that compose it depend on the resultType.
location	string	0/1	Item coordinates. A coordinates couple as always *easting,northing* (*X,Y* or *longitude,latitude* depending on the coordinate system given in the srs request parameter). Example : *-2.33844,47.26113*
distance	number	0/1	required The distance of the item from the geographical search area given in the request. Always in meters.

Possible returns

The case of addresses that are found (status is OK)


```
{
    "status": "OK",
    "items": [
        {
            "name": "Musée du Louvre (Pyramide du Louvre)",
            "resultType": "place",
            "addressLabel": "Musée du Louvre, 99 Musée du Louvre, 75001 Paris, France",
            "location": "2.33754,48.86053",
            "distance": 7304.0,
            "categories": [
                "Musée d'histoire",
                "Lieu d'intérêt/Attraction",
                "Attraction touristique",
                "Musée",
                "Musée d'art"
            ]
        },
        {
            "name": "Métro-Palais Royal Musée du Louvre",
            "resultType": "place",
            "addressLabel": "Place du Palais-Royal, 75001 Paris, France",
            "location": "2.33652,48.86285",
            "distance": 7524.0,
            "categories": [
                "Métro"
            ]
        },
        {
            "name": "Louvres, Île-de-France, France",
            "resultType": "locality",
            "localityType": "city",
            "addressLabel": "Louvres, Île-de-France, France",
            "location": "2.50607,49.04632",
            "distance": 31242.0
        },
        {
            "name": "Métro-Louvre Rivoli",
            "resultType": "place",
            "addressLabel": "Rue de l'Amiral de Coligny, 75001 Paris, France",
            "location": "2.34094,48.86067",
            "distance": 7405.0,
            "categories": [
                "Métro"
            ]
        },
        {
```

```
            "name": "Le Louvre",
            "resultType": "place",
            "addressLabel": "1 Rue de Marengo, 75001 Paris, France",
            "location": "2.33913,48.86185",
            "distance": 7481.0,
            "categories": [
                "Parking à étage",
                "Parking"
            ]
        }
    ]
}
```


The case of parameter ***query*** is empty (status is ERROR)


```
{
    "message": "ServiceException: Autosuggest failed|Mandatory parameter 'query' must be defined and non-
empty",
    "status": "ERROR"
}
```


The case of missing one of the parameters ***location***, ***bounds*** or ***circle*** (status is ERROR)


```
{
    "message": "ServiceException: Autosuggest failed|Required parameter missing. One of mutually exclusive
 parameters 'location', 'bounds' or 'circle' must be present",
    "status": "ERROR"
}
```


Geocoding

Basic principles

The query includes an input address, the service returns one or several possible responses (if there is any ambiguity), including the recognised address, the position, the geocoding score, and the geocoding type.

Geocode example


```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/geocode/v4.json",
        "type":"GET",
        "params": {
                "addressLine": {
                        "sample":"25 rue de dolbiac"},
                "postCode": {
                        "sample":"75013"},
                "city": {
                        "sample":"Paris"},
                "countryCode": {
                        "sample":"FR"},
                "maxResponses" : {
                        "type":"number","min":1,"max":20,
                        "sample":5},
                "srs":{"sample":"wgs84"}
        }
}
```

Parameters / Properties

Input

parameter	description	optional	default
id	convenience identifier for this address. Optional : this identifier has no functional role, it might be used by the caller to identify individual input addresses to geocode and output in results	yes	
comment	convenience comment for this address	yes	
addressLine	address including number, repetition index, type of street and street name.	yes *	
addressLineExt	second address (with a street number and street name) to find if first is too bad	yes	
city	city	yes *	
district	district (part of city)	yes	
postCode	post code	yes *	
region	state, county, …	yes	
countryCode	country on two or three letters (ISO code 3166-1) for example, "fr" or "fra"	yes	
excludePlaces	Should responses show places, or only addresses?	yes	false
improve	try to get better results on the wrong candidates. The resulting output format may be slightly different from the candidates improved.	yes	false
maxResponses	maximum number of address results in the response	yes	5
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	Without any projection, the result will be wgs84
streetMinScore	the value of this parameters varies, as for the score, between 0 and 100. The streetMinScore has the effect of filtering possible candidates in the following way: If the candidate score is equal to or higher than the score, the candidate is selected as it is, If the candidate score is lower than the score and no candidate match has already been selected, the geocoding is suggested at city level 90 is the recommended value to avoid wrong positives candidates, 50 can be used in case of manual control of the candidates	yes	0

(*) At least one of the three parameters postCode, addressLine or city must be assigned a value.

Output

parameter	type	min/max	description
geocodedAddress (or geocodedAddresses in JSON / JSON-P)	geocodedAddress (or array in JSON / JSON-P)	0/ unlimited	Geocoded addresses
id	string	0/1	identifier used in input
comment	string	0/1	comment used in input

Geocoded addresses (geocodedAddress)

parameter	type	min/max	description
score	double	1/1	geocoding score from 0 to 100, with 100 for a perfect correspondance
geocodeType	int	1/1	type of geocoding: - not geocoded = 0 - town = 1 - street = 2 - street enhanced = 3 - street number = 4
geocodeTypeQuality	string	1/1	geocode quality: - "NOT_FOUND" - "ADDRESS_POINT" - "POI" - "NAMED PLACE" - "CITY+POSTCODE" - "CITY" - "DISTRICT+POSTCODE" - "DISTRICT" - "STREET" - "ZONE+POSTCODE" - "STREET NUMBER INTERPOLATION" - "STREET NUMBER BOUND" - "STREET NUMBER ESTIMATION" - "STREET NUMBER NEAREST" - "PONCTUAL POSTCODE"
x	double	1/1	X coordinates
y	double	1/1	Y coordinates
streetNumber	string	0/1	street number
streetWayType	string	0/1	type of street (avenue, street, etc)
streetWayName	string	0/1	street name
streetWay	string	0/1	full name of the street
addressLine	string	0/1	street found and, if there is one, the number
city	string	0/1	town found
region	string	0/1	State, County, … found, varies as a function of country, can also be empty
postCode	string	0/1	post code found
countryCode	string	1/1	Country on three letters (ISO code 3166-1)
places	Array of GeocodedAddressPlaceItem	0/unlimited	list of containing places. Output attribute values of the found address, according "placeTypes" (for example ["751010206","930005Y001XCHE"]). Depends on the country and the repository used.

Places (GeocodedAddressPlaceItem)

parameter	type	min/max	description
placeType	string	0/1	list of types of attributes (for example ["IRIS","INSEE","…"]). Depends on the country and the repository used.

parameter	type	min/max	description
placeValue	string	0/1	list of attributes. Value of attributes for the address found, in relation to *placeTypes* . Depends on the country and the repository used.

Possible returns

Case of an address found (status is OK)


```
{
    "status": "OK",
    "geocodedAddresses": [
        {
            "addressLine": "25 RUE DE TOLBIAC",
            "district": "13E ARRONDISSEMENT",
            "city": "PARIS",
            "countryCode": "FRA",
            "postCode": "75013",
            "score": 99.2,
            "geocodeType": 4,
            "geocodeTypeQuality": "ADDRESS_POINT",
            "x": 2.373478,
            "y": 48.828707,
            "streetNumber": "25",
            "streetWayType": "RUE",
            "streetWayName": "DE TOLBIAC",
            "streetWay": "RUE DE TOLBIAC",
            "places": [
                {
                    "placeType": "INSEE",
                    "placeValue": "75113"
                },
                {
                    "placeType": "IRIS",
                    "placeValue": "751135014"
                },
                {
                    "placeType": "LEVEL_1",
                    "placeValue": "France"
                },
                {
                    "placeType": "LEVEL_2",
                    "placeValue": "Île-de-France"
                },
                {
                    "placeType": "LEVEL_3",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_4",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_5",
                    "placeValue": "13e Arrondissement"
                }
            ]
        },
        {
            "addressLine": "VILLA TOLBIAC",
            "district": "13E ARRONDISSEMENT",
```

```
            "city": "PARIS",
            "countryCode": "FRA",
            "postCode": "75013",
            "score": 93.96,
            "geocodeType": 2,
            "geocodeTypeQuality": "STREET",
            "x": 2.369572,
            "y": 48.826988,
            "streetNumber": "",
            "streetWayType": "VILLA",
            "streetWayName": "TOLBIAC",
            "streetWay": "VILLA TOLBIAC",
            "places": [
                {
                    "placeType": "INSEE",
                    "placeValue": "75113"
                },
                {
                    "placeType": "IRIS",
                    "placeValue": "751135011"
                },
                {
                    "placeType": "LEVEL_1",
                    "placeValue": "France"
                },
                {
                    "placeType": "LEVEL_2",
                    "placeValue": "Île-de-France"
                },
                {
                    "placeType": "LEVEL_3",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_4",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_5",
                    "placeValue": "13e Arrondissement"
                }
            ]
        },
        {
            "addressLine": "PONT DE TOLBIAC",
            "district": "13E ARRONDISSEMENT",
            "city": "PARIS",
            "countryCode": "FRA",
            "postCode": "75013",
            "score": 86.91,
            "geocodeType": 2,
            "geocodeTypeQuality": "STREET",
            "x": 2.380356,
            "y": 48.832558,
            "streetNumber": "",
            "streetWayType": "PONT",
            "streetWayName": "DE TOLBIAC",
            "streetWay": "PONT DE TOLBIAC",
            "places": [
                {
                    "placeType": "INSEE",
                    "placeValue": "75113"
                },
```

```
                {
                    "placeType": "IRIS",
                    "placeValue": "751135099"
                },
                {
                    "placeType": "LEVEL_1",
                    "placeValue": "France"
                },
                {
                    "placeType": "LEVEL_2",
                    "placeValue": "Île-de-France"
                },
                {
                    "placeType": "LEVEL_3",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_4",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_5",
                    "placeValue": "13e Arrondissement"
                }
            ]
        },
        {
            "addressLine": "PONT DE TOLBIAC",
            "district": "12E ARRONDISSEMENT",
            "city": "PARIS",
            "countryCode": "FRA",
            "postCode": "75012",
            "score": 86.82,
            "geocodeType": 2,
            "geocodeTypeQuality": "STREET",
            "x": 2.381517,
            "y": 48.833207,
            "streetNumber": "",
            "streetWayType": "PONT",
            "streetWayName": "DE TOLBIAC",
            "streetWay": "PONT DE TOLBIAC",
            "places": [
                {
                    "placeType": "INSEE",
                    "placeValue": "75112"
                },
                {
                    "placeType": "IRIS",
                    "placeValue": "751124799"
                },
                {
                    "placeType": "LEVEL_1",
                    "placeValue": "France"
                },
                {
                    "placeType": "LEVEL_2",
                    "placeValue": "Île-de-France"
                },
                {
                    "placeType": "LEVEL_3",
                    "placeValue": "Paris"
                },
                {
```

```
                    "placeType": "LEVEL_4",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_5",
                    "placeValue": "12e Arrondissement"
                }
            ]
        },
        {
            "addressLine": "25 RUE NEUVE TOLBIAC",
            "district": "13E ARRONDISSEMENT",
            "city": "PARIS",
            "countryCode": "FRA",
            "postCode": "75013",
            "score": 85.03,
            "geocodeType": 4,
            "geocodeTypeQuality": "ADDRESS_POINT",
            "x": 2.377029,
            "y": 48.830678,
            "streetNumber": "25",
            "streetWayType": "RUE",
            "streetWayName": "NEUVE TOLBIAC",
            "streetWay": "RUE NEUVE TOLBIAC",
            "places": [
                {
                    "placeType": "INSEE",
                    "placeValue": "75113"
                },
                {
                    "placeType": "IRIS",
                    "placeValue": "751135017"
                },
                {
                    "placeType": "LEVEL_1",
                    "placeValue": "France"
                },
                {
                    "placeType": "LEVEL_2",
                    "placeValue": "Île-de-France"
                },
                {
                    "placeType": "LEVEL_3",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_4",
                    "placeValue": "Paris"
                },
                {
                    "placeType": "LEVEL_5",
                    "placeValue": "13e Arrondissement"
                }
            ]
        }
    ]
}
```


Case of an address that is not found (status is OK and no geocodedAddress)


```
{
    "status": "OK",
    "geocodedAddresses": [
```

```
    ]
}
```

Case of a query with a non-existant reprojection system (status is ERROR)

```
{
    "message":"ServiceException: Geocode failed\nGeocode failed\nFailed to process geocoding task
\nUnsupported coordinate system 'epsg:5987'",
    "status":"ERROR"
}
```

FAQ

1. How is the score obtained in the *score* to be interpreted?

   A strict correspondance (equivalent to a score of 100) is equivalent to a *no tolerance* solution in effect, for a file that must not contain any errors at all. Choosing this score, you only accept suggestions of perfect matches between the reference table and the address sought. The higher the tolerance (in other words, the lower this score is set) the more the user accepts letters that are close matches between the addresses in the file to geocode and the reference table. The more likely it will be, also, that the service will assign false coordinates to certain addresses in the file due to a faulty interpretation. It is recommended that you only keep any candidates scoring 90 or higher. This score is a good compromise to the extent that one or two approximations between the addresses to geocode and the reference table are tolerated. It is strongly recommended that candidates with a score of less that 75 are NOT retained: this corresponds to at least 4 spelling or syntax mistakes in the address. Different criteria will have a bearing on the calculation of this score:

   - the number of equivalent letters in the two strings;
   - the number of words;
   - the length of each word making up the string.

   In the case of correspondances between addresses, only the names of streets have a bearing on the search for a sppelling match. The type of street has an additional incidence on the score, but only in the case where the type of street sought and the one suggested are identical.

2. How many characters are returned in the tag *postCode* ?

   Depends on the characteristics of postal codes for each country. Moreover, the returned string will be truncated to the first N characters if the address is not precise enough.

3. What rules exist with regard to X-Y coordinates (format, coordinates range)?

   These are names in a decimal format, with . as a separator of the decimal part. In the case where the geographic coordinates are in degrees, the return must be included within the range -180 and 180 (X=longitude) and -90 and 90 (Y=latitude), with 6 figures after the comma. In the case of projected coordinates, the result is in meters, with two figures after the comma.

4. Is this service always capable of finding a close-matching address?

   If the town does not exist, it will return an empty list.

5. How are *Cedex* entities handled?

   A filter is applied to not take Cedex and the codes that follow it into account during the search for addresses.

6. Can you have all the components of an address as a single input item, in just one field?

   Yes, the *addressline* component can be used to perform a fulltext geocoding via a unique field containing the concatenation of all fields: street number, type of street, street name, post code and name of the town. The addresses input via the *addressLine* component are analysed in two phases:

   - The mechanism for searching on word pattern will find, in the first instance, the town or towns present in the string. A search on the pattern then very quickly finds a label corresponding to a dictionary, in this case a dictionary of towns. This label can be badly spelt, with predictable or repeated errors in the user's entering text at the keyboard when using Internet: duplicated latters, letters omitted, replacement of one letter with another given a similar sound to the resulting word (C instead of K, for example) part of a town name where the town name consists of several sub-entities. Following these lines of enquiry, a series of candidate towns is found in the shortest of timescales.
   - Next, the pairs of towns found + the initial string without the "town" part that has just been recognised are given to the geocoder to perform the usual geocoding operation.

   Note: when the name of a town is also present in the name of a street, for example, "12 Rue d'Aix Antibes" there are potentially two candidates: Rue d'Aix à Antibes, and Rue d'Antibes à Aix. The best candidate is determined by its relative position in the string.

Geocoding (batch)

Basic principles

The query includes an address or several addresses as input, the service returns for each item, one or several possible responses (if there is any ambiguity), including the address recognised, the position, the geocoding score and the type of geocoding.

Batch geocoding example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/geocode/batch/v4.json",
        "method":"POST",
        "body": {
      "addresses": [
          {
            "addressLine" : "200 Quai Charles de Gaulle",
            "city" : "Lyon",
            "region" : "",
            "countryCode" : "FR",
            "postCode" : "69006"
          },
          {
            "addressLine" : "Bruno Kreisky Platz 1",
            "city" : "Wien",
            "region" : "",
            "countryCode" : "AT",
            "postCode" : "1220"
          },
          {
            "addressLine" : "Route des Morillons 15",
            "city" : "Genève",
```

```
            "region" : "",
            "countryCode" : "CH",
            "postCode" : "1202"
          }
      ],
      "streetMinScore": 80,
      "srs": "epsg:4326",
      "maxResponses": 1,
      "improve": false,
      "excludePlaces": true
    }
}
```

Parameters / properties

Input

parameter	description	optional	default
addresses array of GeocodeRequestV4	The addresses to geocode	no	
improve	try to get better results on the wrong candidates. The resulting output format may be slightly different from the candidates improved.	yes	false
maxResponses	maximum number of address results in the response (max = 5)	yes	1
srs	projection (EPSG code such as epsg:4326 or wgs 84)	yes	Without projection, the result is in native projection of the geocoding index, usually wgs84.
streetMinScore	the value of this parameters varies, as for the score, between 0 and 100. The streetMinScore has the effect of filtering possible candidates in the following way: If the candidate score is equal to or higher than the score, the candidate is selected as it is, If the candidate score is lower than the score and no candidate match has already been selected, the geocoding is suggested at city level 90 is the recommended value to avoid wrong positives candidates, 50 can be used in case of manual control of the candidates	yes	0
excludePlaces	Should responses show places, or only addresses?	yes	false

Addresses (GeocodeRequestV4)

parameter	description	optional	default
id	convenience identifier for this address. Optional : this identifier has no functional role, it might be used by the caller to identify individual input addresses to geocode and output in results	yes	
comment	convenience comment for this address	yes	
addressLine	address including the number, repetition index, type of street and street name.	yes *	

parameter	description	optional	default
addressLineExt	second address (with a street number and street name) to find if first is too bad	yes	
city	town	yes *	
district	district (part of city)	yes	
postCode	post code	yes *	
region	State, County, …	yes	
countryCode	country on two or three letters (ISO code 3166-1) for example, "fr" or "fra"	yes	

(*) At least one of the three parameters postCode, addressLine and city must be filled.

Output

parameter	type	min/max	description
geocodedAddresses	array of GeocodedAddressV4	0/ unlimited	Geocoded addresses
id	string	0/1	identifier used in input
comment	string	0/1	comment used in input

Geocoded addresses (GeocodedAddressV4)

parameter	type	min/max	description
score	double	1/1	geocoding score from 0 to 100, with 100 for a perfect correspondance
geocodeType	int	1/1	type of geocoding: - not geocoded = 0 - town = 1 - street = 2 - street enhanced = 3 - street number = 4
geocodeTypeQuality	string	1/1	geocode quality: - "NOT_FOUND" - "ADDRESS_POINT" - "POI" - "NAMED PLACE" - "CITY+POSTCODE" - "CITY" - "DISTRICT+POSTCODE" - "DISTRICT" - "STREET" - "ZONE+POSTCODE" - "STREET NUMBER INTERPOLATION" - "STREET NUMBER BOUND" - "STREET NUMBER ESTIMATION" - "STREET NUMBER NEAREST" - "PONCTUAL POSTCODE"
x	double	1/1	X coordinates
y	double	1/1	Y coordinates
streetNumber	string	0/1	street number

parameter	type	min/max	description
streetWayType	string	0/1	type of street (avenue, street, etc)
streetWayName	string	0/1	street name
streetWay	string	0/1	full name of the street
addressLine	string	0/1	street found and, if there is one, the number
city	string	0/1	town found
region	string	0/1	State, County, … found, varies as a function of country, can also be empty
postCode	string	0/1	post code found
countryCode	string	1/1	Country on three letters (ISO code 3166-1)
places	Array of GeocodedAddressPlaceItem	0/unlimited	list of containing places. Output attribute values of the found address, according "placeTypes" (for example ["751010206","930005Y001XCHE"]). Depends on the country and the repository used.

Places (GeocodedAddressPlaceItem)

parameter	type	min/max	description
placeType	string	0/1	list of types of attributes (for example ["IRIS","INSEE","…"]). Depends on the country and the repository used.
placeValue	string	0/1	list of attributes. Value of attributes for the address found, in relation to *placeTypes* . Depends on the country and the repository used.

Initial address (initialAddress)

parameter	type	min/max	description
addressLine	string	0/1	street found and, where appropriate, the number
city	string	0/1	town found
region	string	0/1	State, County, found, varies as a function of country, and could also be empty
countryCode	string	1/1	cf. description of the input parameter
postCode	string	0/1	post code found

Possible returns

Case of an address found (status is OK)


```
{
    "status": "OK",
    "results": [
        {
            "status": "OK",
            "geocodedAddresses": [
                {
                    "addressLine": "200 QUAI CHARLES DE GAULLE",
                    "district": "6E ARRONDISSEMENT",
                    "city": "LYON",
                    "countryCode": "FRA",
                    "postCode": "69006",
                    "score": 100.0,
```

```
                    "geocodeType": 4,
                    "geocodeTypeQuality": "ADDRESS_POINT",
                    "x": 4.847616,
                    "y": 45.782431,
                    "streetNumber": "200",
                    "streetWayType": "QUAI",
                    "streetWayName": "CHARLES DE GAULLE",
                    "streetWay": "QUAI CHARLES DE GAULLE",
                    "places": [
                        {
                            "placeType": "INSEE",
                            "placeValue": "69386"
                        },
                        {
                            "placeType": "IRIS",
                            "placeValue": "693860101"
                        },
                        {
                            "placeType": "LEVEL_1",
                            "placeValue": "France"
                        },
                        {
                            "placeType": "LEVEL_2",
                            "placeValue": "Auvergne-Rhône-Alpes"
                        },
                        {
                            "placeType": "LEVEL_3",
                            "placeValue": "Rhône"
                        },
                        {
                            "placeType": "LEVEL_4",
                            "placeValue": "Lyon"
                        },
                        {
                            "placeType": "LEVEL_5",
                            "placeValue": "6e Arrondissement"
                        }
                    ]
                }
            ]
        },
        {
            "status": "OK",
            "geocodedAddresses": [
                {
                    "addressLine": "BRUNO-KREISKY-PLATZ 1",
                    "district": "22. BEZIRK-DONAUSTADT",
                    "city": "WIEN",
                    "countryCode": "AUT",
                    "postCode": "1220",
                    "score": 100.0,
                    "geocodeType": 4,
                    "geocodeTypeQuality": "ADDRESS_POINT",
                    "x": 16.413865,
                    "y": 48.234572,
                    "streetNumber": "1",
                    "streetWayType": "",
                    "streetWayName": "BRUNO-KREISKY-PLATZ",
                    "streetWay": "BRUNO-KREISKY-PLATZ",
                    "places": [
                        {
                            "placeType": "INSEE",
                            "placeValue": ""
```

```
                        },
                        {
                            "placeType": "IRIS",
                            "placeValue": ""
                        },
                        {
                            "placeType": "LEVEL_1",
                            "placeValue": "Österreich"
                        },
                        {
                            "placeType": "LEVEL_2",
                            "placeValue": "Wien"
                        },
                        {
                            "placeType": "LEVEL_3",
                            "placeValue": "Wien"
                        },
                        {
                            "placeType": "LEVEL_4",
                            "placeValue": "Wien"
                        },
                        {
                            "placeType": "LEVEL_5",
                            "placeValue": "22. Bezirk-Donaustadt"
                        }
                    ]
                }
            ]
        },
        {
            "status": "OK",
            "geocodedAddresses": [
                {
                    "addressLine": "ROUTE DES MORILLONS 15",
                    "district": "",
                    "city": "GENÈVE",
                    "countryCode": "CHE",
                    "postCode": "1218",
                    "score": 99.98,
                    "geocodeType": 3,
                    "geocodeTypeQuality": "STREET NUMBER NEAREST",
                    "x": 6.13245,
                    "y": 46.231514,
                    "streetNumber": "",
                    "streetWayType": "ROUTE",
                    "streetWayName": "DES MORILLONS",
                    "streetWay": "ROUTE DES MORILLONS",
                    "places": [
                        {
                            "placeType": "INSEE",
                            "placeValue": ""
                        },
                        {
                            "placeType": "IRIS",
                            "placeValue": ""
                        },
                        {
                            "placeType": "LEVEL_1",
                            "placeValue": "Suisse"
                        },
                        {
                            "placeType": "LEVEL_2",
                            "placeValue": "Genève"
```

```
                        },
                        {
                            "placeType": "LEVEL_3",
                            "placeValue": "Genève"
                        },
                        {
                            "placeType": "LEVEL_4",
                            "placeValue": "Genève"
                        },
                        {
                            "placeType": "LEVEL_5",
                            "placeValue": ""
                        }
                    ]
                }
            ]
        }
    ]
}
```


Case of an address that is not found (status is OK and no geocodedAddress)


```
{
  "message": null,
  "status": "OK",
  "results": [
    {
      "message": null,
      "status": "OK",
      "geocodedAddresses": [],
      "initialAddress": {
        "addressLine": "",
        "city": "#sdsdsdsds",
        "region": "",
        "countryCode": "FR",
        "postCode": ""
      }
    }
  ]
}
```


Case of a query with a non-existent reprojection system ⇒ error with faultstring that contains the description

```
An error append : (500) Internal Server Error
ServiceException: Geocode failed
Geocode failed
Failed to process geocoding task
Unsupported coordinate system 'wgs841'
```

FAQ

See Geocoding Web service.

1. How to use the cURL command?

   It is necessary to use an address file, here in json format, with the following command line:

```
curl -X POST 'https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/geocode/batch/v4.json?
appkey=XXXXX&apptoken=YYYYY' -H 'Content-Type: application/json' -F file=@"adresses.json"
```

File: adresses.json

```
{
   "addresses":[
      {
         "addressLine":"200 Quai Charles de Gaulle",
         "city":"Lyon",
         "region":"",
         "countryCode":"FR",
         "postCode":"69006"
      },
      {
         "addressLine":"Bruno Kreisky Platz 1",
         "city":"Wien",
         "region":"",
         "countryCode":"AT",
         "postCode":"1220"
      },
      {
         "addressLine":"Route des Morillons 15",
         "city":"Genève",
         "region":"",
         "countryCode":"CH",
         "postCode":"1202"
      }
   ],
   "streetMinScore":80,
   "srs":"epsg:4326",
   "maxResponses":2
}
```

Address abbreviation

Basic principles

The query takes as input an address (broken down into an address line, post code, town) and the size of the required standardised abreviation (32 or 38).

The service returns as output an abbreviation of the address line part following the rules expressed in the corresponding AFNOR standards (32 character standard and 38 character standard).

Concerning the restitution function of the 38-character address line, only the AFNOR NF Z10-011 standard dating from January 2013 (in 38 characters) will be applicable.

Concerning the restitution function of the 32-character address line;

- In the first place the AFNOR NF Z10-011 standard from January 2013 (in 38 characters) will be aplicable:
  - with regard to the rules for reduction, but up to the target of 32 characters,
  - with regard to the abbreviation of words, including those that have not been defined in Appendix B of the standard NF Z10-011 dating from August 1989 (in 32 characters).
- The state of play currently is that, following the application of AFNOR standard NF Z10-011 in January 2013, the NF Z10-011 standard dating from August 1989 (in 32 characters) will be applicable in relation to abbreviations of words, but only if the maximum length requested is 32 and if the resulting length remains higher than 32 characters. In this case, it will be limited to application of abbreviations

from Appendix B of the NF Z10-011 standard dating from August 1989 (in 32 characters) for which the AFNOR standard NF Z10-011 of January 2013 in its various appendices does not suggest any abbreviation.

This service interrogates files of abbreviations, copying the appendices of the AFNOR standards cited.

Address abbreviation example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/normalizer.json",
        "type":"GET",
        "params": {
                "addressLine": {
                        "sample":"25 Boulevard du Maréchal de Lattre de Tassigny"
                },
                "postCode": {
                        "sample":"77140"
                },
                "city": {
                        "type":"string",
                        "sample":"Nemours"
                },
                "maxLength": {
                        "sample":"32"
                }
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
maxLength	maxLength defines, in the form of an integer, the reference size to attain for the abbreviation of the address line. Values accepted are exclusivement 32 or 38.	yes	38
addressLine	The address line receiving, EITHER the street in the form number, repetition index, type of street and street name, OR the place name (in France this is the LIEU-DIT). The accuracy of the address abbreviation as output from the WS can only be ensured on condition that an address line parameter resulting from a validated address using an Address repository is supplied.	no	
city	Town corresponding to the address line entered. No treatment will be applied to this parameter.	yes	
postCode	Post code corresponding to the address line supplied. No treatment will be applied to this parameter.	yes	

Output

parameter	type	min/max	description
id	long	1/1	Automated increment of the response identifier.
status	string	0/1	Indicates the success or failure of the abbreviation process «OK» or «ERROR».
statusDetails	string	0/1	Empty, in the case of a succesful abbreviation process. With the corresponding error message, in the event of failure,

parameter	type	min/max	description
			"Invalid maxLength parameter. It needs to be either 32 or 38". "The address contraction can't reach the expected maxLength". "The supplied addressline was above 100 characters".
maxLength	int	1/1	Takes the value supplied in input «32» or «38».
address	array (normalizedAddress)	0/1	Standardised address.
normedAddressLineLength	int	1/1	Length of the address line obtained, following the operations of abbreviation applied by the service.

Standardised address (normalizedAddress)

parameter	type	min/max	description
normedAddressLine	string	0/1	Final result of the contraction of the address line applied by the service.
city	string	0/1	Returns the town supplied as input, without modification, (otherwise empty).
postCode	string	0/1	Returns the post code supplied as input, without modification, (otherwise empty).

Possible returns

case of a standardised address (status is OK)

```
{
  "id": 1,
  "status": "OK",
  "statusDetails": "",
  "maxLength": 32,
  "address": {
    "normedAddressLine": "25 BD MAL DE LATTRE DE TASSIGNY",
    "city": "Nemours",
    "postCode": "77140"
  },
  "normedAddressLineLength": 31
}
```

Case of setting the reduction size to a value other than «32» or «38» (status is ERROR)

```
{
  "id": 2,
  "status": "ERROR",
  "statusDetails": "Invalid maxLength parameter. It needs to be either 32 or 38",
  "maxLength": 0,
  "address": {
    "normedAddressLine": "",
    "city": "",
    "postCode": ""
  },
  "normedAddressLineLength": 0
}
```

Case of an address of more than 100 characters (status is ERROR)

```
{
```

```
  "id": 3,
  "status": "ERROR",
  "statusDetails": "The supplied addressline was above 100 characters",
  "maxLength": 0,
  "address": {
    "normedAddressLine": "",
    "city": "",
    "postCode": ""
  },
  "normedAddressLineLength": 0
}
```

Case of failure in contraction of the address line in terms of the expected size (status is ERROR)

```
{
  "id": 4,
  "status": "ERROR",
  "statusDetails": "The address contraction can't reach the expected maxLength",
  "maxLength": 0,
  "address": {
    "normedAddressLine": "125 B BD MAL J B F X L TASSIGNY INF",
    "city": "Nemours",
    "postCode": "77140"
  },
  "normedAddressLineLength": 35
}
```

Reverse geocoding

Basic principles

The query includes a pair of coordinates as input, the service returns one or several possible responses (depending on the maximum number of responses desired), including the address, the post code, the town, the country, and the distance between the result and the start point (coordinates indicated as input).

Reverse geocoding example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/reverseGeocoding/v4.json",
        "type":"GET",
        "params": {
                "locations": {

 "sample":"14.411633,50.086454;-0.127515,51.503264;13.377888,52.516224;21.014335,52.225451"
                },
                "srs": {
                        "sample":"epsg:4326"
                },
                "maxDistance": {
                        "sample":"1000"
                },
                "maxDistanceBetweenCandidates": {
                        "sample":"100"
                },
                "maxCandidates": {
                        "sample":"2"
                },
                "fields": {
                        "sample":"Postcode left; Postcode right"
```

```
                }

        }
}
```

Parameters / properties

Input

parameter	description	optional	default
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	
locations	Coordinates for points separated by “,”. Pair of XY coordinates are separated by “;”.	no	
maxDistance	maximum search distance to find neighbouring streets at the reverse geocoding start point if the parameter is passed, it is advisable to assign a value that is significantly higher than 1 (in meters).	yes	1000
maxCandidates	maximum number of result addresses in the response if the parameter is passed, the minimum value to indicate is 1.	yes	1
maxDistanceBetweenCandidates	Maximum distance (in meters) between each candidate. if the parameter is passed, it is advisable to assign a value that is significantly higher than 1 (in meters).	yes	100
fields	List of fields to include in the response, separated by a *;* character Available fields are: End right number, Begin right number, Begin left number, End left number, Iso country code, Country, Reverse geocoding name, Postcode, Postcode left, Postcode right, City name, City name left, City name right, Provider link id, TollPayPoint, Route type, Ramp, Urban, Tunnel, Toll, Bicycles, Automobiles, Frontage, Bridge	yes	[]

Output

Reminder for the coordinates of the start point of the search(reverseGeocodingV4Response)

property	type	min/max	description
location	string	0/1	X,Y coordinates for the start point for the reverse geocoding
srs	string	0/1	projection (EPSG code such as epsg:4326 or wgs84)
addresses		0/ unlimited	list of (candidate) addresses returned

Candidates returned (addresses)

property	type	min/max	description
addressLine	string	0/1	street found
postCode	string	0/1	post code found
city	string	0/1	town associated with the candidate returned
country	string	0/1	country associated to the candidate returned
distanceMetersToLocation	double	1/1	Distance (in meters) between the candidate returned and the start point

property	type	min/max	description
coordinates	string	1/1	Coordinates of the address
streetSide	string		"right" or "left"
fieldsValues	string		values for the requested fields on the street found

Possible returns

Example of a correct reverse geocoding (status is OK)


```
{
    "status": "OK",
    "reverseGeocodingResults":[
    {
      "location": "14.411633,50.086454",
      "addresses":[
      {
        "addressLine": "KARL#V MOST",
        "postCode": "118 00",
        "city": "PRAHA 1",
        "country": "#esko",
        "distanceMetersToLocation": 0.61,
        "coordinates": "14.411635,50.086459",
        "streetSide": "Left",
        "fieldsValues":["118 00", "118 00"]
      },
      {
        "addressLine": "KARL#V MOST",
        "postCode": "118 00",
        "city": "PRAHA 1",
        "country": "#esko",
        "distanceMetersToLocation": 40.46,
        "coordinates": "14.41108,50.08653",
        "streetSide": "Left",
        "fieldsValues":["118 00", "118 00"]
      }
      ]
    },
    {
      "location": "-0.127515,51.503264",
      "addresses":[
      {
        "addressLine": "9 DOWNING STREET",
        "postCode": "SW1A 2",
        "city": "SW1",
        "country": "England",
        "distanceMetersToLocation": 6.24,
        "coordinates": "-0.127521,51.503208",
        "streetSide": "Right",
        "fieldsValues":["SW1A 2", "SW1A 2"]
      },
      {
        "addressLine": "10 DOWNING STREET",
        "postCode": "SW1A 2",
        "city": "SW1",
        "country": "England",
        "distanceMetersToLocation": 7.11,
        "coordinates": "-0.12757,51.50321",
        "streetSide": "Right",
        "fieldsValues":["SW1A 2", "SW1A 2"]
      }
```

```
      ]
    },
    {
      "location": "13.377888,52.516224",
      "addresses":[
      {
      "addressLine": "PARISER PLATZ",
      "postCode": "10117",
      "city": "MITTE",
      "country": "Deutschland",
      "distanceMetersToLocation": 8.04,
      "coordinates": "13.377873,52.516296",
      "streetSide": "Right",
      "fieldsValues":["10117", "10117"]
      },
      {
      "addressLine": "PARISER PLATZ",
      "postCode": "10117",
      "city": "MITTE",
      "country": "Deutschland",
      "distanceMetersToLocation": 9.46,
      "coordinates": "13.3778,52.51629",
      "streetSide": "Right",
      "fieldsValues":["10117", "10117"]
      }
      ]
    },
    {
      "location": "21.014335,52.225451",
      "addresses":[
      {
        "addressLine": "ULICA MARSZA#KOWSKA",
        "postCode": "00-683",
        "city": "#RÓDMIE#CIE",
        "country": "Polska",
        "distanceMetersToLocation": 8.3,
        "coordinates": "21.014221,52.225425",
        "streetSide": "Right",
        "fieldsValues":["00-683", "00-683"]
      },
      {
        "addressLine": "ULICA MARSZA#KOWSKA",
        "postCode": "00-545",
        "city": "#RÓDMIE#CIE",
        "country": "Polska",
        "distanceMetersToLocation": 9.92,
        "coordinates": "21.014471,52.225482",
        "streetSide": "Left",
        "fieldsValues":["00-545", "00-545"]
      }
      ]
    }
    ],
    "fieldsNames":[
      "Postcode left",
      "Postcode right"
    ]
}
```


Case of a query that does not snap (status is ERROR)

```
An error append : (500) Internal Server Error
ServiceException: Error in reverse geocoding computation
```

```
Error in smartrouting
Failed to execute ReverseGeocode
com.geoconcept.smartrouting.SmartRoutingNativeException: failed to connect { 14.411633, 50.086454,
 0.000000 }
failed to connect { 14.411633, 50.086454, 0.000000 }
```

FAQ

1. How is the maximum distance interpreted?

   The <maxDistance> tag is mandatory and must have a value higher than 1. Take care, nonetheless, if the maximum distance is not high enough, as the risk would be not to have any candidate returned at all.

2. What is the role of the maxCandidates parameter?

   This parameter allows you to define the number of responses you want to obtain, in the event that there are a large number of potential candidates. If no value is indicated, the value assigned by default is 1.

3. What is the significance of the maxDistanceBetweenCandidates parameter?

   If this parameter is filled in incorrectly, the webservice cannot return a positive response. We strongly advise assigning a value of > 1. If no value is mentioned, the value assigned by default is 100. There is a significant risk of not having any candidates returned at all.

Find an object

Basic principles

This web service uses an object's geographic coordinates to retrieve information contained in the fields of the object in a Geoconcept map. The call to the web service requires as parameters the layer name, the structure of the map and of the field(s) to interrogate.

Find an object example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/find/findObject/v2.json",
        "method":"POST",
    "body":
    {
       "srs":"epsg:4326",
       "geometries":"1,-5.617883,36.027288",
       "layerName":"STANDARD",
       "targets":[
          {
             "className":"Administrative unit",
             "subClassName":"Order1",
             "fields":"Name;Government code"
          },
          {
             "className":"Administrative unit",
             "subClassName":"Order2",
             "fields":"Name;Government code"
          },
          {
             "className":"Administrative unit",
             "subClassName":"Order8",
```

```
            "fields":"Name;Government code"
         }
      ]
   }
}
```

Parameters / properties

Input

parameter	description	optional	default
srs	projection (EPSG code such as epsg:4326 or wgs84)	no	
geometries	Coordinates to find (series of Id,X,Y triplets separated by the ";" character)	no	
layerName	Name of the layer to use	no	
targets	Properties of searched objects	no	

Targets

parameter	description	optional	default
targetID	Identifier	yes	
className	Name of the Class	no	
subclassName	Name of the Subclass	no	
fields	Name of the fields separated by the ";" character	no	
maxDistance	Maximum search radius (in meters)	yes	
maxCandidates	Maximum number of candidates to return	yes	

Output

parameter	type	min/max	description
id	string	0/1	Objects identifier
targets	array (Targets Infos)	0/ unlimited	Objects

Target Infos (findTargetResult)

parameter	type	min/max	description
targetID	string	0/1	Identifier
objects array (Objects Info)	string	0/ unlimited	Object descriptions

Objects Info

parameter	type	min/max	description
distance	integer	1/1	Distance in meters betwen the origin point and the found object -1 if the calculated distance is not available
wktGeometry	string	0/1	Object geometry

Possible returns

Case of a response that is found (findObjectsResults/status est OK)

```
{
  "message": null,
  "status": "OK",
  "results": [
    {
      "id": "1",
      "targets": [
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Andalucía",
                "01"
              ]
            }
          ]
        },
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Cádiz",
                "11"
              ]
            }
          ]
        },
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Tarifa",
                "11035"
              ]
            }
          ]
        }
      ]
    }
  ]
}
```


Absence of an object on the position searched (status is OK)


```
{
  "message": null,
  "status": "OK",
  "results": [
    {
      "id": "1",
      "targets": [
        {
          "objects": []
        },
        {
          "objects": []
        },
        {
          "objects": []
```

```
        }
      ]
    }
  ]
}
```

Faulty projection / srs (status is ERROR)

```
An error append : (500) Internal Server Error
CoordinateTransformEngineException: Coordinate tranformation failed
```

Case, absence of an argument for ***geometries*** (status est ERROR)

```
An error append : (500) Internal Server Error
IllegalArgumentException: Geometry has too few fields : String[][{1,-45.617883}]
```

Case of a layer that is not found (status is ERROR)

```
An error append : (500) Internal Server Error
WebServiceException: Layer name 'Europ' does not exist
```

Case of a Class that is not found (status est ERROR)

```
An error append : (500) Internal Server Error
WebServiceException: Error in finding objects
failed to execute text request
failed to execute gcis request (text response)
failed to execute gcis request
exception occured while servicing request
Failed to service request
com.geoconcept.gc.GcException: native returned exception (code=1)
native returned exception
[FindObject-88304] unknown type 'Administrative uni'
```

Case of a Subclass that is not found (status est ERROR)

```
An error append : (500) Internal Server Error
WebServiceException: Error in finding objects
failed to execute text request
failed to execute gcis request (text response)
failed to execute gcis request
exception occured while servicing request
Failed to service request
com.geoconcept.gc.GcException: native returned exception (code=1)
native returned exception
[FindObject-88305] unknown subtype 'Order'
```

Case of a field not found (serviceResult/status est ERROR)

```
An error append : (500) Internal Server Error
WebServiceException: Error in finding objects
failed to execute text request
failed to execute gcis request (text response)
failed to execute gcis request
exception occured while servicing request
Failed to service request
com.geoconcept.gc.GcException: native returned exception (code=1)
native returned exception
```

```
[FindObject-88300] unknown field 'Government cod'
```

FAQ

1. What are the *layerName* availables?

   For the moment, only the *STANDARD* layer is available.

2. What are the *className*, *subClassName* and *fieds* availables?

   See list below:

className	subClassName	fields
Administrative unit	Country	Name, Country code (ISO), Country code (EBU)
	Order8	Name, Government code
	Order2	Name, Government code
	Order1	Name, Government code
Postal boundaries	Postal areas	Name, Postcode, Country code (ISO), Admin1, Admin2, Admin3, Admin4, Admin5

> The coverage of the Administrative unit and Postal boundaries is not homogeneous, it depends on the administrative levels of each country and of the avaibility of the data providers. Please contact us for specific coverage.

Sample on fews coutries

Country	Postal Boundary	Postal items	Order1	Order1 items	Order1 example	Order2	Order2 items	Order2 example	Order8	Order8 items	Order8 example
France	Yes	60 53	Yes	14	Ile de France (11)	Yes	97	Tarn (81)	Yes	36 560	Castres (81065)
Belgium	Yes	1 147	Yes	3	Bruxelles (4)	Yes	11	Anvers (01)	Yes	589	Saint-Gilles (00083)
Germany	Yes	8 203	Yes	16	Schleswig-Holstein (01)	Yes	401	Hamburg (02000)	Yes	11 130	Flensburg (01001000)
Spain	Yes	10 806	Yes	19	Aragón (02)	Yes	52	Alicante (03)	Yes	8 126	Cerdido (15025)
Italy	Yes	4 605	Yes	20	Sardegna (20)	Yes	107	Torino (001)	Yes	7 991	Trinitapoli (110010)
Ireland	Yes	23	Yes	26	Dublin (26)	N/A			N/A		
Luxembourg	Yes	3 982	Yes	3	Luxembourg (1)	Yes	12	Esch-sur-Alzette (01)	Yes	106	Kiischpelt (00871)

Country	Postal Boundary	Postal items	Order1	Order1 items	Order1 example	Order2	Order2 items	Order2 example	Order8	Order8 items	Order8 example
Netherlands	Yes	4 043	Yes	12	Groningen (20)	N/A			Yes	388	Molenwaard (6452)
Portugal	Yes	532	Yes	29	Aveiro (1)	N/A			Yes	308	Horta (1)
United Kingdom	Yes	9 671	Yes	2	Guernsey (1)	N/A			N/A		
Switzerland	Yes	3 184	Yes	26	Jura (26)	Yes	148	Genève (25)	Yes	2 240	Zurich (0261)
Greece	Yes	1 083	Yes	13	Ανατολική Μακεδονία Θράκη (11)	Yes	51	Χανιά (434)	Yes	1 034	Γαύδος : (9462)
Brazil	Yes	18 263	Yes	5	Norte (1)	Yes	27	Oropeza (101)	Yes	327	Sucre (10101)
…											

3. Is it possible to use other topological relationships between objects?

   No, for the moment, only the *Intersect* relationship is available.

4. Is it possible to pass several *geometries* when calling a Web Service?

   Yes, it will suffice to specify more triplets separated by the ";" character

```
{
  "srs": "epsg:4326",
  "geometries": "1,-5.617883,36.027288;2,25.825734,71.109870",
  "layerName": "Europe",
  "targets": [
    {
      "className": "Administrative unit",
      "subClassName": "Order1",
      "fields": "Name;Government code"
    },
    {
      "className": "Administrative unit",
      "subClassName": "Order2",
      "fields": "Name;Government code"
    },
    {
      "className": "Administrative unit",
      "subClassName": "Order8",
      "fields": "Name;Government code"
    }
  ]
}
```

   Return:

```
{
  "message": null,
  "status": "OK",
  "results": [
    {
      "id": "1",
```

```
      "targets": [
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Andalucía",
                "01"
              ]
            }
          ]
        },
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Cádiz",
                "11"
              ]
            }
          ]
        },
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Tarifa",
                "11035"
              ]
            }
          ]
        }
      ]
    },
    {
      "id": "2",
      "targets": [
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Finnmark",
                "20"
              ]
            }
          ]
        },
        {
          "objects": []
        },
        {
          "objects": [
            {
              "distance": 0,
              "fields": [
                "Nordkapp",
                "2019"
              ]
            }
          ]
```

```
        }
      ]
    }
  ]
}
```

Territory management

Territory management entails the fair apportionment of points between several geographical areas based on appropriate criteria for the activity in question: balancing of accounts between sales personnel based on potential turnover, apportionment of delivery areas around warehouses, assignment of customers requiring assistance taking account of the technicians' expertise.

A dedicated API allows to integrate territory management as a component into third-party applications such as CRM, ERP… It allows to inject data (parameters, items, sites, indicators…) and to export the result.

Geoconcept Territory Manager API full documentation [https://mygeoconcept.com/doc/gtmapi/docs/en/secto-api-book/index.html]

Route calculation

Basic principles

This web service calculates an itinerary between two points and returns a full route sheet. It is possible to add optional intermediate steps.

Route calculation example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/route/v7.json",
        "type":"GET",
        "params": {
                "origin": {
                        "sample":"5.3655,43.3293"
                },
                "destination": {
                        "sample":"5.3940,43.2673"
                },
                "waypoints": {
                        "sample":""
                },
                "exclusions": {
                        "sample":"ZFE"
                }
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
origin	Coordinates of the start point (longitude, latitude).	no	

parameter	description	optional	default
	The longitude and latitude coordinates are separated by the , character.		
destination	Coordinates of the end point (longitude,latitude). The longitude and latitude coordinates are separated by the , character	no	
waypoints	List of the coordinates of the route stops. The longitude and latitude coordinates are separated by the "," character. The pairs of coordinates are separated from each other by the ";" character.	yes	
method	Method to compute the route between the start point and the resources: - distance: shortest route - time: fastest route - flying: as the crow flies	yes	time
format	- standard: result = route summary / bounds / geometry in wkt format / simplified geometry in wkt format / list of concatenated segments (without geometries) - extended: result = route summary / bounds / simplified geometry in wkt format / list of non-concatenated segments (with geometries) - summary: result = route summary - geometry: result = route summary / bounds geometry in wkt format - simplifiedgeometry: result = route summary / bounds / simplified geometry in wkt format - geometries: result = route summary / geometry in wkt format - brief: result = route summary / List of segments with duration and distance - standardext: result = route summary / bounds / list of concatenated segments with duration and distance and geometry - node: result = route summary / Id of snapped nodes - compressedgeometry: result = route summary / bounds / compressed geometry with polyline encoding - compressedsimplifiedgeometry: result = route summary / bounds / simplified compressed geometry with polyline encoding - completegeometry: result = route summary / bounds / geometry in wkt format / concatenated segments list with geometries in wkt format - compresscompletegeometry: result = route summary / bounds / compressed geometry with polyline encoding / concatenated segments list with geometries with polyline encoding - tollcost: result = route summary / amount tolls cost. The amount is calculated according to the type of vehicle defined in *configName* as well as the date / time of departure of the *startDateTime* . - custom: declare list of format items in *customFormat* parameter.	yes	standard
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	
tolerance	Tolerance distance (in meters) for the geometry simplification.	yes	
originDateTime	Origin date and time (format ISO8601: local time) example: 2014-01-21T09:00:00.000+0100 (or 2014-01-21T09:00:00.000%2B01:00) for a start on 21 January 2014, at 9.00am in Paris	yes	
destinationDateTime	Destination date and time (format ISO8601: local time) example: 2014-01-21T09:00:00.000+0100 (or 2014-01-21T09:00:00.000%2B01:00) for a departure on 21 January 2014, at 9.00am in Paris	yes	
exclusions	List of restriction rules to use, separated by the ; character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic)	yes	

parameter	description	optional	default
	- ZFE (Zone &#65533; faible &#65533;mission, France-only) - … - Detailled list available on the FAQ bellow.		
timeLine	List of intermediate durations to calculate positions along the route trajectory. Expressed in seconds, separated by the *;* character. The returned position corresponds to the previous node, on the route, connectable to the network. For example, on a highway, the position returned is the last exit or service area before the requested position.	yes	
snapMethod	Snap to graph method - standard: to the nearest connectable road segment - extended: via restricted road sections (pedestrian routes…) - nearest: to the nearest road section only - unrestricted: without any restriction rules - nodes: snap directly to nodes supplied by the originNode originNode, destinationNode and waypointNodes parameters or, if these first ones are not filled, snap directly to the nearest nodes origin, destination and waypoint parameters.	yes	standard
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` Care: WKT geometries must be closed.	yes	
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - pedestrian (not available for Japan) - taxi - truck	yes	
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000" - fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45"	yes	

parameter	description	optional	default
	- cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"		
formatItems	List of format items to use, if route *format* is set to custom - ROUTE_DISTANCE - ROUTE_DURATION - ROUTE_DISTANCE_FORMATTED - ROUTE_DURATION_FORMATTED - ROUTE_BOUNDS - SRS - ROUTE_CARBON_FOOTPRINT - START_FINISH_DATETIME - SUBROUTE_DISTANCE - SUBROUTE_DURATION - SUBROUTE_DISTANCE_FORMATTED - SUBROUTE_DURATION_FORMATTED - SEGMENT_DISTANCE - SEGMENT_DURATION - SEGMENT_DISTANCE_FORMATTED - SEGMENT_DURATION_FORMATTED - SEGMENT_NAME - SEGMENT_NAVIGATIONINSTRUCTION - SEGMENT_POINTSLIST - SEGMENT_FIELDSVALUES - SEGMENT_UNCONSOLIDATED - ROUTE_GEOMETRY_WKT - ROUTE_SIMPLIFIEDGEOMETRY_WKT - ROUTE_GEOMETRY_COMPRESSED - ROUTE_SIMPLIFIEDGEOMETRY_COMPRESSED - SUBROUTE_GEOMETRY_WKT - SUBROUTE_GEOMETRY_COMPRESSED - TIMELINE - POINTS_GRAPHNODES - ROUTE_TOLLCOST - COMPUTATION_TIME	oui	
fields	list of fields to retrieve value for each leg (for standard and extended *formats*) - Iso country code - Country - Postcode left - Postcode right - City name left - City name right	oui	

parameter	description	optional	default
	- Ramp - Urban - Tunnel - Toll - Bicycles - Automobiles - Frontage - Bridge		

Output

Route (routeResult)

parameter	type	min/max	description
distance	string	0/1	Total route distance, formatted: - xx.xx Km - xx m (if the distance is less than 1 km)
duration	string	0/1	Total duration of the route, formatted: - HH:mm:ss (HH=hours, mm=minutes, ss=seconds)
distanceMeters	double	0/1	Total route distance in meters.
durationSeconds	double	0/1	Total duration of the route in seconds.
bounds	double	0/1	Bounds of the result route geometry.
wktGeometry	string	0/1	Geometry of the route in WKT format
wktSimplified	string	0/1	Simplified geometry in WKT format.
legs	Array of SubRouteV7	0/ unlimited	List of route segments.
startDateTime	string	0/1	Start date and time (format: ISO8601 without any area code: local time) Example: 2014-01-21T09:00:00.000+01:00 for a departure on 21 January 2014, at 9.00am in Paris
finishDateTime	string	0/1	Arrival date and time (format: ISO8601 without any area code: local time) Example: 2014-01-21T09:00:00.000+01:00 for an arrival time on 21 January 2014, at 9.00am in Paris
timeLine	Array of TimeLineItemV7	0/ unlimited	List of calculated positions.
srs	string	0/1	projection passed as input (EPSG code such as epsg:4326 or wgs84)
originNode	string	0/1	Start node identifier (entered if format=NODE).
destinationNode	string	0/1	Arrival node identifier (entered if format=NODE).
waypointNodes	waypointNodes array (string)	0/ unlimited	Step nodes identifiers (entered if format=node).
carbonFootprint	double	0/1	Carbon footprint (CO2 emissions in kilograms).
energyConsumption	double	0/1	Energy consumption (Energy consumed in Litres or kWh for electrical vehicles).
tollCost	Array of tollCost	0/ unlimited	Toll cost informations (entered if format=TOLLCOST).

parameter	type	min/max	description
fieldsNames	Array of string	0/ unlimited	value of fields.

Itinerary portion (SubRouteV7)

parameter	type	min/max	description
distance	string	0/1	Total route distance, formatted: - xx.xx Km - xx m (if the distance is less than 1 km)
duration	string	0/1	Total duration of the route, formatted: - HH:mm:ss (HH=hours, mm=minutes, ss=seconds)
distanceMeters	double	0/1	Distance covered by the route segment in meters.
durationSeconds	double	0/1	Duration of the route segment in seconds.
steps	Array of SegmentV7	0/ unlimited	List of component segments in the route segments

Route segment (SegmentV7)

parameter	type	min/max	description
distance	string	0/1	Total route distance, formatted: - xx.xx Km - xx m (if the distance is less than 1 km)
duration	string	0/1	Total duration of the route, formatted: - HH:mm:ss (HH=hours, mm=minutes, ss=seconds)
distanceMeters	double	0/1	Distance covered by the route segment in meters.
durationSeconds	double	0/1	Duration of the route segment in seconds.
navigationInstruction	string	0/1	Navigation instruction code: - F: Straight on - FR: Turn slightly to the right - FL: Turn slightly to the left - R: Turn right - L: Turn left - BR: Turn hard right - BL: Turn hard left - B: U-turn - round_about_entry: Enter roundabout - round_about_exit: Exit from roundabout
name	string	0/1	Segment street name.
points	string	0/ unlimited	List of coordinates separated by the , character
fieldsNames	Array of string	0/ unlimited	value of fields.

Route position (TimeLineItemV7)

parameter	type	min/max	description
durationSeconds	double	0/1	Route duration in seconds up to this position.
message	string	0/1	Error message for this position.

parameter	type	min/max	description
status	string	0/1	Status of this position.
distanceMeters	double	0/1	Distance of the route in meters up to this position.
location	string	0/1	Coordinates of the position.

Toll Cost (tollCost)

parameter	type	min/max	description
amount	double	0/1	Total cost of tolls .
currency	string	0/1	Currency.
costsByCountry	Array of costsByCountry	0/ unlimited	List of toll cost by countries.

Cost by countries (costsByCountry)

parameter	type	min/max	description
amount	double	0/1	Cost of tolls.
country	string	0/1	Country code (3-letter ISO country codes).

Possible returns

Case of a route found (routeResult/status is OK)


```
{
    "status": "OK",
    "distance": "5.23 Km",
    "duration": "0:22:24",
    "distanceMeters": 5229.86,
    "durationSeconds": 1344.15,
    "bounds": "-0.167612,51.48793;-0.125122,51.50068",
    "wktGeometry": "LINESTRING (-0.125122 51.494445, -0.125159 51.494452, -0.12519 51.49439, -0.12517
 51.49399, [...])",
    "wktSimplifiedGeometry": "LINESTRING (-0.125122 51.494445, -0.125159 51.494452, -0.12519 51.49439,
 -0.12517 51.49399, [...])",
    "carbonFootprint": 2.493,
    "energyConsumption": 0.266,
    "computationTime": 3669.643,
    "legs": [
        {
            "distance": "3.37 Km",
            "duration": "0:14:26",
            "distanceMeters": 3374.55,
            "durationSeconds": 866.17,
            "steps": [
                {
                    "distance": "54 m",
                    "duration": "0:00:15",
                    "distanceMeters": 54.45,
                    "durationSeconds": 15.72,
                    "name": "MILLBANK"
                },
                [...]
            ]
        },
        {
```

```
            "distance": "1.86 Km",
            "duration": "0:07:57",
            "distanceMeters": 1855.31,
            "durationSeconds": 477.98,
            "steps": [
                {
                    "distance": "11 m",
                    "duration": "0:00:03",
                    "distanceMeters": 11.07,
                    "durationSeconds": 3.16,
                    "name": "BASIL STREET"
                },
               [...]
            ]
        }
    ]
}
```


Case of a forgotten specification of a start or finish point (routeResult/status is ERROR)


```
{
    "message": "Origin, destination and waypoints should be represented by a couple of coordinates",
    "status": "ERROR",
    "legs": [
    ]
}
```


Case of a faulty format assigned to the start or finish point or to route stops (routeResult/status is ERROR)


```
{
    "message": "Origin, destination and waypoints should be represented by a couple of coordinates",
    "status": "ERROR",
    "legs": [
    ]
}
```


Case of a snap-to-graph error (serviceResult/status is ERROR)


```
{
    "message": "ServiceException: Error in route computation|Error in smartrouting|Failed to execute
 calculateRoute|com.geoconcept.smartrouting.SmartRoutingNativeException: failed to connect waypoint
 { -100.125122, 51.494445, 0.000000 } srs=epsg:4326|failed to connect waypoint { -100.125122, 51.494445,
 0.000000 } srs=epsg:4326",
    "status": "ERROR"
}
```


FAQ

1. Is it possible to give priority to either journey time or journey distance?

   Yes, by changing the method: distance or time.

2. Is it important what order the points are in?

   The points, whether they are start points, finish points, or way points are read and used in that order. The first point declared is considered as a start point, the second as an arrival point, and the others as way points. The order of the points is therefore very important.

3. How perform a route calculation without including any road tolls?

   Place the exclusion *Toll* in `exclusions` .

4. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255
emergency vehicle						
pedestrian						
taxi						
truck	4	19000	39000	387	1850	255

5. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on `exclusions` toll road sections are not used.
BoatFerry	if added on `exclusions` ferry itineraries (shipping) over water are not used.
RailFerry	if added on `exclusions` ferry (or train) itineraries (over water and land respectively) are not used.
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.

Exclusion	Description
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

6. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …
   - Pollutant: goods harmful for water
   - Toxic: organic, poison, radioactive, corrosive, …
   - Dangerous: tunnel categories and others restrictions

7. What are Speed Patterns? How are they used?

   With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

   - standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
   - night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
   - busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
   - peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)

- default *default* corresponds to average speeds over a whole day

To use these, you will need to pass to parameter, when calling the web service, the following parameter `computeOptions` with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
Example:

```
&computeOptions=speedPattern:fast-speed&configName=car
```

8. How do you use heavy goods vehicle attributes?

   You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter `computeOptions` with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
   Example for a journey with a vehicle with a height of 4.5m:

   ```
   &computeOptions=height:450
   ```

9. How is the carbon footprint for a journey calculated?

   Calculation of carbon footprint takes into account vehicle type, fuel type, and vehicle speed, road section by road section: 10 kms in a town context does not have the same carbon footprint as 10km in the countryside. It is displayed in the response when format is equal to *standard* (default) or *extended*.
   The carbon footprint of the route is calculated automatically. You can overdefine the default values when the call to the web service is made using `computeOptions` with options *fuelType*, *averageConsumption* and/or *customAverageCO2UnitEmission*.
   Available values are fuelType with, as possible values:

   - NoFuel
   - Diesel
   - UnleadedFuel
   - LGP
   - CustomFuelType (enables personalization of a value in kgs of CO2 per litre to specify in customAverageCO2UnitEmission)

   averageConsumption allows you to indicate the average consumption for the vehicle for 100 kilometers.

10. What is the algorithm used for compressedgeometry and compressedsimplifiedgeometry formats?

    The utilization of the *encoded polyline* compression algorithm (Encoded Polyline Algorithm Format [https://developers.google.com/maps/documentation/utilities/polylinealgorithm]) enabling storage of coordinates in a single ASCII character string, thereby significantly reducing overall data volumes.

Route calculation (batch)

Basic principles

This web service calculates a set of itineraries between two points and returns a for each a full route sheet. It is possible to add optional intermediate steps.

Route calculation example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/route/batch.json",
        "method":"POST",
        "body": {
      "routeRequests": [
       {
          "origin":{
             "x":2.348,
             "y":48.853783
          },
          "destination":{
             "x":2.349,
             "y":48.854
          },
          "srs":"epsg:4326",
          "configName":"car",
          "format":"summary"
       },
       {
          "origin":{
             "x":-1.524361036,
             "y":47.23876778
          },
          "destination":{
             "x":-1.525361036,
             "y":47.23776778
          },
          "srs":"epsg:4326",
          "configName":"car",
          "format":"summary"
       }
      ]
    }
}
```

Parameters / properties

Input

parameter	type	optional	description
routeRequests	array of RouteRequestV5	No	Table of routes as input.
timeOut	string	Yes	Time out value for computations (in milliseconds).

Routes as input (RouteRequestV5)

Detailed on Route calculation web service.

Output

parameter	type	min/max	description
result	array of RouteResultV5	0/ unlimited	The routes results

Routes results (RouteResultV5)

Detailed on Route calculation web service.

Possible returns

Case of a route found (routeResult/status is OK)


```
{
    "message": null,
    "status": "OK",
    "results": [
        {
            "message": null,
            "status": "OK",
            "distance": "1.66 Km",
            "duration": "0:07:56",
            "distanceMeters": 1658.9,
            "durationSeconds": 476.69,
            "bounds": null,
            "wktGeometry": null,
            "wktSimplifiedGeometry": null,
            "compressedGeometry": null,
            "compressedSimplifiedGeometry": null,
            "legs": [
            ],
            "startDateTime": null,
            "finishDateTime": null,
            "srs": "epsg:4326",
            "originNode": null,
            "waypointNodes": null,
            "destinationNode": null,
            "carbonFootprint": null
        },
        {
            "message": null,
            "status": "OK",
            "distance": "159 m",
            "duration": "0:00:47",
            "distanceMeters": 158.76,
            "durationSeconds": 47.66,
            "bounds": null,
            "wktGeometry": null,
            "wktSimplifiedGeometry": null,
            "compressedGeometry": null,
            "compressedSimplifiedGeometry": null,
            "legs": [
            ],
            "startDateTime": null,
            "finishDateTime": null,
            "srs": "epsg:4326",
            "originNode": null,
            "waypointNodes": null,
            "destinationNode": null,
            "carbonFootprint": null
        }
    ]
```

```
}
```

See Route calculation web service.

FAQ

See Route calculation web service.

Estimated Time of Arrival (ETA)

Basic principles

This web service gives time to arrive to destination accounting real-time traffic.

Route calculation example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/eta.json",
        "type":"GET",
        "params": {
                "origin": {
                        "sample":"-1.351448,47.446923"
                },
                "destination": {
                        "sample":"-1.34529,47.4479931"
                },
                "configName": {
                        "sample":"car"
                }
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
configName	Name of the configuration to use: - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - taxi - truck	yes	car
origin	Coordinates of the start point (longitude, latitude). The longitude and latitude coordinates are separated by the , character.	no	
destination	Coordinates of the end point (longitude,latitude). The longitude and latitude coordinates are separated by the , character	no	
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	epsg:4326

Output

Route (routeResult)

parameter	type	min/max	description
time	double	0/1	Remaining journey time in seconds.
distance	double	0/1	Total route distance in meters.

Possible returns

Case of a route found (etaResult/status is OK)

```
{
   "time": 110,
   "distance": 630
}
```

Case of a forgotten specification of a start or finish point (etaResult/status is ERROR)

```
{
   "message": "ServiceException: invalid coordinates ''",
   "status": "ERROR"
}
```

Case of a non existing graphName (etaResult/status is ERROR)

```
{
   "message": "ServiceException: Cannot retrieve Graph Definition for configuration xxx",
   "status": "ERROR"
}
```

Waypoint sequence

Basic principles

This web service allows you to define optimum routes while minimising travel times and distances. The ordering of points to visit is optimized on the basis of geographic and operational constraints (time slots, duration of the visit…).

Optimization example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/optim/route/v4.json",
        "method":"POST",
        "body":
    {
      "origin": {
        "x": 2.33906,
        "y": 48.84467,
        "id": "1",
        "duration": 600000,
        "timeWindows": [
          {
            "start": 0,
            "end": 3600000
          }
        ]
      },
```

```
      "destination": {
        "x": 2.37125,
        "y": 48.82769,
        "id": "5",
        "duration": 600000
      },
      "steps": [
        {
          "x": 2.32792,
          "y": 48.81033,
          "id": "2",
          "duration": 600000,
          "timeWindows": [
            {
              "start": 360000,
              "end": 7200000
            }
          ]
        },
        {
          "x": 2.35373,
          "y": 48.83176,
          "id": "3",
          "duration": 600000
        },
        {
          "x": 2.32453,
          "y": 48.80319,
          "id": "4",
          "duration": 600000
        }
      ],
      "method": "TIME",
      "matrixProvider": "SMARTROUTING",
      "srs": "wgs84"
    }
}
```


Parameters / properties

Input

parameter	description	optional	default
origin (optimRouteStep)	origin point (id,x,y)	yes	
destination (optimRouteStep)	destination point (id,x,y)	yes	
steps (Array of optimRouteStep)	points to visit (id,x,y;id,x,y;…) Several slots for visiting times in milliseconds can be specified for each point: (id,x,y,start, finish)	no	
method	shortest (distance) or fastest (time) route	yes	time
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle	yes	

parameter	description	optional	default
	- pedestrian (not available for Japan) - taxi - truck		
snapMethod	Snap to graph method - standard: to the nearest connectable road segment - extended: via restricted road sections (pedestrian routes…) - nearest: to the nearest road section only - unrestricted: without any restriction rules - nodes: snap directly to nodes supplied by the originNode originNode, destinationNode and waypointNodes parameters or, if these first ones are not filled, snap directly to the nearest nodes origin, destination and waypoint parameters.	yes	standard
exclusions	List of restriction rules to use, separated by the *;* character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone à faible émission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
startDateTime	Start date and time (format ISO8601: local time) example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a start on 21 January 2014, at 9.00am in Paris	yes	
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` Care: WKT geometries must be closed.	yes	
timeLine	List of intermediate durations to calculate positions along the route trajectory. Expressed in seconds, separated by the *;* character	yes	
snapMethod	Snap to graph method - standard: to the nearest connectable road segment - extended: via restricted road sections (pedestrian routes…) - nearest: to the nearest road section only - unrestricted: without any restriction rules - nodes: snap directly to nodes supplied by the originNode originNode, destinationNode and waypointNodes parameters or, if these first ones are not filled, snap directly to the nearest nodes origin, destination and waypoint parameters.	yes	standard
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000"	yes	

parameter	description	optional	default
	- fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"		
matrixProvider	matrix calculation provider - globe: as the crow flies on the globe (the coordinates must be in long/lat) - euclidean: as the crow flies on the map (the coordinates must be in meters) - smartrouting: calculation on the road network by SmartRouting - provided: matrix parameter	yes	smartrouting
matrix	time/distance matrix (id1,id2,time,distance;…)	yes	
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	wgs84
directSpeed	Default speed during utilisation of globe or euclidean matrices	yes	50

Steps (optimRouteStep)

parameter	type	min/ max	description	optional	default
x	double	1/1	Longitude of the step	no	
y	double	1/1	Latitude of the step	no	
id	string	0/1	ID of the step	no	
duration	long	1/1	duration of a visit, in milliseconds	yes	0
timeWindows	Array of optimTimeWindow	1/1	Time windows	yes	

Time windows (optimTimeWindow)

parameter	type	min/ max	description	optional	default
start	long	1/1	Start of the time window, in milliseconds	no	
end	long	1/1	Time window end, in milliseconds	no	

Output

parameter	type	min/max	description
steps/step	Array of optimRouteStepOut	0/ unlimited	List of the ordered steps
distanceMeters	long	1/1	Total journey distance in meters
durationSeconds	long	1/1	Total journey time in seconds
unreachableSteps	array (OptimUnreachableStep)	0/ unlimited	List of steps not reachable

Steps (optimRouteStepOut)

parameter	type	min/max	description
id	string	0/1	ID of the step
x	double	1/1	Longitude of the step
y	double	1/1	Latitude of the step
duration	long	1/1	Duration of the step
effectiveStart	long	1/1	Effective start time of the step, in milliseconds
driveDistanceBefore	int	1/1	Drive time before the step, in meters
driveDistanceAfter	int	1/1	Drive distance after the step, in meters
driveTimeBefore	long	1/1	Drive time before the step, in milliseconds
driveTimeAfter	long	1/1	Drive time after the step, in milliseconds
timeWindows	Array of optimTimeWindowOut	1/1	Time windows

Time windows (optimTimeWindowOut)

parameter	type	min/max	description
start	long	1/1	Time window start, in milliseconds
end	long	1/1	Time window end, in milliseconds

List of steps not reachable (OptimUnreachableStep)

paramètre	type	min/max	description
id	string	1/1	ID of the step
x	double	1/1	Longitude of the step
y	double	1/1	Latitude of the step

Possible returns

Case of an optimization applied successfully


```
{
   "steps": [
      {
        "x": 2.33906,
        "y": 48.84467,
        "id": "1",
        "duration": 600000,
        "effectiveStart": 0,
        "driveDistanceBefore": 0,
        "driveDistanceAfter": 5057,
```

```
        "driveTimeBefore": 0,
        "driveTimeAfter": 737740,
        "timeWindows": [
          {
            "start": 0,
            "end": 3600000
          },
          [...]
        ]
      },
      [...]
    ],
    "distanceMeters": 13502,
    "durationSeconds": 2015
}
```

FAQ

1. Can you force a start and/or arrival step in the calculation of a route?

   Yes, you just have to use the origin and / or destination parameters respectively. The journey between the steps route stops is optimized taking into account any operational constraints, and without taking into account any logistical constraint (departure from home, arrival in a depot, ..;).

2. How can you specify, for each step, a visit duration and a visit time window?

   using the duration parameter (to indicate the duration set aside for a visit) and the timeWindowStart and timeWindowEnd parameters (to specify the start and finish time for a visit).

3. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on **`exclusions`** toll road sections are not used.
BoatFerry	if added on **`exclusions`** ferry itineraries (shipping) over water are not used.
RailFerry	if added on **`exclusions`** ferry (or train) itineraries (over water and land respectively) are not used.
Tunnel	if added on **`exclusions`** tunnels are not used.
Bridge	if added on **`exclusions`** bridges are not used.
Frontage	if added on **`exclusions`** Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on **`exclusions`** tarmac road sections are not used.
Private	if added on **`exclusions`** road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on **`exclusions`** road sections unauthorised for pedestrian use are not used. The **`configName`** parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on **`exclusions`** road sections with restricted automobile access are not used. The **`configName`** parameter with a value of car uses this exclusion.

Exclusion	Description
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

4. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …
   - Pollutant: goods harmful for water
   - Toxic: organic, poison, radioactive, corrosive, …
   - Dangerous: tunnel categories and others restrictions

Isochrones/Isodistances

Basic principles

This web service calculates an isochrone/isodistance from a point and returns the geometry of the zone calculated.

Isochrones example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/isochrone/v5.json",
        "type":"GET",
```

```
        "params": {
                "location": {
                        "sample":"2.321134,48.796575"
                },
                "distance": {
                        "sample":"1000"
                },
                "time": {
                        "sample":"60"
                },
                "method": {
                        "sample":"time"
                },
                "exclusions": {
                        "sample":""
                }
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
id	Isochrone identifier	yes	
location	Start (or arrival point, if the **`reverse`** parameter is set to True). The coordinates are separated by , characters.	no	
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	epsg:4326
exclusions	List of restriction rules to use, separated by the ; character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone ♦ faible ♦mission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
method	"time" for isochrone or "distance" for isodistance	no	time
time	Maximum access time, in seconds	yes	
distance	Maximum access distance, in meters	yes	
reverse	if *true*, the **`location`** is considered as an arrival point	yes	false
smoothing	Smoothing	yes	false
resolution	Resolution (int) of the resulting isochrone, in meters. Do not set this parameter to stay in automatic resolution mode	yes	auto
holes	Display holes in the result zone (the geometry returned is more voluminous when this parameter is set to *true*)	yes	false
startDateTime	Start date and time (format ISO8601: local time) Example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a departure on 21 Januray 2014, at 9.00am in Paris	yes	
snapMethod	Method for snapping to the graph - standard: at the nearest connectable road section - extended: via restricted road sections (pedestrian walkways….) - nearest: only to the nearest road section - unrestricted: without any restriction rules	yes	standard

parameter	description	optional	default
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` NOTE: WKT geometries must be closed.	yes	
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - pedestrian (not available for Japan) - taxi - truck	yes	
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000" - fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"	yes	

Output

Isochrone (isochroneResult)

parameter	type	min/max	description
id	string	0/1	Isochrone identifier
location	string	0/1	Start (or Finish if the `reverse` is set to true).
srs	string	0/1	projection
time	string	0/1	Maximum access time, in seconds
distance	string	0/1	Maximum access distance, in meters
wktGeometry	string	0/1	Geometry of the isochrone, in wkt format

Possible returns

Case of an isochrone/isodistance that has been found (status is OK)


```
{
  "message": null,
  "status": "OK",
  "id": null,
  "location": "2.321134,48.796575",
  "srs": "epsg:4326",
  "time": "60",
  "distance": null,
  "wktGeometry": "POLYGON ((2.32113 48.795216, 2.32113 48.795996, 2.319976 48.795996, 2.319976 48.795606,
 2.3194 48.795606, 2.3194 48.795216, 2.318823 48.795216, 2.318823 48.794826, 2.318246 48.794826, 2.318246
 48.795606, 2.317092 48.795606, 2.317092 48.795996, 2.318246 48.795996, 2.318246 48.796386, 2.320553
 48.796386, 2.320553 48.796776, 2.3194 48.796776, 2.3194 48.797166, 2.319976 48.797166, 2.319976 48.797556,
 2.320553 48.797556, 2.320553 48.797166, 2.32113 48.797166, 2.32113 48.796386, 2.321707 48.796386, 2.321707
 48.797556, 2.322284 48.797556, 2.322284 48.797946, 2.322861 48.797946, 2.322861 48.797166, 2.323438
 48.797166, 2.323438 48.796776, 2.322284 48.796776, 2.322284 48.796386, 2.324015 48.796386, 2.324015
 48.795606, 2.323438 48.795606, 2.323438 48.795996, 2.322284 48.795996, 2.322284 48.795606, 2.321707
 48.795606, 2.321707 48.794436, 2.32113 48.794436, 2.32113 48.795216))"
}
```


Case whereby method = distance and distance is not supplied (status is ERROR)


```
{
  message: "distance parameter must not be null or 0 !",
  status: "ERROR"
}
```


Case whereby method + time are not supplied (status is ERROR)


```
{
  "message": "time parameter must not be null or 0 !",
  "status": "ERROR"
}
```


Case whereby the location tag is missing (status is ERROR)


```
{
message: "Location must be not null",
status: "ERROR"
}
```


Case whereby the start point is supplied with errors (status is ERROR)

```
{{
  "message": "Location point must have 2 components separated with a ,",
  "status": "ERROR"
}
```

Case of a snap-to-graph error (status is ERROR)

```
An error occurred : (500) error
ServiceException: Error in isochron computation|Error in smartrouting|Failed to execute
 calculateConcentricReachableAreas|com.geoconcept.smartrouting.SmartRoutingNativeException: failed to
 connect isochrone orign { -42.321134, 48.796575, 0.000000 }|failed to connect isochrone orign { -42.321134,
 48.796575, 0.000000 }
```

FAQ

1. Is it possible to give priority to either journey time or distance?

   Yes, by changing the method to "time" **`method`** for isochrone, or "distance" for isodistance

2. How do you perform an isochrone/isodistance calculation without any tolls?

   Place the exclusion *Toll* in **`exclusions`** .

3. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255
emergency vehicle						
pedestrian						
taxi						
truck	4	19000	39000	387	1850	255

4. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on **`exclusions`** toll road sections are not used.
BoatFerry	if added on **`exclusions`** ferry itineraries (shipping) over water are not used.
RailFerry	if added on **`exclusions`** ferry (or train) itineraries (over water and land respectively) are not used.

Exclusion	Description
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

5. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …

- Pollutant: goods harmful for water
- Toxic: organic, poison, radioactive, corrosive, …
- Dangerous: tunnel categories and others restrictions

6. What are Speed Patterns? How are they used?

   With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

   - standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
   - night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
   - busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
   - peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)
   - default *default* corresponds to average speeds over a whole day

   To use these, you will need to pass to parameter, when calling the web service, the following parameter **`computeOptions`** with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
   Example:

   ```
   &computeOptions=speedPattern:fast-speed&configName=car
   ```

7. How do you use heavy goods vehicle attributes?

   You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter **`computeOptions`** with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
   Example for a journey with a vehicle with a height of 4.5m:

   ```
   &computeOptions=height:450
   ```

Multi Isochrones/Isodistances

Basic principles

This web service simultaneously calculates several isochrones/isodistances from points and allow/avoid overlapping between geometries. It returns the geometries of the zones calculated.

Isochrones example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/isochrone/multi/v2.json",
        "method":"POST",
        "body":
    {
    "locations" : [ {
```

```
        "x":2.321134,
        "y":48.796575
      },{
        "x":2.324340,
        "y":48.803376
      } ],
      "method":"time",
      "time" : 180,
      "overlapping" : false
      }
  }
```

Parameters / properties

Input

parameter	description	optional	default
location (array of Point)	Start (or arrival point, if the **`reverse`** parameter is set to True).	no	
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	epsg:4326
exclusions	List of restriction rules to use, separated by the ; character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone à faible émission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
method	"time" for isochrone or "distance" for isodistance	no	time
time	Maximum access time, in seconds	yes	
distance	Maximum access distance, in meters	yes	
reverse	if *true*, the **`location`** is considered as an arrival point	yes	false
smoothing	Smoothing the geometries if the **`overlapping`** parameter is set to True	yes	false
resolution	Resolution (int) of the resulting isochrone, in meters. Do not set this parameter to stay in automatic resolution mode	yes	auto
holes	Display holes in the result zone (the geometry returned is more voluminous when this parameter is set to *true*)	yes	false
overlapping	To allow/avoid overlapping between geometries	yes	true
startDateTime	Start date and time (format ISO8601: local time) Example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a departure on 21 Januray 2014, at 9.00am in Paris	yes	
snapMethod	Method for snapping to the graph - standard: at the nearest connectable road section - extended: via restricted road sections (pedestrian walkways….) - nearest: only to the nearest road section - unrestricted: without any restriction rules	yes	standard
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))`	yes	

parameter	description	optional	default
	NOTE: WKT geometries must be closed.		
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - pedestrian (not available for Japan) - taxi - truck	yes	
computeOptions	Selection of variables to use in calculations, using semi-colons ; as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000" - fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"	yes	

Geolocalized point (Point)

parameter	description	optional	default
id	Point identifier	No	
x	First coordinate or longitude	No	
y	Second coordonnée or latitude	No	
nodeId	Point node id on the graph	No	

Output

parameter	type	min/max	description
Isochrones	array of IsochroneItemResult	0/ unlimited	List of calculated areas.

List of calculated isochrones (MultiIsochronesResult)

parameter	type	min/max	description
id	string	0/1	Isochrone identifier
time	string	0/1	Maximum access time, in seconds
distance	string	0/1	Maximum access distance, in meters
wktGeometry	string	0/1	Geometry of the isochrone, in wkt format
location	string	0/1	Start (or Finish if the `reverse` is set to true).
srs	string	0/1	projection

Possible returns

Case of an mulisochrone/isodistance that has been found (status is OK)

```
{
    "status": "OK",
    "isochrones": [
        {
            "location": "2.321134,48.796575",
            "time": "180",
            "wktGeometry": "POLYGON ((2.326875 48.79561, 2.32712 48.79606, [...]))"
        },
        {
            "location": "2.32434,48.803376",
            "time": "180",
            "wktGeometry": "POLYGON ((2.32947 48.80445, 2.329245 48.80438, [...]))"
        }
    ]
}
```

Case whereby method = distance and distance is not supplied (status is ERROR)

```
{
  message: "distance parameter must not be null or 0 !",
  status: "ERROR"
}
```

Case whereby method + time are not supplied (status is ERROR)

```
{
  "message": "time parameter must not be null or 0 !",
  "status": "ERROR"
}
```

Case whereby the location tag is missing (status is ERROR)

```
{
message: "Location must be not null",
status: "ERROR"
}
```

Case whereby the start point is supplied with errors (status is ERROR)

```
{{
  "message": "Location point must have 2 components separated with a ,",
  "status": "ERROR"
}
```

Case of a snap-to-graph error (status is ERROR)

```
An error occurred : (500) error
ServiceException: Error in isochron computation|Error in smartrouting|Failed to execute
 calculateConcentricReachableAreas|com.geoconcept.smartrouting.SmartRoutingNativeException: failed to
 connect isochrone orign { -42.321134, 48.796575, 0.000000 }|failed to connect isochrone orign { -42.321134,
 48.796575, 0.000000 }
```

FAQ

1. Is it possible to give priority to either journey time or distance?

   Yes, by changing the method to "time" `method` for isochrone, or "distance" for isodistance

2. How do you perform an isochrone/isodistance calculation without any tolls?

   Place the exclusion *Toll* in `exclusions` .

3. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255
emergency vehicle						
pedestrian						
taxi						
truck	4	19000	39000	387	1850	255

4. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on `exclusions` toll road sections are not used.
BoatFerry	if added on `exclusions` ferry itineraries (shipping) over water are not used.
RailFerry	if added on `exclusions` ferry (or train) itineraries (over water and land respectively) are not used.

Exclusion	Description
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

5. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …

- Pollutant: goods harmful for water
- Toxic: organic, poison, radioactive, corrosive, …
- Dangerous: tunnel categories and others restrictions

6. What are Speed Patterns? How are they used?

   With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

   - standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
   - night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
   - busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
   - peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)
   - default *default* corresponds to average speeds over a whole day

   To use these, you will need to pass to parameter, when calling the web service, the following parameter **`computeOptions`** with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
   Example:

   ```
   &computeOptions=speedPattern:fast-speed&configName=car
   ```

7. How do you use heavy goods vehicle attributes?

   You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter **`computeOptions`** with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
   Example for a journey with a vehicle with a height of 4.5m:

   ```
   &computeOptions=height:450
   ```

Matrix calculation

> This web service is deprecated use compact matrix calculation web service instead.

Basic principles

This web service calculates a route matrix for a series of points and returns a distance matrix. The distance and time computations are based on the actual road network.

Matrix example

```
{
   "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/matrix/v4.json",
```

```
    "method": "POST",
    "body":
    {
       "origins": [
          {
             "x": 6.961438,
             "y": 50.932713
          },
          {
             "x": 6.942641,
             "y": 50.92259
          }
       ],
       "destinations": [
          {
             "x": 6.973438,
             "y": 50.939732
          },
          {
             "x": 7.106549,
             "y": 50.879707
          }
       ],
       "srs": "WGS84",
       "method": "time",
    }
 }
```

Parameters / properties

Input

parameter	description	optional	default
srs	projection (ESPG code such as epsg:4326 or wgs84)	yes	
origins	List of coordinates of points of origin. The longitude and latitude coordinates are separated by the "," character. The pairs of coordinates are separated by the ";" character.	yes	
destinations	List of coordinates of destination points. The latitude and longitude coooordinates are separated by the , character. The pairs of coordinates are separated by the ";" character.	yes	
method	shortest (distance) or fastest (time) route	yes	time
exclusions	List of restriction rules to use, separated by the ; character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone &#65533; faible &#65533;mission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
startDateTime	Start date and time (format ISO8601: local time). Example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a start on 21 January 2014, at 9.00am in Paris	yes	
snapMethod	Snap-to-graph method - standard: to the nearest connectable road section - extended: via restricted road sections (pedestrian walkways…) - nearest: to the nearest road section only - unrestricted: without any restriction rules	yes	standard

parameter	description	optional	default
	- nodes: snap directly to nodes supplied by the originNode originNode, destinationNode and waypointNodes parameters or, if these first ones are not filled, snap directly to the nearest nodes origin, destination and waypoint parameters.		
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84 : `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` Care: WKT geometries must be closed.	yes	
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - pedestrian (not available for Japan) - taxi - truck	yes	
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000" - fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"	yes	

Output

Matrix (matrixResult)

parameter	type	min/max	description
origins	Array of string	0/ unlimited	List of the origin positions. Every position is a string of the longitude and latitude coordinates separated by the "," character.
destinations	Array of string	0/ unlimited	List of the destination positions. Every position is a string of the longitude and latitude coordinates separated by the "," character.
srs	string	0/1	projection passed as input (EPSG code such as epsg:4326 or wgs84)
rows	Array of matrixRow	0/ unlimited	Matrix line

Matrix line (matrixRow)

parameter	type	min/max	description
cells	Array of matrixCell	0/ unlimited	Matrix cell

Matrix cell (matrixCell)

parameter	type	min/max	description
distanceMeters	double	1/1	Matrix cell distance (in meters)
durationSeconds	double	1/1	Matrix cell duration (in seconds)
status	string	0/1	Matrix cell status: - OK: the itinerary has been found for this cell - KO: no itinerary found for this cell

Possible returns

Case of an itinerary that has been found (matrixResultV3/status is OK)


```
{
  "message": null,
  "status": "OK",
  "origins": [
    "6.961438,50.932713",
    "6.942641,50.92259"
  ],
  "destinations": [
    "6.973438,50.939732",
    "7.106549,50.879707"
  ],
  "srs": "epsg:4326",
  "rows": [
    {
      "cells": [
        {
          "distanceMeters": 2119.74,
          "durationSeconds": 332.65,
          "status": "OK"
        },
```

```
        {
          "distanceMeters": 14177.7,
          "durationSeconds": 1014.98,
          "status": "OK"
        }
      ]
    },
    {
      "cells": [
        {
          "distanceMeters": 3702.02,
          "durationSeconds": 559.35,
          "status": "OK"
        },
        {
          "distanceMeters": 15231.24,
          "durationSeconds": 1181.95,
          "status": "OK"
        }
      ]
    }
  ]
}
```


Case of forgotten origin and destination specification (matrixResult/status is ERROR)


```
{
  "message":"Origins and destinations must be not null",
  "status":"ERROR"
}
```


Case of a faulty type (serviceResult/status is ERROR)


```
{
  "message":"NumberFormatException: For input string: \"ghs5\"",
  "status":"ERROR"
}
```


Case of a snap-to-graph error (matrixResult/status is OK) / (cell/status est KO)


```
{
  "message": null,
  "status": "OK",
  "origins": [
    "6.961438,50.932713",
    "50.92259,6.942641"
  ],
  "destinations": [
    "6.973438,50.939732",
    "7.106549,50.879707"
  ],
  "srs": "epsg:4326",
  "rows": [
    {
      "cells": [
        {
          "distanceMeters": 2119.74,
          "durationSeconds": 332.65,
          "status": "OK"
        },
```

```
        {
          "distanceMeters": 14177.7,
          "durationSeconds": 1014.98,
          "status": "OK"
        }
      ]
    },
    {
      "cells": [
        {
          "distanceMeters": -1,
          "durationSeconds": -1,
          "status": "KO"
        },
        {
          "distanceMeters": -1,
          "durationSeconds": -1,
          "status": "KO"
        }
      ]
    }
  ]
}
```


FAQ

1. How can I perform a toll-free route matrix calculation?

   Place the exclusion *Toll* in **`exclusions`** .

2. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255
emergency vehicle						
pedestrian						
taxi						
truck	4	19000	39000	387	1850	255

3. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on **`exclusions`** toll road sections are not used.

Exclusion	Description
BoatFerry	if added on `exclusions` ferry itineraries (shipping) over water are not used.
RailFerry	if added on `exclusions` ferry (or train) itineraries (over water and land respectively) are not used.
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

4. What are the Hazardous material exclusions?

Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

- Flammable: explosives, flammable, gas, combustible, …
- Pollutant: goods harmful for water
- Toxic: organic, poison, radioactive, corrosive, …
- Dangerous: tunnel categories and others restrictions

5. What are Speed Patterns? How are they used?

   With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

   - standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
   - night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
   - busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
   - peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)
   - default *default* corresponds to average speeds over a whole day

   To use these, you will need to pass to parameter, when calling the web service, the following parameter **`computeOptions`** with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
   Example:

   ```
   &computeOptions=speedPattern:fast-speed&configName=car
   ```

6. How do you use heavy goods vehicle attributes?

   You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter **`computeOptions`** with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
   Example for a journey with a vehicle with a height of 4.5m:

   ```
   &computeOptions=height:450
   ```

Compact matrix calculation

Basic principles

This web service calculates a route matrix for a series of points and returns a distance matrix. The distance and time computations are based on the actual road network.

Compact matrix example

```
{
   "url": "https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/compactMatrix/v5.json",
```

```
    "method": "POST",
    "body":
    {
       "origins": [
          {
             "x": 6.961438,
             "y": 50.932713
          },
          {
             "x": 6.942641,
             "y": 50.92259
          }
       ],
       "destinations": [
          {
             "x": 6.973438,
             "y": 50.939732
          },
          {
             "x": 7.106549,
             "y": 50.879707
          }
       ],
       "srs": "WGS84",
       "method": "time",
    }
 }
```

Settings / Properties

Input

parameter	description	optional	default
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	
origins	List of the coordinates for points of origin. Longitude and Latitude coordinates are separated by , characters. The pairs of coordinates are separated by the ";" character.	yes	
destinations	List of coordinates for destination points. Longitude and Latitude coordinates are separated by , characters. The pairs of coordinates are separated by the ";" character.	yes	
method	shortest route (distance) or fastest route (time)	yes	time
exclusions	List of restriction rules to use, separated by the *;* character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone faible mission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
startDateTime	Start date and time (format ISO8601: local time) Example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a departure on 21 January 2014, at 9h in Paris	yes	
snapMethod	Snap to graph method - standard: to the nearest connectable road section - extended: via restricted road sections (pedestrian routes…) - nearest: only to the nearest road section - unrestricted: without any restriction rules	yes	standard

parameter	description	optional	default
	- nodes: Snap directly to nodes supplied by the originNode originNode, destinationNode and waypointNodes parameters or, if these first ones are not filled, snap directly to the nearest nodes origin, destination and waypoint parameters.		
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the projection (srs parameter) requested Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` CARE: WKT geometries must be closed.	yes	
maxTargets	Stops calculating routes on each line when maxTargets routes have already been calculated (results in a partially calculated matrix, but with at least the first column filled)	yes	no limit
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - pedestrian (not available for Japan) - taxi - truck	yes	
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000" - fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" -	yes	

parameter	description	optional	default
	customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"		

Output

Matrix (compactMatrixResult)

parameter	type	min/max	description
distanceMeters	Bi-dimensional Array of double	0/ unlimited	Distance matrix result (in meters)
durationSeconds	Bi-dimensional Array of double	0/ unlimited	Time matrix result (in seconds)

Possible returns

Case of a found itinerary (compactMatrixResult/status is OK)

```
{
  "message": null,
  "status": "OK",
  "distanceMeters": [
    [
      2120,
      14178
    ],
    [
      3702,
      15231
    ]
  ],
  "durationSeconds": [
    [
      333,
      1015
    ],
    [
      559,
      1182
    ]
  ]
}
```

Case of a forgotten origin or destination specification tool (compactMatrixResult/status is ERROR)

```
{
  "message": "Origins must be not empty",
  "status": "ERROR"
}
```

Case of a faulty type (serviceResult/status is ERROR)

```
{
  "message":"NumberFormatException: For input string: \"sdfkl\"",
```

```
  "status":"ERROR"
}
```

Case of a snap to graph error (compactMatrixResult/status is OK) / (cell/status is KO)

```
{
  "message": null,
  "status": "OK",
  "distanceMeters": [
    [
      2120,
      14178
    ],
    [
      -1,
      -1
    ]
  ],
  "durationSeconds": [
    [
      333,
      1015
    ],
    [
      -1,
      -1
    ]
  ]
}
```

FAQ

1. How can I perform a toll-free route matrix calculation?

   Place the exclusion *Toll* in `exclusions` .

2. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255
emergency vehicle						
pedestrian						
taxi						

configName	axles	weightPerAxle	weight	height	length	width
truck	4	19000	39000	387	1850	255

3. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on `exclusions` toll road sections are not used.
BoatFerry	if added on `exclusions` ferry itineraries (shipping) over water are not used.
RailFerry	if added on `exclusions` ferry (or train) itineraries (over water and land respectively) are not used.
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.

Exclusion	Description
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

4. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …
   - Pollutant: goods harmful for water
   - Toxic: organic, poison, radioactive, corrosive, …
   - Dangerous: tunnel categories and others restrictions

5. What are Speed Patterns? How are they used?

   With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

   - standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
   - night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
   - busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
   - peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)
   - default *default* corresponds to average speeds over a whole day

   To use these, you will need to pass to parameter, when calling the web service, the following parameter `computeOptions` with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
   Example:

   ```
   &computeOptions=speedPattern:fast-speed&configName=car
   ```

6. How do you use heavy goods vehicle attributes?

   You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter `computeOptions` with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
   Example for a journey with a vehicle with a height of 4.5m:

   ```
   &computeOptions=height:450
   ```

Search Around

Basic principles

This web service allows you to classify a list of resources positioned around a particular point in terms of distance or time. It takes as input a start point (in this case, the target location) and a list of resources with their geographic locations, plus a priority category of either 1 or 2 (this is an optional parameter). The resources are records saved in a database of the user's choice, depending on their requirements. It is the allotted task of the client to select these resources in the database. The resources are returned sorted firstly on the attribute priority1, and then by time or distance (according to which has been chosen). The priorities are optional. The default value of the priorities is 0. This depends on the configured graph, the name of which has been specified in the Geoconcept Web administration interface.

Search around example

```
{
  "url": "https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/searchAround/v4.json",
  "method": "POST",
  "body":
  {
      "location": {
        "x": -1.553927,
        "y": 47.21858
      },
      "resources": [
        {
          "id": "1",
          "x": -1.593927,
          "y": 47.18858,
          "priority1": 1,
          "priority2": 1
        },
        {
          "id": "2",
          "x": -1.556927,
          "y": 47.21958,
          "priority1": 2,
          "priority2": 1
        }
      ],
      "method": "time",
      "reverse": "false"
    }
}
```

Parameters / properties

Input

parameter	description	optional	default
location	Coordinates of the start point (or arrival point)	no	
method	Method to compute the route between the start point and the resources: - distance : shortest route - time : fastest route - flying : as the crow flies	yes	time

parameter	description	optional	default
reverse	if *true*, the location point is considered as an arrival	yes	false
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	epsg:4326
exclusions	List of restriction rules to use, separated by the *;* character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone faible mission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
resources	List of resources, separated by ";" characters. Each resource takes the form "id,x,y,node,priority1,priority2" Priorities are optional.	yes	
startDateTime	Start date and time (format ISO8601: local time) example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a start on 21 January 2014, at 9.00am in Paris	yes	
snapMethod	Snap to graph method - standard: to the nearest connectable road segment - extended: via restricted road sections (pedestrian thoroughfare…) - nearest: to the nearest road section only - unrestricted: without any restriction rules - nodes: snap directly to the nodes indicated by locationNode and the *node* parameters of ressources or, if locationNode is not filled, snap directly to the nearest node location parameter.	yes	standard
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` Care: WKT geometries must be closed.	yes	
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - pedestrian (not available for Japan) - taxi - truck	yes	
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2"	yes	

parameter	description	optional	default
	- weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000" - fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"		

(*) At least one of the two parameters location, and locationNode must be entered.

Output

Search Around

parameter	type	min/max	description
location	couple of double, separated with a comma	0/1	Coordinates of the start point (or arrival point). The longitude and latitude coordinates are separated by the , character.
method	string	0/1	Specified input method (time, distance or flying), and time by default.
srs	string	0/1	Input projection specified.
exclusions	string	0/ unlimited	List of rules for specified input restrictions
searchAroundResult	Array of SearchAroundElement	0/ unlimited	List of results for each candidate.

Search Around element (SearchAroundElement)

parameter	type	min/max	description
id	string	0/1	Candidate identifier
distanceMeters	double	1/1	Distance in meters
durationSeconds	double	1/1	Time in seconds

Possible returns

Case of a proximity search finding(searchAroundResponse/status is OK)

```
{
  "message": null,
  "status": "OK",
  "location": "-1.553927,47.21858",
  "locationNode": null,
  "method": "TIME",
  "srs": "epsg:4326",
  "exclusions": [],
  "searchAroundResult": [
    {
      "id": "1",
      "distanceMeters": 6681,
      "durationSeconds": 877
    },
    {
      "id": "2",
      "distanceMeters": 425,
      "durationSeconds": 100
    }
  ]
}
```


Case of a snap-to-graph error (searchAroundResponse/status is OK)


```
{
  "message": null,
  "status": "OK",
  "location": "47.21858,-1.553927",
  "locationNode": null,
  "method": "TIME",
  "srs": "epsg:4326",
  "exclusions": [],
  "searchAroundResult": [
    {
      "id": "1",
      "distanceMeters": -1,
      "durationSeconds": -1
    },
    {
      "id": "2",
      "distanceMeters": -1,
      "durationSeconds": -1
    }
  ]
}
```


Case of an absent location parameter (searchAroundResponse/status is ERROR)


```
{
  "message":"Location must be not null",
  "status":"ERROR",
  "location":null,
  "locationNode":null,
  "method":null,
  "srs":null,
  "exclusions":[],
  "searchAroundResult":[]
}
```

Case of an incomplete location parameter, or one with the wrong separator (searchAroundResponse/status is ERROR)

```
{
  "message": "Location point must have 2 components separated with a ,",
  "status": "ERROR",
  "location": null,
  "locationNode": null,
  "method": null,
  "srs": null,
  "exclusions": [],
  "searchAroundResult": []
}
```

Case of the absence of a resource (searchAroundResponse/status is ERROR)

```
{
  "message":"resources not defined",
  "status":"ERROR",
  "location":null,
  "locationNode":null,
  "method":null,
  "srs":null,
  "exclusions":[],
  "searchAroundResult":[]
}
```

Case of a badly formatted resource (searchAroundResponse/status is ERROR)

```
{
  "message":"ServiceException: not enough fields in candidate 1,-1.59392",
  "status":"ERROR"
}
```

FAQ

1. Is it possible to give priority to either journey time or journey distance?

   Yes, by changing the method: distance, time or flying = *as the crow flies* (at a speed of 30 km/h)..

2. How does one structure the classification of returned addresses, in relation to the distance, the time, and to priorities generally?

   The classification creates a hierarchy of priority 1 in increasing order, then priority2 in increasing order, then by time or distance as the crow flies in increasing order.

3. What format should the priority take and is it possible to define a classification order? (increasing/decreasing)

   The priority is an integer, and currently the result is always in increasing order. To obtain the reverse ordering, you should put n-priority in the attribute.

4. If just one address in the list has a priority of 0, is the priority taken into account for one of the addresses in the list?

   Currently, a priority of 0 is not treated in any special way. You need to therefore set all priorities to 0 if you want to ignore the criterion.

5. How do you perform a search around calculation without including any road tolls?

   Place the exclusion *Toll* in `exclusions` .

6. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255
emergency vehicle						
pedestrian						
taxi						
truck	4	19000	39000	387	1850	255

7. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on `exclusions` toll road sections are not used.
BoatFerry	if added on `exclusions` ferry itineraries (shipping) over water are not used.
RailFerry	if added on `exclusions` ferry (or train) itineraries (over water and land respectively) are not used.
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.

Exclusion	Description
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

8. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …
   - Pollutant: goods harmful for water
   - Toxic: organic, poison, radioactive, corrosive, …
   - Dangerous: tunnel categories and others restrictions

9. What are Speed Patterns? How are they used?

   With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

   - standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
   - night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
   - busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
   - peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)

- default *default* corresponds to average speeds over a whole day

To use these, you will need to pass to parameter, when calling the web service, the following parameter `computeOptions` with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
Example:

```
&computeOptions=speedPattern:fast-speed&configName=car
```

10. How do you use heavy goods vehicle attributes?

    You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter `computeOptions` with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
    Example for a journey with a vehicle with a height of 4.5m:

```
&computeOptions=height:450
```

Search Along

Basic principles

The Search Along web service allows you to identify the best candidates from which to select the next nearest step in a pre-existing itinerary.

The algorithm explores all possible solutions before returning the best candidate selected as a function of best score for the requested criteria.

Search along example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/searchAlong/v2.json",
        "method":"POST",
        "body":
      {
       "routes":[
          {
             "id":"dep",
             "origin":{
                "x":-1.553927,
                "y":47.218580
             },
             "destination":{
                "x":-1.593927,
                "y":47.188580
             },
              "waypoints" : [ {
                "x":-1.563927,
                "y":47.224680
               }, {
                "x":-1.559927,
                "y":47.212458
            } ]
          }
       ],
```

```
        "resources":[
           {
              "id":"res1",
              "x":-1.511092,
              "y":47.208354
           },
           {
              "id":"res2",
              "x":-1.549524,
              "y":47.195483
           }
        ],
        "method":"time",
        "srs":"epsg:4326",
        "maxDetourDurationSeconds":-1,
        "maxDetourDistanceMeters":-1,
        "maxDetourOriginDurationSeconds":-1,
        "maxDetourOriginDistanceMeters":-1,
        "maxDetourDestinationDurationSeconds":-1,
        "maxDetourDestinationDistanceMeters":-1
        }
}
```

Parameters / properties

Input

parameter	description	optional	default
routes array of routes/ route (inputRouteV2)	Table of journey input data. Pre-existing itineraries in which candidate steps must be inserted.	yes *	
resources (array of Point)	Candidate resources	yes	
method	Shortest (distance) or fastest (time) journey	yes	time
exclusions	List of restriction rules to use, separated by the ; character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone à faible émission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
snapMethod	Snap-to-graph method - standard: to the nearest connectable road section - extended: via restricted road sections (pedestrian routes…) - nearest: to the nearest road section only - unrestricted: without any restriction rules - nodes: Snap directly to nodes provided by the locationNode parameter or, if no value has been assigned, to the node nearest to the location parameter	yes	standard
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` Care: WKT geometries must be closed.	yes	

parameter	description	optional	default
startDateTime	Start date and time (format ISO8601: local time) example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a start on 21 January 2014, at 9.00am in Paris	yes	
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	
configName	Name of the configuration to use: - bicycle (not available for Japan) - bus - car - delivery light vehicle - delivery truck vehicle - emergency truck - emergency vehicle - pedestrian (not available for Japan) - taxi - truck	yes	
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000" - fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"	yes	
maxDetourDurationSeconds	Filter: maximum detour authorised (in seconds)	yes	
maxDetourDistanceMeters	Filter: maximum detour authorised (in meters)	yes	
maxDetourOriginDurationSeconds	Filter: maximum authorised duration between the journey start point and the candidate (in seconds)	yes	
maxDetourOriginDistanceMeters	Filter: maximum authorised detour between the candidate and the journey arrival point (in meters)	yes	

parameter	description	optional	default
maxDetourDestinationDurationSeconds	Filter. Maximum authorised duration between the candidate and the journey arrival point (in seconds)	yes	
maxDetourDestinationDistanceMeters	Filter. Maximum authorised detour from the arrival point (in meters)	yes	

(*) At least one of the two parameters routes, and routeNodes must be entered.

Journeys in input (inputRouteV2)

parameter	description	optional	default
id	Journey identifier	No	
origin (Point)	Coordinates of the journey departure point	No	
destination (Point)	Coordinates of the journey arrival point	No	
waypoints (array of Point)	waypoints of the journey	No	

Geolocalized point (Point)

parameter	description	optional	default
id	Point identifier	No	
x	First coordinate or longitude	No	
y	Second coordonnée or latitude	No	
nodeId	Point node id on the graph	No	

Output

parameter	type	min/max	description
routes	array of routes/route (searchAlongRouteResultV2)	0/ unlimited	List of calculated journeys.

List of calculated journeys (searchAlongRouteResultV2)

parameter	type	min/max	description
id	string	0/1	Journey identifier
directDistanceMeters	double	1/1	Total distance without detour (in meters)
directDurationSeconds	double	1/1	Total duration without detour (in seconds)
resources	(searchAlongResourceResultV2)	0/ unlimited	List of candidates

Candidates (searchAlongResourceResultV2)

parameter	type	min/max	description
id	string	0/1	Candidate identifier
totalDistanceMeters	double	1/1	Total distance with detour (in meters)
totalDurationSeconds	double	1/1	Total duration with detour (in seconds)
detourDistanceMeters	double	1/1	Detour distance (in meters)
detourDurationSeconds	double	1/1	Detour duration (in seconds)
detourOriginDistanceMeters	double	1/1	Detour distance from the journey departure point to the candidate (in meters)

parameter	type	min/max	description
detourOriginDurationSeconds	double	1/1	Detour duration from the journey departure point to the candidate (in seconds)
detourDestinationDistanceMeters	double	1/1	Detour distance from the candidate to the arrival point (in meters)
detourDestinationDurationSeconds	double	1/1	Detour duration from the candidate to the arrival point (in seconds)
subPathIndex	double	1/1	Index of the sub path in the input route for this candidate
previousStepId	string	1/1	Id of the origin of the sub path in the input route for this candidate
nextStepId	string	1/1	Id of the destination of the sub path in the input route for this candidate

FAQ

1. Is it possible to give priority to either journey time or journey distance?

   Yes, by changing the method: distance or time

2. How does one structure the classification of returned addresses, in relation to the distance, the time, and to priorities generally?

   The classification creates a hierarchy of priority 1 in increasing order, then priority2 in increasing order, then by time or distance as the crow flies in increasing order.

3. What format should the priority take and is it possible to define a classification order? (increasing/decreasing)

   The priority is an integer, and currently the result is always in increasing order. To obtain the reverse ordering, you should put n-priority in the attribute.

4. If just one address in the list has a priority of 0, is the priority taken into account for one of the addresses in the list?

   Currently, a priority of 0 is not treated in any special way. You need to therefore set all priorities to 0 if you want to ignore the criterion.

5. How do you perform a search around calculation without including any road tolls?

   Place the exclusion *Toll* in `exclusions` .

6. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255

configName	axles	weightPerAxle	weight	height	length	width
emergency vehicle						
pedestrian						
taxi						
truck	4	19000	39000	387	1850	255

7. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on `exclusions` toll road sections are not used.
BoatFerry	if added on `exclusions` ferry itineraries (shipping) over water are not used.
RailFerry	if added on `exclusions` ferry (or train) itineraries (over water and land respectively) are not used.
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName`

Exclusion	Description
	with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

8. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …
   - Pollutant: goods harmful for water
   - Toxic: organic, poison, radioactive, corrosive, …
   - Dangerous: tunnel categories and others restrictions

9. What are Speed Patterns? How are they used?

   With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

   - standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
   - night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
   - busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
   - peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)
   - default *default* corresponds to average speeds over a whole day

   To use these, you will need to pass to parameter, when calling the web service, the following parameter `computeOptions` with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
   Example:

   ```
   &computeOptions=speedPattern:fast-speed&configName=car
   ```

10. How do you use heavy goods vehicle attributes?

    You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter `computeOptions` with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
    Example for a journey with a vehicle with a height of 4.5m:

```
&computeOptions=height:450
```

Pickup and delivery

Basic principles

The pickup and delivery service can be used to list the best solutions for pickup/collection and drop-off/delivery, whether for the transport of people or of goods.

The web service takes two types of data input:

- Existing journeys
- Pickup/collection points and drop-off/delivery points

The algorithm explores the solutions to find a pair that includes one pick-up point and one drop-off point in the existing journeys. It returns, for each journey, the best detours possible among the candidates (pick-up and drop-off points).

In the case where only one candidate (pickup or drop-off) is available, the web service tries to insert one step in existing journeys instead of two.

pickup and delivery example

```
{
        "url":"https://api.geoconcept.com/EU/GCW/geoconcept-web/api/lbs/pickupDelivery.json",
        "method":"POST",
        "body":
    {
      "routes" : [ {
        "id" : "trip_1",
        "departure" : {
          "lat" : "47.333990",
          "lon" : "-1.805604"
        },
        "arrival" : {
          "lat" : "47.479740",
          "lon" : "-1.095251"
        },
        "waypoints" : [ {
          "id" : "waypoint_1",
          "location" : {
            "lat" : "47.157530",
            "lon" : "-1.421958"
          }
        }]
      }],
      "pickupPoints" : [ {
        "id" : "depMeet_1",
        "location" : {
          "lat" : "47.228659",
          "lon" : "-1.600995"
        }
      }],
      "deliveryPoints" : [ {
        "id" : "arrMeet_1",
        "location" : {
          "lat" : "47.293823",
```

```
          "lon" : "-1.480789"
        }
      }],
      "constraints" : {
        "maxDetourDuration" : 3600,
        "minSharedDistance" : 0.1
      },
      "sortOptions" : {
        "routesAndCandidatesSortCriterium" : "MIN_DETOUR_DURATION"
      },
      "configName":"car",
      "size" : {
        "routes" : 1,
        "candidates" : 1
      }
    }
}
```

Parameters / properties

Input

parameter	type	optional	description
routes	array of routes/route (pickupDeliveryRouteInput)	No	Table of journeys as input. Pre-existing journeys in which candidates must be inserted.
pickupPoints	array of pickupPoints/pickupPoint (wayPoint)	Yes	Table of candidate pick-up points to insert in journeys.
deliveryPoints	array of deliveryPoints/deliveryPoint (wayPoint)	Yes	Table of possible drop-off candidates to insert in journeys.
constraints	(pickupDeliveryConstraints)	Yes	Definition of constraints to filter the journeys to use.
sortOptions	(pickupDeliverySortingOptions)	Yes	Definition of the sort on the results.

parameter	type	optional	description
srs	projection (EPSG code such as epsg:4326 or wgs84)	yes	
configName	Name of the configuration to use: - car - truck	yes	
snapMethod	Snap to graph method - standard: to the nearest connectable road segment - extended: via restricted road sections (pedestrian routes…) - nearest: to the nearest road section only - unrestricted: without any restriction rules - nodes: snap directly to nodes supplied by the originNode originNode, destinationNode and waypointNodes parameters or, if these first ones are not filled, snap directly to the nearest nodes origin, destination and waypoint parameters.	yes	standard
exclusions	List of restriction rules to use, separated by the *;* character Available exclusion rules: - Toll - Infrastructure (Tunnel, Bridge) - Hazardous material (Flammable, Dangerous, Pollutant, Toxic) - ZFE (Zone **&#65533;** faible **&#65533;**mission, France-only) - … - Detailled list available on the FAQ bellow.	yes	
startDateTime	Start date and time (format ISO8601: local time) example: 2014-01-21T09:00:00.000+01:00 (or 2014-01-21T09:00:00.000%2B01:00) for a start on 21 January 2014, at 9.00am in Paris	yes	
avoidArea	Forbidden transit zone in WKT format (POLYGON or MULTIPOLYGON) in the requested projection (srs parameter) Example in wgs84: `POLYGON ((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689))` - `MULTIPOLYGON (((-1.556892 47.21689, -1.556892 47.216904, [...] -1.556892 47.21689)), ((-1.558639 47.218144, -1.558639 47.218193, [...] -1.558639 47.218144)))` Care: WKT geometries must be closed.	yes	
size	(pickupDeliverySizes)	No	Number of journeys and candidates to return.
dataVersionHash	string	Yes	Graph identifier (non-utilised)
computeOptions	Selection of variables to use in calculations, using semi-colons *;* as separators: - speedPattern: utilization as a *speed pattern*. Use: "speedPattern:slow-speed" - length: maximum authorised length in centimeters. Use: "length:950" - width: maximum authorised width in centimeters. Use: "width:255" - height: maximum authorised height in centimeters. Use: "height:360" - weight: maximum authorised weight in kilograms. Use: "weight:18000" - axles: maximum authorised number of axels. Use: "axles:2" - weightPerAxle: maximum authorised weight per axel in kilograms. Use: "weightPerAxle:9000"	yes	

parameter	type	optional	description
	- fuelType: fuel type used to calculate carbon footprint for the journey. Available values are: "UndefinedFuelType", "NoFuel", "Diesel", "UnleadedFuel", "LGP", "CustomFuelType" and "Electrical". Use: "fuelType:Diesel" - averageConsumption: average consumption in litres for 100 kilometers or Wh per kilometers if electrical vehicle. Use: "averageConsumption:6.45" - cityAverageConsumption: average consumption value in city mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "cityAverageConsumption:4.65" - combinedAverageConsumption: average consumption value in combined mode, in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "combinedAverageConsumption:5.12" - highwayAverageConsumption: average consumption value in highway mode, in value in litres per 100 kilometers, or Wh per kilometers if electrical vehicle. Use: "highwayAverageConsumption:5.64" - customAverageCO2UnitEmission: defines the carbon footprint in kilograms per litre. Use: "customAverageCO2UnitEmission:2.724" - snapSpeed: snap-to-graph speed in kilometers per hour. Use: "snapSpeed:10"		

(pickupDeliveryRouteInput) journeys as input

parameter	type	optional	description
id	string	No	Journey identifier
departure	(location)	No	Start point coordinates
arrival	(location)	No	Finish point coordinates
distance	long	Yes	Journey distance (in meters)
duration	long	Yes	Journey duration (in seconds)
waypoints	Array of wayPoint	Yes	List of journey steps

(wayPoint) steps

parameter	type	optional	description
id	string	No	Step identifier
location	(location)	No	Step coordinates
stopoverPosition	long	Yes	Position of the

parameter	type	optional	description
			step in the journey, if the step is a stop. If no values are assigned, this step is just a waypoint.

(location) coordinates

parameter	type	optional	description
lat	double	No	Latitude of the point (in wgs84)
long	double	No	Longitude of the point (in wgs84)
nodeId	string	Yes	Graph node. Caution: a physical node does not have the same ID in another graph.

(pickupDeliveryConstraints) contraintes

parameter	type	optional	description
maxDetourDuration	long	Yes	Maximum duration of detour (in seconds). Only resulting journeys for which the duration added is less than this limit will be returned by the web service.

parameter	type	optional	description
			The detour duration is defined as: (duration with detour) - (duration without detour)
minSharedDistance	long	Yes	Minimum value of common distance ratio. Only those resulting journeys for which the common distance ratio exceeds this limit will be returned by the web service. The common distance ratio is defined as: (common distance) / (total distance with detour)

(pickupDeliverySortingOptions) sort

parameter	type	optional	description
routesAndCandidatesSortCriteria	(sortCriteriaEnum)	Yes	Sort options: - MIN_DETOUR_DURATION = sort (ascending)

parameter	type	optional	description
			results according to detour duration - MIN_DETOUR_DISTANCE = sort (ascending) results according to detour distance - MAX_SHARED_DISTANCE = sort (descending) results according to common distance ratio

(pickupDeliverySizes) number of returns

parameter	type	optional	description
routes	long	No	Maximum number of journeys to return.
candidates	long	No	Maximum number of candidates to return for each journey.

Output

parameter	type	description
results	array of results/result (pickupDeliveryRouteResult)	List of journeys calculated.

(pickupDeliveryRouteResult) list of journeys calculated

parameter	type	description
routeId	string	List of journeys calculated.
candidates	array of candidates/ candidate (pickupDeliveryRouteCandidate)	List of possible detours for the journey.

(pickupDeliveryRouteCandidate) list of detours possible for the journey

parameter	type	description
pickup	(meetingPointCandidate)	Pick-up candidate.
delivery	(meetingPointCandidate)	Drop-off candidate.
sharedRouteDuration	long	Common journey duration (in seconds).
sharedRouteDistance	long	Common journey distance (in meters).
totalDurationWithDetour	long	Total journey duration including the detour (in seconds).
totalDistanceWithDetour	long	Total journey distance including the detour (in meters).
detourDuration	long	Detour duration = (duration with detour) - (original journey duration)
detourDistance	long	Detour distance = (distance with detour) - (original journey distance)
sharedDistance	double	Common distance = ratio between (common distance) and (Total journey distance including the detour)

(meetingPointCandidate) candidates

parameter	type	description
meetingPointId	string	Identifier for the pickup/drop-off location
duration	long	For a pickup link: duration from the journey departure point to the pickup point (in seconds) For a drop-off point: duration from the journey drop-off point to the arrival point (in seconds)
distance	long	For a pickup link: distance from the journey departure point to the pickup point (in meters) For a drop-off point: the distance from the journey drop-off point to the arrival point (in meters)
meetingPointSubPath	int	Optimum segment position (in the journey) for the pickup/drop-off point.

Possible returns

Case of an optimization applied successfully


```
{
   "steps": [
      {
        "x": 2.33906,
        "y": 48.84467,
        "id": "1",
        "duration": 600000,
        "effectiveStart": 0,
        "driveDistanceBefore": 0,
        "driveDistanceAfter": 5057,
        "driveTimeBefore": 0,
        "driveTimeAfter": 737740,
        "timeWindows": [
          {
            "start": 0,
            "end": 3600000
          },
          [...]
```

```
          ]
        },
        [...]
    ],
    "distanceMeters": 13502,
    "durationSeconds": 2015
}
```

FAQ

1. How perform a route calculation without including any road tolls?

   Place the exclusion *Toll* in `exclusions` .

2. What are the maximum values allowed by predefined configName?

   See list below:

configName	axles	weightPerAxle	weight	height	length	width
bicycle						
bus	2	9500	19000	350	1350	250
car						
delivery light vehicle	2	1700	3400	300	680	220
delivery truck vehicle	2	9500	19000	350	1350	250
emergency truck vehicle	2	9000	18000	360	920	255
emergency vehicle						
pedestrian						
taxi						
truck	4	19000	39000	387	1850	255

3. What are the exclusions available?

   See list below:

Exclusion	Description
Toll	if added on `exclusions` toll road sections are not used.
BoatFerry	if added on `exclusions` ferry itineraries (shipping) over water are not used.
RailFerry	if added on `exclusions` ferry (or train) itineraries (over water and land respectively) are not used.
Tunnel	if added on `exclusions` tunnels are not used.
Bridge	if added on `exclusions` bridges are not used.
Frontage	if added on `exclusions` Frontage roads (local roads running parallel to a higher-speed, limited-access road) are not used.

Exclusion	Description
Paved	if added on `exclusions` tarmac road sections are not used.
Private	if added on `exclusions` road sections for which maintenance is provided by a private organization are not used.
Pedestrians	if added on `exclusions` road sections unauthorised for pedestrian use are not used. The `configName` parameter with a value of pedestrian uses this exclusion.
Automobiles	if added on `exclusions` road sections with restricted automobile access are not used. The `configName` parameter with a value of car uses this exclusion.
Trucks	if added on `exclusions` road sections with restricted access for lorries are not used. The `configName` with a value of truck or delivery truck vehicle or emergency truck uses this exclusion.
Bus	if added on `exclusions` road sections with restricted access for buses are not used. The `configName` with a value of bus uses this exclusion.
Deliveries	if added on `exclusions` road sections with restricted access for deliveries are not used. The `configName` with a value of delivery light vehicle or delivery truck vehicle uses this exclusion.
Taxis	if added on `exclusions` road sections with restricted access for taxis are not used. The `configName` with a value of taxi uses this exclusion.
Emergencies	if added on `exclusions` road sections with restricted access for emergencies are not used. The `configName` with a value of emergency truck or emergency vehicle uses this exclusion.
Carpools	if added on `exclusions` road sections with restricted access for carpools are not used.
Throughtraffic	if added on `exclusions` road sections with restricted access for non-local traffic are not used.
ZFE	if added on `exclusions` low carbon emmission areas in France will be excluded
(Hazardous material)	See description bellow.

4. What are the Hazardous material exclusions?

   Hazardous material defines a restriction to a road for any vehicle carrying the specific hazardous material. Hazardous material restrictions are government regulations and can be different per country.

   - Flammable: explosives, flammable, gas, combustible, …
   - Pollutant: goods harmful for water
   - Toxic: organic, poison, radioactive, corrosive, …
   - Dangerous: tunnel categories and others restrictions

5. What are Speed Patterns? How are they used?

With the aim of suggesting journey times that take traffic conditions into account as accurately as possible, cars and trucks have 5 speed profiles (Speed Patterns) so various levels of road congestion can be taken into account over the timespan of a whole day:

- standard *normal-speed* corresponds to the speed at a time when the traffic congestion is moderate (11h)
- night *fast-speed* corresponds to very fluid traffic conditions, most often found at night time (3h)
- busy *slow-speed* corresponds to the times when traffic congestion is at its densest (8h)
- peak travel time *very-slow-speed* corresponds to times when traffic is densest in urban areas, and is slower than the category given above (8h)
- default *default* corresponds to average speeds over a whole day

To use these, you will need to pass to parameter, when calling the web service, the following parameter **`computeOptions`** with the option *speedPattern* and specify the value of the Speed Pattern requested, always remembering to include a vehicle configName.
Example:

```
&computeOptions=speedPattern:fast-speed&configName=car
```

6. How do you use heavy goods vehicle attributes?

   You need to calculate an itinerary using either a vehicle configName using the restrictions, or by overwriting the call to the web service using the parameter **`computeOptions`** with the options *length*, *width*, *height*, *weight*, *axles* and/or *weightPerAxle*.
   Example for a journey with a vehicle with a height of 4.5m:

```
&computeOptions=height:450
```

TourSolver Cloud API

Overview

Introduction

This documentation presents the REST APIs of TourSolver Cloud.

The REST APIs provide programmatic access to creation, launching and reading of TourSolver optimizations. Thus you'll be able to plan and optimize order deliveries while conforming to many constraints, defined on customers, resources and depots. You can also use this API to publish your optimized tour plans to Toursolver Mobile App and follow the fulfilment.

The main services are :

- optimize : send your data and start the optimization
- status : follow the optimization process (cost, distance and time evolution)
- stop : you can set a maximum optimization time, but you can also stop the optimization when you decide (ex: if global cost has not changed for a while)

- result / toursResult : retrieve the optimized result (bulk or tour organized)
- exportToOperationalPlanning : publish the result on Toursolver Mobile app
- fulfillment : follow the fulfillment of the tours by yours resources on the field

If you don’t need to follow the optimization process, you can use our webhook system : just start the optimization and the result will be automatically posted when ready to your server.

If you don’t want to build a result page, just integrate Toursolver result page in your app (more details in the tutorial).

Web APIs for TourSolver Cloud

Version information

Version : 2.0

Tags

- ToursolverWebService : Toursolver optimization web service.

Authentication

In order to be able to use this api, you must provide your api key through the tsCloudApiKey parameter. This parameter can be passed either in the query or as a http header.

Caution : if you send to many requests in a short time, you may receive a 429 http code. When it happens, that means that your request as not been treated, you will have to send it again later.

Resources

This section presents the services.

This API consumes and produces json and xml. To specify the format you want to use, just add HTTP headers Accept and Content-Type.

If you use json, you should add these headers :

- Accept: application/json
- Content-Type: application/json

If you use xml, you should add these headers :

- Accept: application/xml
- Content-Type: application/xml

The root url for this API is :

- https://api.geoconcept.com/tsapi/ for most of the world.

```
Note: the former api URL (https://geoservices.geoconcept.com/ToursolverCloud/api/ts/toursolver/) is
 still working but will be removed in the future.
```

- https://geoservices.geoconcept.cn/ToursolverCloud/api/ts/toursolver/ for China only.

Webhook integration :

> Webhook allow you to automatically receive optimization result when optimization finishes. Rather than requiring you to pull information via our API, webhooks will push information to your endpoint. ToursolverCloud will send a HTTP POST payload to the webhook's configured in ToursolverCloud configuration menu. You will be able to specify in your optimization request if you want the result to be sent automatically to this webhook when the optimization ends and what result type you need (orders list or shipments list). Have a look to the sendResultToWeebhook parameter of the TSOptions object for more details.

Webhooks retry logic is as follows :

> In case the original notification sending attempt fails (due to receiving a non-2xx response code or exceeding timeout of 10 seconds), we will try 3 more times: after 3, 30 and 150 seconds. If it still fails for each of those attempts, the webhook will be disabled and an email will be sent to warn you.

ToursolverWebService

Toursolver optimization web service.

Parameters

Type	Name	Description	Schema	Default
Body	body *optional*	visits to add, with id of target simulation or job, and language code	json_AddVisitsRequest	

Responses

HTTP Code	Description	Schema
201	Success	json_AddVisitsResult

Consumes

- `application/xml`
- `application/json`
- `text/xml`

Produces

- `application/xml`

- `application/json`
- `text/xml`

Example HTTP request

=.Request body

```
{
  "id" : "...",
  "orders" : [ {
    "allSkillsRequired" : true,
    "assignResources" : [ "...", "..." ],
    "courierPenalty" : 12345.0,
    "customDataMap" : {
      "property1" : "...",
      "property2" : "..."
    },
    "delayPenaltyPerHour" : 12345.0,
    "excludeResources" : [ "...", "..." ],
    "fixedVisitDuration" : "...",
    "frequency" : "...",
    "id" : "...",
    "minDuration" : "...",
    "minPartDuration" : "...",
    "punctuality" : 12345,
    "quantities" : [ 12345.0, 12345.0 ],
    "requiredSkills" : [ "...", "..." ],
    "resourceCompatibility" : 12345.0,
    "sequenceNumber" : 12345,
    "timeWindows" : [ {
      "beginTime" : "08:00",
      "endTime" : "18:00"
    }, {
      "beginTime" : "08:00",
      "endTime" : "18:00"
    } ],
    "travelTimeModifier" : {
      "offset" : "...",
      "value" : 12345.0,
      "length" : "..."
    },
    "type" : 12345,
    "unloadingDurationPerUnit" : "...",
    "x" : 12345.0,
    "y" : 12345.0,
    "active" : true,
    "wholeVisitInTimeWindow" : true,
    "label" : "...",
    "evaluationInfos" : {
      "orderOriginalResourceId" : "...",
      "orderOriginalVisitDay" : "...",
      "orderPosition" : 12345
    },
    "possibleVisitDaysList" : [ "...", "..." ],
    "tsOrderMaximumSpacing" : 12345,
    "tsOrderMinimumSpacing" : 12345,
    "tsOrderLastVisit" : 12345,
    "customerId" : "...",
    "email" : "...",
    "phone" : "...",
    "tsOrderBefore" : "...",
    "tsOrderBeforeMaxTimeSpacing" : "...",
    "tsOrderBeforeMinTimeSpacing" : "...",
```

```
      "getNotifications" : true,
      "tsOrderFixed" : true
    }, {
      "allSkillsRequired" : true,
      "assignResources" : [ "...", "..." ],
      "courierPenalty" : 12345.0,
      "customDataMap" : {
        "property1" : "...",
        "property2" : "..."
      },
      "delayPenaltyPerHour" : 12345.0,
      "excludeResources" : [ "...", "..." ],
      "fixedVisitDuration" : "...",
      "frequency" : "...",
      "id" : "...",
      "minDuration" : "...",
      "minPartDuration" : "...",
      "punctuality" : 12345,
      "quantities" : [ 12345.0, 12345.0 ],
      "requiredSkills" : [ "...", "..." ],
      "resourceCompatibility" : 12345.0,
      "sequenceNumber" : 12345,
      "timeWindows" : [ {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      }, {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      } ],
      "travelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "type" : 12345,
      "unloadingDurationPerUnit" : "...",
      "x" : 12345.0,
      "y" : 12345.0,
      "active" : true,
      "wholeVisitInTimeWindow" : true,
      "label" : "...",
      "evaluationInfos" : {
        "orderOriginalResourceId" : "...",
        "orderOriginalVisitDay" : "...",
        "orderPosition" : 12345
      },
      "possibleVisitDaysList" : [ "...", "..." ],
      "tsOrderMaximumSpacing" : 12345,
      "tsOrderMinimumSpacing" : 12345,
      "tsOrderLastVisit" : 12345,
      "customerId" : "...",
      "email" : "...",
      "phone" : "...",
      "tsOrderBefore" : "...",
      "tsOrderBeforeMaxTimeSpacing" : "...",
      "tsOrderBeforeMinTimeSpacing" : "...",
      "getNotifications" : true,
      "tsOrderFixed" : true
    } ],
    "language" : "..."
  }
```


Example HTTP response

=.Response 201

```
{
  "application/json" : {
    "status" : "...",
    "message" : "..."
  }
}
```

```
GET /toursolver/depots
```

Description

Get known depots

Get depots defined in ToursolverCloud GUI.

Parameters

Type	Name	Description	Schema	Default
Query	depotName *optional*	if a depot name is specified only one depot will be returned.	string	

Responses

HTTP Code	Description	Schema
200	Success	json_DepotsResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200

```
{
  "application/json" : {
    "depots" : [ {
      "id" : "...",
      "openingDaysList" : [ "...", "..." ],
      "timeWindows" : [ {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      }, {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      } ],
      "availability" : true,
      "resourceNames" : "...",
      "excludeResources" : "...",
      "travelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
```

```
      },
      "fixedLoadingDuration" : "...",
      "loadingDurationPerUnit" : "...",
      "priority" : 12345,
      "requiredProducts" : "...",
      "allProductsRequired" : true,
      "deliveryQuantities" : [ 12345, 12345 ],
      "pickupQuantities" : [ 12345, 12345 ],
      "x" : 12345.0,
      "y" : 12345.0
    }, {
      "id" : "...",
      "openingDaysList" : [ "...", "..." ],
      "timeWindows" : [ {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      }, {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      } ],
      "availability" : true,
      "resourceNames" : "...",
      "excludeResources" : "...",
      "travelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "fixedLoadingDuration" : "...",
      "loadingDurationPerUnit" : "...",
      "priority" : 12345,
      "requiredProducts" : "...",
      "allProductsRequired" : true,
      "deliveryQuantities" : [ 12345, 12345 ],
      "pickupQuantities" : [ 12345, 12345 ],
      "x" : 12345.0,
      "y" : 12345.0
    } ],
    "message" : "...",
    "status" : "ERROR"
  }
}
```


```
POST /toursolver/exportToOperationalPlanning
```

Description

Export to operational planning.

Exports the result of an optimization to operational planning that mobile resources will be able to browse on the field through a mobile app.

This command only works on completed optimizations. Mobile resource identifiers must have been set previously in TsCloud app. If optimization period overlaps an already existing operational planning, export will work only if the force attribut has been set to true.

Parameters

Type	Name	Description	Schema	Default
Body	body		json_OperationalExportRequest	

Type	Name	Description	Schema	Default
	optional			

Responses

HTTP Code	Description	Schema
201	Success	json_TtoursolverServiceResult

Consumes

- `application/xml`
- `application/json`
- `text/xml`

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP request

=.Request body

```
{
  "resourceMapping" : [ {
    "id" : "...",
    "operationalId" : "..."
  }, {
    "id" : "...",
    "operationalId" : "..."
  } ],
  "startDate" : 12345,
  "force" : true,
  "taskId" : "...",
  "dayNums" : [ 12345, 12345 ]
}
```

Example HTTP response

=.Response 201

```
{
  "application/json" : {
    "message" : "...",
    "status" : "OK"
  }
}
```

```
GET /toursolver/fulfillment
```

Description

Get fulfillment information.

Get fulfillment information for a specified period of time.

Parameters

Type	Name	Description	Schema	Default
Query	endDate *optional*	planning period end	string	
Query	lastUpdate *optional*	if specified, only operationalOrders that have been modified since this date will be returned.	string	
Query	startDate *optional*	planning period start	string	
Query	userLogin *optional*	a mobile resource login (optional)	string	

Responses

HTTP Code	Description	Schema
200	Success	json_FulfillmentResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200

```
{
  "application/json" : {
    "operationalOrderAchievements" : [ {
      "operationalResourceId" : "...",
      "plannedOrder" : {
        "dayId" : "...",
        "stopPosition" : 12345,
        "stopY" : 12345.0,
        "stopX" : 12345.0,
        "stopId" : "...",
        "stopType" : 12345,
        "stopDriveTime" : "...",
        "stopStartTime" : "...",
        "stopDuration" : "...",
        "stopStatus" : 12345,
        "stopDriveDistance" : 12345,
        "stopElapsedDistance" : 12345,
        "resourceId" : "..."
      },
      "order" : {
        "allSkillsRequired" : true,
        "assignResources" : [ "...", "..." ],
        "courierPenalty" : 12345.0,
        "customDataMap" : {
          "property1" : "...",
          "property2" : "..."
        },
        "delayPenaltyPerHour" : 12345.0,
        "excludeResources" : [ "...", "..." ],
        "fixedVisitDuration" : "...",
```

```
          "frequency" : "...",
          "id" : "...",
          "minDuration" : "...",
          "minPartDuration" : "...",
          "punctuality" : 12345,
          "quantities" : [ 12345.0, 12345.0 ],
          "requiredSkills" : [ "...", "..." ],
          "resourceCompatibility" : 12345.0,
          "sequenceNumber" : 12345,
          "timeWindows" : [ { }, { } ],
          "travelTimeModifier" : { },
          "type" : 12345,
          "unloadingDurationPerUnit" : "...",
          "x" : 12345.0,
          "y" : 12345.0,
          "active" : true,
          "wholeVisitInTimeWindow" : true,
          "label" : "...",
          "evaluationInfos" : { },
          "possibleVisitDaysList" : [ "...", "..." ],
          "tsOrderMaximumSpacing" : 12345,
          "tsOrderMinimumSpacing" : 12345,
          "tsOrderLastVisit" : 12345,
          "customerId" : "...",
          "email" : "...",
          "phone" : "...",
          "tsOrderBefore" : "...",
          "tsOrderBeforeMaxTimeSpacing" : "...",
          "tsOrderBeforeMinTimeSpacing" : "...",
          "getNotifications" : true,
          "tsOrderFixed" : true
        },
        "date" : 12345,
        "start" : 12345,
        "end" : 12345,
        "status" : "FINISHED",
        "type" : "LUNCHBREAK",
        "lon" : 12345.0,
        "lat" : 12345.0,
        "lastSynchroStatusChange" : 12345,
        "synchroStatus" : "PUBLISHED",
        "achievementStart" : 12345,
        "achievementEnd" : 12345,
        "achievementComment" : "...",
        "achievementStartLat" : 12345.0,
        "achievementStartLon" : 12345.0,
        "achievementEndLat" : 12345.0,
        "achievementEndLon" : 12345.0,
        "geocode" : {
          "addressComplement" : "...",
          "address" : "...",
          "postcode" : "...",
          "city" : "...",
          "region" : "...",
          "country" : "...",
          "score" : 12345.0,
          "geocodeType" : 12345,
          "geocodeCity" : "...",
          "geocodePostalCode" : "...",
          "geocodeAddressLine" : "..."
        },
        "signatureSvg" : "...",
        "signaturePicture" : "...",
```

```
        "data" : {
          "property1" : "...",
          "property2" : "..."
        },
        "pictures" : [ "...", "..." ],
        "simulationId" : "...",
        "simulationDayId" : "...",
        "timeWindowEnd" : 12345,
        "timeWindowSmsId" : "...",
        "timeWindowSmsStatus" : "...",
        "timeWindowStart" : 12345,
        "apprauchSmsId" : "...",
        "apprauchSmsStatus" : "...",
        "feedbackSmsId" : "...",
        "feedbackSmsStatus" : "...",
        "id" : "..."
      }, {
        "operationalResourceId" : "...",
        "plannedOrder" : {
          "dayId" : "...",
          "stopPosition" : 12345,
          "stopY" : 12345.0,
          "stopX" : 12345.0,
          "stopId" : "...",
          "stopType" : 12345,
          "stopDriveTime" : "...",
          "stopStartTime" : "...",
          "stopDuration" : "...",
          "stopStatus" : 12345,
          "stopDriveDistance" : 12345,
          "stopElapsedDistance" : 12345,
          "resourceId" : "..."
        },
        "order" : {
          "allSkillsRequired" : true,
          "assignResources" : [ "...", "..." ],
          "courierPenalty" : 12345.0,
          "customDataMap" : {
            "property1" : "...",
            "property2" : "..."
          },
          "delayPenaltyPerHour" : 12345.0,
          "excludeResources" : [ "...", "..." ],
          "fixedVisitDuration" : "...",
          "frequency" : "...",
          "id" : "...",
          "minDuration" : "...",
          "minPartDuration" : "...",
          "punctuality" : 12345,
          "quantities" : [ 12345.0, 12345.0 ],
          "requiredSkills" : [ "...", "..." ],
          "resourceCompatibility" : 12345.0,
          "sequenceNumber" : 12345,
          "timeWindows" : [ { }, { } ],
          "travelTimeModifier" : { },
          "type" : 12345,
          "unloadingDurationPerUnit" : "...",
          "x" : 12345.0,
          "y" : 12345.0,
          "active" : true,
          "wholeVisitInTimeWindow" : true,
          "label" : "...",
          "evaluationInfos" : { },
```

```
          "possibleVisitDaysList" : [ "...", "..." ],
          "tsOrderMaximumSpacing" : 12345,
          "tsOrderMinimumSpacing" : 12345,
          "tsOrderLastVisit" : 12345,
          "customerId" : "...",
          "email" : "...",
          "phone" : "...",
          "tsOrderBefore" : "...",
          "tsOrderBeforeMaxTimeSpacing" : "...",
          "tsOrderBeforeMinTimeSpacing" : "...",
          "getNotifications" : true,
          "tsOrderFixed" : true
        },
        "date" : 12345,
        "start" : 12345,
        "end" : 12345,
        "status" : "REFUSED",
        "type" : "RELOADBREAK",
        "lon" : 12345.0,
        "lat" : 12345.0,
        "lastSynchroStatusChange" : 12345,
        "synchroStatus" : "UPDATED",
        "achievementStart" : 12345,
        "achievementEnd" : 12345,
        "achievementComment" : "...",
        "achievementStartLat" : 12345.0,
        "achievementStartLon" : 12345.0,
        "achievementEndLat" : 12345.0,
        "achievementEndLon" : 12345.0,
        "geocode" : {
          "addressComplement" : "...",
          "address" : "...",
          "postcode" : "...",
          "city" : "...",
          "region" : "...",
          "country" : "...",
          "score" : 12345.0,
          "geocodeType" : 12345,
          "geocodeCity" : "...",
          "geocodePostalCode" : "...",
          "geocodeAddressLine" : "..."
        },
        "signatureSvg" : "...",
        "signaturePicture" : "...",
        "data" : {
          "property1" : "...",
          "property2" : "..."
        },
        "pictures" : [ "...", "..." ],
        "simulationId" : "...",
        "simulationDayId" : "...",
        "timeWindowEnd" : 12345,
        "timeWindowSmsId" : "...",
        "timeWindowSmsStatus" : "...",
        "timeWindowStart" : 12345,
        "apprauchSmsId" : "...",
        "apprauchSmsStatus" : "...",
        "feedbackSmsId" : "...",
        "feedbackSmsStatus" : "...",
        "id" : "..."
      } ],
      "lastKnownPosition" : [ {
        "date" : 12345,
```

```
        "lon" : 12345.0,
        "lat" : 12345.0,
        "accuracy" : 12345.0,
        "privateLife" : true,
        "gpsStatus" : "0",
        "batteryLevel" : 12345,
        "id" : "..."
      }, {
        "date" : 12345,
        "lon" : 12345.0,
        "lat" : 12345.0,
        "accuracy" : 12345.0,
        "privateLife" : true,
        "gpsStatus" : "0",
        "batteryLevel" : 12345,
        "id" : "..."
      } ],
      "message" : "...",
      "status" : "OK"
    }
  }
```


```
GET /toursolver/gatewayToken
```

Description

Get a gateway token.

Get a gateway token that can be used to get connected to Toursolver GUI through the gateway.

Use this token to get connected to TsCloud GUI by passing it in the query like this : https://app.geoconcept.com/ToursolverCloud/ts/login?token=xxx

Note : this token will not work with the old (geoservices) Toursolver url.

Parameters

Type	Name	Description	Schema	Default
Query	login *optional*	in multi-user context, you can specify the login. If not specified, the default user will be used.	string	

Responses

HTTP Code	Description	Schema
200	Success	json_LoginTokenResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200


```
{
  "application/json" : {
```

```
    "token" : "...",
    "validUntil" : 12345,
    "message" : "...",
    "status" : "ERROR"
  }
}
```

```
GET /toursolver/logintoken
```

Description

[DEPRECATED] Get a login token.

Get a login token that can be used to get connected to Toursolver GUI. This token has a 30s life time.

Use this token to get connected to TsCloud GUI by passing it in the query like this : https://geoservices.geoconcept.com/ToursolverCloud/ts/login?token=xxx

This method is deprecated and will be removed in the future, it works only with the old (geoservices) Toursolver url, you should use gatewayToken instead.

Parameters

Type	Name	Description	Schema	Default
Query	login *optional*	in multi-user context, you can specify the login. If not specified, the default user will be used.	string	

Responses

HTTP Code	Description	Schema
200	Success	json_LoginTokenResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200

```
{
  "application/json" : {
    "token" : "...",
    "validUntil" : 12345,
    "message" : "...",
    "status" : "ERROR"
  }
}
```

```
POST /toursolver/optimize
```

Description

Start Optimization.

Launch Toursolver planning optimization : starts the planning process on Orders and Resources data.

Use this method to plan tours. The process consists in assigning Orders object elements to Resources object on a way that respects the constraints and minimizes the cost of the computed planning. This method is composed of 2 processes: a data checking and optimizing process.

Parameters

Type	Name	Description	Schema	Default
Body	body *optional*	Optimization request with all constraints	json_OptimizeRequest	

Responses

HTTP Code	Description	Schema
201	Success	json_OptimizeResult

Consumes

- `application/xml`
- `application/json`
- `text/xml`

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP request

=.Request body

```
{
  "depots" : [ {
    "id" : "...",
    "openingDaysList" : [ "...", "..." ],
    "timeWindows" : [ {
      "beginTime" : "08:00",
      "endTime" : "18:00"
    }, {
      "beginTime" : "08:00",
      "endTime" : "18:00"
    } ],
    "availability" : true,
    "resourceNames" : "...",
    "excludeResources" : "...",
    "travelTimeModifier" : {
      "offset" : "...",
      "value" : 12345.0,
      "length" : "..."
    },
    "fixedLoadingDuration" : "...",
    "loadingDurationPerUnit" : "...",
    "priority" : 12345,
    "requiredProducts" : "...",
    "allProductsRequired" : true,
```

```
      "deliveryQuantities" : [ 12345, 12345 ],
      "pickupQuantities" : [ 12345, 12345 ],
      "x" : 12345.0,
      "y" : 12345.0
    }, {
      "id" : "...",
      "openingDaysList" : [ "...", "..." ],
      "timeWindows" : [ {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      }, {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      } ],
      "availability" : true,
      "resourceNames" : "...",
      "excludeResources" : "...",
      "travelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "fixedLoadingDuration" : "...",
      "loadingDurationPerUnit" : "...",
      "priority" : 12345,
      "requiredProducts" : "...",
      "allProductsRequired" : true,
      "deliveryQuantities" : [ 12345, 12345 ],
      "pickupQuantities" : [ 12345, 12345 ],
      "x" : 12345.0,
      "y" : 12345.0
    } ],
    "orders" : [ {
      "allSkillsRequired" : true,
      "assignResources" : [ "...", "..." ],
      "courierPenalty" : 12345.0,
      "customDataMap" : {
        "property1" : "...",
        "property2" : "..."
      },
      "delayPenaltyPerHour" : 12345.0,
      "excludeResources" : [ "...", "..." ],
      "fixedVisitDuration" : "...",
      "frequency" : "...",
      "id" : "...",
      "minDuration" : "...",
      "minPartDuration" : "...",
      "punctuality" : 12345,
      "quantities" : [ 12345.0, 12345.0 ],
      "requiredSkills" : [ "...", "..." ],
      "resourceCompatibility" : 12345.0,
      "sequenceNumber" : 12345,
      "timeWindows" : [ {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      }, {
        "beginTime" : "08:00",
        "endTime" : "18:00"
      } ],
      "travelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
```

```
    },
    "type" : 12345,
    "unloadingDurationPerUnit" : "...",
    "x" : 12345.0,
    "y" : 12345.0,
    "active" : true,
    "wholeVisitInTimeWindow" : true,
    "label" : "...",
    "evaluationInfos" : {
      "orderOriginalResourceId" : "...",
      "orderOriginalVisitDay" : "...",
      "orderPosition" : 12345
    },
    "possibleVisitDaysList" : [ "...", "..." ],
    "tsOrderMaximumSpacing" : 12345,
    "tsOrderMinimumSpacing" : 12345,
    "tsOrderLastVisit" : 12345,
    "customerId" : "...",
    "email" : "...",
    "phone" : "...",
    "tsOrderBefore" : "...",
    "tsOrderBeforeMaxTimeSpacing" : "...",
    "tsOrderBeforeMinTimeSpacing" : "...",
    "getNotifications" : true,
    "tsOrderFixed" : true
  }, {
    "allSkillsRequired" : true,
    "assignResources" : [ "...", "..." ],
    "courierPenalty" : 12345.0,
    "customDataMap" : {
      "property1" : "...",
      "property2" : "..."
    },
    "delayPenaltyPerHour" : 12345.0,
    "excludeResources" : [ "...", "..." ],
    "fixedVisitDuration" : "...",
    "frequency" : "...",
    "id" : "...",
    "minDuration" : "...",
    "minPartDuration" : "...",
    "punctuality" : 12345,
    "quantities" : [ 12345.0, 12345.0 ],
    "requiredSkills" : [ "...", "..." ],
    "resourceCompatibility" : 12345.0,
    "sequenceNumber" : 12345,
    "timeWindows" : [ {
      "beginTime" : "08:00",
      "endTime" : "18:00"
    }, {
      "beginTime" : "08:00",
      "endTime" : "18:00"
    } ],
    "travelTimeModifier" : {
      "offset" : "...",
      "value" : 12345.0,
      "length" : "..."
    },
    "type" : 12345,
    "unloadingDurationPerUnit" : "...",
    "x" : 12345.0,
    "y" : 12345.0,
    "active" : true,
    "wholeVisitInTimeWindow" : true,
```

```
      "label" : "...",
      "evaluationInfos" : {
        "orderOriginalResourceId" : "...",
        "orderOriginalVisitDay" : "...",
        "orderPosition" : 12345
      },
      "possibleVisitDaysList" : [ "...", "..." ],
      "tsOrderMaximumSpacing" : 12345,
      "tsOrderMinimumSpacing" : 12345,
      "tsOrderLastVisit" : 12345,
      "customerId" : "...",
      "email" : "...",
      "phone" : "...",
      "tsOrderBefore" : "...",
      "tsOrderBeforeMaxTimeSpacing" : "...",
      "tsOrderBeforeMinTimeSpacing" : "...",
      "getNotifications" : true,
      "tsOrderFixed" : true
    } ],
    "resources" : [ {
      "available" : true,
      "avgConsumption" : 8.0,
      "briefingDuration" : "00:20:00",
      "capacities" : [ 12.0, 12.0 ],
      "providedSkills" : "...",
      "customDataMap" : {
        "property1" : "...",
        "property2" : "..."
      },
      "dailyWorkTime" : "...",
      "debriefingDuration" : "...",
      "driveRestAtCustomer" : true,
      "driveRestAtDepot" : true,
      "fixedLoadingDuration" : "...",
      "fuelType" : 12345,
      "id" : "...",
      "legalDailyDriveDuration" : "...",
      "legalDailyRestDuration" : "...",
      "legalDriveRestDuration" : "...",
      "legalMaxDriveDuration" : "00:30:00",
      "legalMinRestDuration" : "...",
      "loadBeforeDeparture" : true,
      "loadingDurationPerUnit" : "...",
      "loadOnReturn" : true,
      "lunch" : {
        "start" : "12:30",
        "end" : "14:00",
        "duration" : "60"
      },
      "maxNightsOutPerJourney" : 12345,
      "minDriveDuration" : "...",
      "nightPenalty" : 12345.0,
      "nonUsePenalty" : 12345.0,
      "openStart" : true,
      "openStop" : true,
      "optimumStartTime" : true,
      "overnightMinDriving" : "...",
      "overtimeDurations" : [ { }, { } ],
      "overtimePenalties" : [ 12345.0, 12345.0 ],
      "payWholeDay" : true,
      "penaltyPerVisit" : 12345.0,
      "speedAdjustment" : 12345,
      "startTravelTimeModifier" : {
```

```
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "stopTravelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "extraTravelPenalties" : [ {
        "distance" : 12345,
        "penalty" : 12345.0
      }, {
        "distance" : 12345,
        "penalty" : 12345.0
      } ],
      "travelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "usePenalty" : 12345.0,
      "weeklyWorkTime" : "...",
      "workEndTime" : "...",
      "workingDays" : "...",
      "workPenalty" : 12345.0,
      "workStartTime" : "...",
      "startX" : 12345.0,
      "endX" : 12345.0,
      "startY" : 12345.0,
      "endY" : 12345.0,
      "travelPenalty" : 12345.0,
      "minimumQuantity" : 12345.0,
      "fixedUnloadingDuration" : "...",
      "unloadingDurationPerUnit" : "...",
      "maximumReloads" : 12345,
      "maximumReloadsPenalty" : 12345.0,
      "noReload" : true,
      "otherWorkStartTimes" : "...",
      "otherWorkEndTimes" : "...",
      "otherWorkingDays" : [ "...", "..." ],
      "providedProducts" : "...",
      "useInPlanningPenalty" : 12345.0,
      "maximumDistance" : 12345,
      "maximumVisits" : 12345,
      "mobileLogin" : "...",
      "useAllCapacities" : true,
      "globalCapacity" : 12345.0,
      "additionalCostOrderCustomDataName" : "...",
      "additionalCostOperator" : "MAX",
      "additionalCosts" : [ {
        "type" : "...",
        "value" : 12345.0
      }, {
        "type" : "...",
        "value" : 12345.0
      } ],
      "openTimeStart" : true,
      "openDistanceStart" : true,
      "openTimeStop" : true,
      "openDistanceStop" : true,
      "tomTomWebFleetEnabled" : true,
      "tomTomWebFleetIdentifier" : "...",
```

```
      "vehicleCode" : "...",
      "fuelCode" : "..."
    }, {
      "available" : true,
      "avgConsumption" : 8.0,
      "briefingDuration" : "00:20:00",
      "capacities" : [ 12.0, 12.0 ],
      "providedSkills" : "...",
      "customDataMap" : {
        "property1" : "...",
        "property2" : "..."
      },
      "dailyWorkTime" : "...",
      "debriefingDuration" : "...",
      "driveRestAtCustomer" : true,
      "driveRestAtDepot" : true,
      "fixedLoadingDuration" : "...",
      "fuelType" : 12345,
      "id" : "...",
      "legalDailyDriveDuration" : "...",
      "legalDailyRestDuration" : "...",
      "legalDriveRestDuration" : "...",
      "legalMaxDriveDuration" : "00:30:00",
      "legalMinRestDuration" : "...",
      "loadBeforeDeparture" : true,
      "loadingDurationPerUnit" : "...",
      "loadOnReturn" : true,
      "lunch" : {
        "start" : "12:30",
        "end" : "14:00",
        "duration" : "60"
      },
      "maxNightsOutPerJourney" : 12345,
      "minDriveDuration" : "...",
      "nightPenalty" : 12345.0,
      "nonUsePenalty" : 12345.0,
      "openStart" : true,
      "openStop" : true,
      "optimumStartTime" : true,
      "overnightMinDriving" : "...",
      "overtimeDurations" : [ { }, { } ],
      "overtimePenalties" : [ 12345.0, 12345.0 ],
      "payWholeDay" : true,
      "penaltyPerVisit" : 12345.0,
      "speedAdjustment" : 12345,
      "startTravelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "stopTravelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "extraTravelPenalties" : [ {
        "distance" : 12345,
        "penalty" : 12345.0
      }, {
        "distance" : 12345,
        "penalty" : 12345.0
      } ],
      "travelTimeModifier" : {
```

```
      "offset" : "...",
      "value" : 12345.0,
      "length" : "..."
    },
    "usePenalty" : 12345.0,
    "weeklyWorkTime" : "...",
    "workEndTime" : "...",
    "workingDays" : "...",
    "workPenalty" : 12345.0,
    "workStartTime" : "...",
    "startX" : 12345.0,
    "endX" : 12345.0,
    "startY" : 12345.0,
    "endY" : 12345.0,
    "travelPenalty" : 12345.0,
    "minimumQuantity" : 12345.0,
    "fixedUnloadingDuration" : "...",
    "unloadingDurationPerUnit" : "...",
    "maximumReloads" : 12345,
    "maximumReloadsPenalty" : 12345.0,
    "noReload" : true,
    "otherWorkStartTimes" : "...",
    "otherWorkEndTimes" : "...",
    "otherWorkingDays" : [ "...", "..." ],
    "providedProducts" : "...",
    "useInPlanningPenalty" : 12345.0,
    "maximumDistance" : 12345,
    "maximumVisits" : 12345,
    "mobileLogin" : "...",
    "useAllCapacities" : true,
    "globalCapacity" : 12345.0,
    "additionalCostOrderCustomDataName" : "...",
    "additionalCostOperator" : "AVERAGE",
    "additionalCosts" : [ {
      "type" : "...",
      "value" : 12345.0
    }, {
      "type" : "...",
      "value" : 12345.0
    } ],
    "openTimeStart" : true,
    "openDistanceStart" : true,
    "openTimeStop" : true,
    "openDistanceStop" : true,
    "tomTomWebFleetEnabled" : true,
    "tomTomWebFleetIdentifier" : "...",
    "vehicleCode" : "...",
    "fuelCode" : "..."
  } ],
  "options" : {
    "evaluation" : true,
    "startFromEvaluationInfo" : true,
    "maxOptimDuration" : "...",
    "reloadDuration" : "...",
    "noReload" : true,
    "distanceType" : "METERS",
    "teamId" : "...",
    "sendResultToWebhook" : "ORDERS",
    "countVisitCostOnceIfSameLocation" : true,
    "countDepotsInDeliveryCost" : "FIRST",
    "excludeVisitCostIfMaxAdditionalCost" : true,
    "routingMethod" : "DISTANCE",
    "allowToll" : true,
```

```
    "allowTunnel" : true,
    "allowBridge" : true,
    "vehicleCode" : "...",
    "fuelCode" : "...",
    "speedPattern" : "...",
    "useForbiddenTransitAreas" : true,
    "useOTSolver" : true,
    "balanceType" : "HOURS",
    "balanceValue" : 12345
  },
  "countryCode" : "...",
  "simulationName" : "...",
  "language" : "...",
  "beginDate" : 12345,
  "userLogin" : "...",
  "organization" : "..."
}
```

Example HTTP response

=.Response 201

```
{
  "application/json" : {
    "taskId" : "...",
    "message" : "...",
    "status" : "OK"
  }
}
```

```
GET /toursolver/resources
```

Description

Get known resources

Get resources defined in ToursolverCloud GUI.

Parameters

Type	Name	Description	Schema	Default
Query	resourceName *optional*	if a resource name is specified (team name will be ignored), only one resource will be returned.	string	
Query	teamName *optional*	if a team name is specified, only the resources of this team will returned	string	

Responses

HTTP Code	Description	Schema
200	Success	json_ResourcesResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200


```
{
  "application/json" : {
    "resources" : [ {
      "available" : true,
      "avgConsumption" : 8.0,
      "briefingDuration" : "00:20:00",
      "capacities" : [ 12.0, 12.0 ],
      "providedSkills" : "...",
      "customDataMap" : {
        "property1" : "...",
        "property2" : "..."
      },
      "dailyWorkTime" : "...",
      "debriefingDuration" : "...",
      "driveRestAtCustomer" : true,
      "driveRestAtDepot" : true,
      "fixedLoadingDuration" : "...",
      "fuelType" : 12345,
      "id" : "...",
      "legalDailyDriveDuration" : "...",
      "legalDailyRestDuration" : "...",
      "legalDriveRestDuration" : "...",
      "legalMaxDriveDuration" : "00:30:00",
      "legalMinRestDuration" : "...",
      "loadBeforeDeparture" : true,
      "loadingDurationPerUnit" : "...",
      "loadOnReturn" : true,
      "lunch" : {
        "start" : "12:30",
        "end" : "14:00",
        "duration" : "60"
      },
      "maxNightsOutPerJourney" : 12345,
      "minDriveDuration" : "...",
      "nightPenalty" : 12345.0,
      "nonUsePenalty" : 12345.0,
      "openStart" : true,
      "openStop" : true,
      "optimumStartTime" : true,
      "overnightMinDriving" : "...",
      "overtimeDurations" : [ { }, { } ],
      "overtimePenalties" : [ 12345.0, 12345.0 ],
      "payWholeDay" : true,
      "penaltyPerVisit" : 12345.0,
      "speedAdjustment" : 12345,
      "startTravelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "stopTravelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "extraTravelPenalties" : [ {
        "distance" : 12345,
        "penalty" : 12345.0
      }, {
        "distance" : 12345,
```

```
        "penalty" : 12345.0
      } ],
      "travelTimeModifier" : {
        "offset" : "...",
        "value" : 12345.0,
        "length" : "..."
      },
      "usePenalty" : 12345.0,
      "weeklyWorkTime" : "...",
      "workEndTime" : "...",
      "workingDays" : "...",
      "workPenalty" : 12345.0,
      "workStartTime" : "...",
      "startX" : 12345.0,
      "endX" : 12345.0,
      "startY" : 12345.0,
      "endY" : 12345.0,
      "travelPenalty" : 12345.0,
      "minimumQuantity" : 12345.0,
      "fixedUnloadingDuration" : "...",
      "unloadingDurationPerUnit" : "...",
      "maximumReloads" : 12345,
      "maximumReloadsPenalty" : 12345.0,
      "noReload" : true,
      "otherWorkStartTimes" : "...",
      "otherWorkEndTimes" : "...",
      "otherWorkingDays" : [ "...", "..." ],
      "providedProducts" : "...",
      "useInPlanningPenalty" : 12345.0,
      "maximumDistance" : 12345,
      "maximumVisits" : 12345,
      "mobileLogin" : "...",
      "useAllCapacities" : true,
      "globalCapacity" : 12345.0,
      "additionalCostOrderCustomDataName" : "...",
      "additionalCostOperator" : "MAX",
      "additionalCosts" : [ {
        "type" : "...",
        "value" : 12345.0
      }, {
        "type" : "...",
        "value" : 12345.0
      } ],
      "openTimeStart" : true,
      "openDistanceStart" : true,
      "openTimeStop" : true,
      "openDistanceStop" : true,
      "tomTomWebFleetEnabled" : true,
      "tomTomWebFleetIdentifier" : "...",
      "vehicleCode" : "...",
      "fuelCode" : "..."
    }, {
      "available" : true,
      "avgConsumption" : 8.0,
      "briefingDuration" : "00:20:00",
      "capacities" : [ 12.0, 12.0 ],
      "providedSkills" : "...",
      "customDataMap" : {
        "property1" : "...",
        "property2" : "..."
      },
      "dailyWorkTime" : "...",
      "debriefingDuration" : "...",
```

```
        "driveRestAtCustomer" : true,
        "driveRestAtDepot" : true,
        "fixedLoadingDuration" : "...",
        "fuelType" : 12345,
        "id" : "...",
        "legalDailyDriveDuration" : "...",
        "legalDailyRestDuration" : "...",
        "legalDriveRestDuration" : "...",
        "legalMaxDriveDuration" : "00:30:00",
        "legalMinRestDuration" : "...",
        "loadBeforeDeparture" : true,
        "loadingDurationPerUnit" : "...",
        "loadOnReturn" : true,
        "lunch" : {
          "start" : "12:30",
          "end" : "14:00",
          "duration" : "60"
        },
        "maxNightsOutPerJourney" : 12345,
        "minDriveDuration" : "...",
        "nightPenalty" : 12345.0,
        "nonUsePenalty" : 12345.0,
        "openStart" : true,
        "openStop" : true,
        "optimumStartTime" : true,
        "overnightMinDriving" : "...",
        "overtimeDurations" : [ { }, { } ],
        "overtimePenalties" : [ 12345.0, 12345.0 ],
        "payWholeDay" : true,
        "penaltyPerVisit" : 12345.0,
        "speedAdjustment" : 12345,
        "startTravelTimeModifier" : {
          "offset" : "...",
          "value" : 12345.0,
          "length" : "..."
        },
        "stopTravelTimeModifier" : {
          "offset" : "...",
          "value" : 12345.0,
          "length" : "..."
        },
        "extraTravelPenalties" : [ {
          "distance" : 12345,
          "penalty" : 12345.0
        }, {
          "distance" : 12345,
          "penalty" : 12345.0
        } ],
        "travelTimeModifier" : {
          "offset" : "...",
          "value" : 12345.0,
          "length" : "..."
        },
        "usePenalty" : 12345.0,
        "weeklyWorkTime" : "...",
        "workEndTime" : "...",
        "workingDays" : "...",
        "workPenalty" : 12345.0,
        "workStartTime" : "...",
        "startX" : 12345.0,
        "endX" : 12345.0,
        "startY" : 12345.0,
        "endY" : 12345.0,
```

```
      "travelPenalty" : 12345.0,
      "minimumQuantity" : 12345.0,
      "fixedUnloadingDuration" : "...",
      "unloadingDurationPerUnit" : "...",
      "maximumReloads" : 12345,
      "maximumReloadsPenalty" : 12345.0,
      "noReload" : true,
      "otherWorkStartTimes" : "...",
      "otherWorkEndTimes" : "...",
      "otherWorkingDays" : [ "...", "..." ],
      "providedProducts" : "...",
      "useInPlanningPenalty" : 12345.0,
      "maximumDistance" : 12345,
      "maximumVisits" : 12345,
      "mobileLogin" : "...",
      "useAllCapacities" : true,
      "globalCapacity" : 12345.0,
      "additionalCostOrderCustomDataName" : "...",
      "additionalCostOperator" : "SUM",
      "additionalCosts" : [ {
        "type" : "...",
        "value" : 12345.0
      }, {
        "type" : "...",
        "value" : 12345.0
      } ],
      "openTimeStart" : true,
      "openDistanceStart" : true,
      "openTimeStop" : true,
      "openDistanceStop" : true,
      "tomTomWebFleetEnabled" : true,
      "tomTomWebFleetIdentifier" : "...",
      "vehicleCode" : "...",
      "fuelCode" : "..."
    } ],
    "message" : "...",
    "status" : "ERROR"
  }
}
```


```
GET /toursolver/result
```

Description

Get optimization result.

Get the result of planning optimization. Status of the optimization must be terminated.

Parameters

Type	Name	Description	Schema	Default
Query	taskId *optional*	the id of optimization task, as returned by the optimize service	string	

Responses

HTTP Code	Description	Schema
200	Success	json_OptimResultResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200


```
{
  "application/json" : {
    "taskId" : "...",
    "plannedOrders" : [ {
      "dayId" : "...",
      "stopPosition" : 12345,
      "stopY" : 12345.0,
      "stopX" : 12345.0,
      "stopId" : "...",
      "stopType" : 12345,
      "stopDriveTime" : "...",
      "stopStartTime" : "...",
      "stopDuration" : "...",
      "stopStatus" : 12345,
      "stopDriveDistance" : 12345,
      "stopElapsedDistance" : 12345,
      "resourceId" : "..."
    }, {
      "dayId" : "...",
      "stopPosition" : 12345,
      "stopY" : 12345.0,
      "stopX" : 12345.0,
      "stopId" : "...",
      "stopType" : 12345,
      "stopDriveTime" : "...",
      "stopStartTime" : "...",
      "stopDuration" : "...",
      "stopStatus" : 12345,
      "stopDriveDistance" : 12345,
      "stopElapsedDistance" : 12345,
      "resourceId" : "..."
    } ],
    "unplannedOrders" : [ {
      "stopID" : "...",
      "reason" : "..."
    }, {
      "stopID" : "...",
      "reason" : "..."
    } ],
    "warnings" : [ {
      "objectType" : "...",
      "id" : "...",
      "constraint" : 12345,
      "value" : "...",
      "message" : "...",
      "messageId" : 12345,
      "i18nMessageCode" : "...",
      "constraintName" : "..."
    }, {
      "objectType" : "...",
      "id" : "...",
      "constraint" : 12345,
```

```
      "value" : "...",
      "message" : "...",
      "messageId" : 12345,
      "i18nMessageCode" : "...",
      "constraintName" : "..."
    } ],
    "simulationId" : "...",
    "message" : "...",
    "status" : "ERROR"
  }
}
```

```
GET /toursolver/simulation
```

Description

Get simulation by ID

Get simulation launched in ToursolverCloud (GUI).

Parameters

Type	Name	Description	Schema	Default
Query	simulationId *optional*		string	

Responses

HTTP Code	Description	Schema
200	Success	json_SimulationResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200

```
{
  "application/json" : {
    "simulation" : {
      "depots" : [ {
        "id" : "...",
        "openingDaysList" : [ "...", "..." ],
        "timeWindows" : [ { }, { } ],
        "availability" : true,
        "resourceNames" : "...",
        "excludeResources" : "...",
        "travelTimeModifier" : { },
        "fixedLoadingDuration" : "...",
        "loadingDurationPerUnit" : "...",
        "priority" : 12345,
        "requiredProducts" : "...",
        "allProductsRequired" : true,
        "deliveryQuantities" : [ 12345, 12345 ],
        "pickupQuantities" : [ 12345, 12345 ],
```

```
          "x" : 12345.0,
          "y" : 12345.0
        }, {
          "id" : "...",
          "openingDaysList" : [ "...", "..." ],
          "timeWindows" : [ { }, { } ],
          "availability" : true,
          "resourceNames" : "...",
          "excludeResources" : "...",
          "travelTimeModifier" : { },
          "fixedLoadingDuration" : "...",
          "loadingDurationPerUnit" : "...",
          "priority" : 12345,
          "requiredProducts" : "...",
          "allProductsRequired" : true,
          "deliveryQuantities" : [ 12345, 12345 ],
          "pickupQuantities" : [ 12345, 12345 ],
          "x" : 12345.0,
          "y" : 12345.0
        } ],
        "resources" : [ {
          "available" : true,
          "avgConsumption" : 8.0,
          "briefingDuration" : "00:20:00",
          "capacities" : [ 12.0, 12.0 ],
          "providedSkills" : "...",
          "customDataMap" : {
            "property1" : "...",
            "property2" : "..."
          },
          "dailyWorkTime" : "...",
          "debriefingDuration" : "...",
          "driveRestAtCustomer" : true,
          "driveRestAtDepot" : true,
          "fixedLoadingDuration" : "...",
          "fuelType" : 12345,
          "id" : "...",
          "legalDailyDriveDuration" : "...",
          "legalDailyRestDuration" : "...",
          "legalDriveRestDuration" : "...",
          "legalMaxDriveDuration" : "00:30:00",
          "legalMinRestDuration" : "...",
          "loadBeforeDeparture" : true,
          "loadingDurationPerUnit" : "...",
          "loadOnReturn" : true,
          "lunch" : { },
          "maxNightsOutPerJourney" : 12345,
          "minDriveDuration" : "...",
          "nightPenalty" : 12345.0,
          "nonUsePenalty" : 12345.0,
          "openStart" : true,
          "openStop" : true,
          "optimumStartTime" : true,
          "overnightMinDriving" : "...",
          "overtimeDurations" : [ { }, { } ],
          "overtimePenalties" : [ 12345.0, 12345.0 ],
          "payWholeDay" : true,
          "penaltyPerVisit" : 12345.0,
          "speedAdjustment" : 12345,
          "startTravelTimeModifier" : { },
          "stopTravelTimeModifier" : { },
          "extraTravelPenalties" : [ { }, { } ],
          "travelTimeModifier" : { },
```

```
      "usePenalty" : 12345.0,
      "weeklyWorkTime" : "...",
      "workEndTime" : "...",
      "workingDays" : "...",
      "workPenalty" : 12345.0,
      "workStartTime" : "...",
      "startX" : 12345.0,
      "endX" : 12345.0,
      "startY" : 12345.0,
      "endY" : 12345.0,
      "travelPenalty" : 12345.0,
      "minimumQuantity" : 12345.0,
      "fixedUnloadingDuration" : "...",
      "unloadingDurationPerUnit" : "...",
      "maximumReloads" : 12345,
      "maximumReloadsPenalty" : 12345.0,
      "noReload" : true,
      "otherWorkStartTimes" : "...",
      "otherWorkEndTimes" : "...",
      "otherWorkingDays" : [ "...", "..." ],
      "providedProducts" : "...",
      "useInPlanningPenalty" : 12345.0,
      "maximumDistance" : 12345,
      "maximumVisits" : 12345,
      "mobileLogin" : "...",
      "useAllCapacities" : true,
      "globalCapacity" : 12345.0,
      "additionalCostOrderCustomDataName" : "...",
      "additionalCostOperator" : "MIN",
      "additionalCosts" : [ { }, { } ],
      "openTimeStart" : true,
      "openDistanceStart" : true,
      "openTimeStop" : true,
      "openDistanceStop" : true,
      "tomTomWebFleetEnabled" : true,
      "tomTomWebFleetIdentifier" : "...",
      "vehicleCode" : "...",
      "fuelCode" : "..."
    }, {
      "available" : true,
      "avgConsumption" : 8.0,
      "briefingDuration" : "00:20:00",
      "capacities" : [ 12.0, 12.0 ],
      "providedSkills" : "...",
      "customDataMap" : {
        "property1" : "...",
        "property2" : "..."
      },
      "dailyWorkTime" : "...",
      "debriefingDuration" : "...",
      "driveRestAtCustomer" : true,
      "driveRestAtDepot" : true,
      "fixedLoadingDuration" : "...",
      "fuelType" : 12345,
      "id" : "...",
      "legalDailyDriveDuration" : "...",
      "legalDailyRestDuration" : "...",
      "legalDriveRestDuration" : "...",
      "legalMaxDriveDuration" : "00:30:00",
      "legalMinRestDuration" : "...",
      "loadBeforeDeparture" : true,
      "loadingDurationPerUnit" : "...",
      "loadOnReturn" : true,
```

```
          "lunch" : { },
          "maxNightsOutPerJourney" : 12345,
          "minDriveDuration" : "...",
          "nightPenalty" : 12345.0,
          "nonUsePenalty" : 12345.0,
          "openStart" : true,
          "openStop" : true,
          "optimumStartTime" : true,
          "overnightMinDriving" : "...",
          "overtimeDurations" : [ { }, { } ],
          "overtimePenalties" : [ 12345.0, 12345.0 ],
          "payWholeDay" : true,
          "penaltyPerVisit" : 12345.0,
          "speedAdjustment" : 12345,
          "startTravelTimeModifier" : { },
          "stopTravelTimeModifier" : { },
          "extraTravelPenalties" : [ { }, { } ],
          "travelTimeModifier" : { },
          "usePenalty" : 12345.0,
          "weeklyWorkTime" : "...",
          "workEndTime" : "...",
          "workingDays" : "...",
          "workPenalty" : 12345.0,
          "workStartTime" : "...",
          "startX" : 12345.0,
          "endX" : 12345.0,
          "startY" : 12345.0,
          "endY" : 12345.0,
          "travelPenalty" : 12345.0,
          "minimumQuantity" : 12345.0,
          "fixedUnloadingDuration" : "...",
          "unloadingDurationPerUnit" : "...",
          "maximumReloads" : 12345,
          "maximumReloadsPenalty" : 12345.0,
          "noReload" : true,
          "otherWorkStartTimes" : "...",
          "otherWorkEndTimes" : "...",
          "otherWorkingDays" : [ "...", "..." ],
          "providedProducts" : "...",
          "useInPlanningPenalty" : 12345.0,
          "maximumDistance" : 12345,
          "maximumVisits" : 12345,
          "mobileLogin" : "...",
          "useAllCapacities" : true,
          "globalCapacity" : 12345.0,
          "additionalCostOrderCustomDataName" : "...",
          "additionalCostOperator" : "SUM",
          "additionalCosts" : [ { }, { } ],
          "openTimeStart" : true,
          "openDistanceStart" : true,
          "openTimeStop" : true,
          "openDistanceStop" : true,
          "tomTomWebFleetEnabled" : true,
          "tomTomWebFleetIdentifier" : "...",
          "vehicleCode" : "...",
          "fuelCode" : "..."
        } ],
        "orders" : [ {
          "allSkillsRequired" : true,
          "assignResources" : [ "...", "..." ],
          "courierPenalty" : 12345.0,
          "customDataMap" : {
            "property1" : "...",
```

```
      "property2" : "..."
    },
    "delayPenaltyPerHour" : 12345.0,
    "excludeResources" : [ "...", "..." ],
    "fixedVisitDuration" : "...",
    "frequency" : "...",
    "id" : "...",
    "minDuration" : "...",
    "minPartDuration" : "...",
    "punctuality" : 12345,
    "quantities" : [ 12345.0, 12345.0 ],
    "requiredSkills" : [ "...", "..." ],
    "resourceCompatibility" : 12345.0,
    "sequenceNumber" : 12345,
    "timeWindows" : [ { }, { } ],
    "travelTimeModifier" : { },
    "type" : 12345,
    "unloadingDurationPerUnit" : "...",
    "x" : 12345.0,
    "y" : 12345.0,
    "active" : true,
    "wholeVisitInTimeWindow" : true,
    "label" : "...",
    "evaluationInfos" : { },
    "possibleVisitDaysList" : [ "...", "..." ],
    "tsOrderMaximumSpacing" : 12345,
    "tsOrderMinimumSpacing" : 12345,
    "tsOrderLastVisit" : 12345,
    "customerId" : "...",
    "email" : "...",
    "phone" : "...",
    "tsOrderBefore" : "...",
    "tsOrderBeforeMaxTimeSpacing" : "...",
    "tsOrderBeforeMinTimeSpacing" : "...",
    "getNotifications" : true,
    "tsOrderFixed" : true
  }, {
    "allSkillsRequired" : true,
    "assignResources" : [ "...", "..." ],
    "courierPenalty" : 12345.0,
    "customDataMap" : {
      "property1" : "...",
      "property2" : "..."
    },
    "delayPenaltyPerHour" : 12345.0,
    "excludeResources" : [ "...", "..." ],
    "fixedVisitDuration" : "...",
    "frequency" : "...",
    "id" : "...",
    "minDuration" : "...",
    "minPartDuration" : "...",
    "punctuality" : 12345,
    "quantities" : [ 12345.0, 12345.0 ],
    "requiredSkills" : [ "...", "..." ],
    "resourceCompatibility" : 12345.0,
    "sequenceNumber" : 12345,
    "timeWindows" : [ { }, { } ],
    "travelTimeModifier" : { },
    "type" : 12345,
    "unloadingDurationPerUnit" : "...",
    "x" : 12345.0,
    "y" : 12345.0,
    "active" : true,
```

```
          "wholeVisitInTimeWindow" : true,
          "label" : "...",
          "evaluationInfos" : { },
          "possibleVisitDaysList" : [ "...", "..." ],
          "tsOrderMaximumSpacing" : 12345,
          "tsOrderMinimumSpacing" : 12345,
          "tsOrderLastVisit" : 12345,
          "customerId" : "...",
          "email" : "...",
          "phone" : "...",
          "tsOrderBefore" : "...",
          "tsOrderBeforeMaxTimeSpacing" : "...",
          "tsOrderBeforeMinTimeSpacing" : "...",
          "getNotifications" : true,
          "tsOrderFixed" : true
        } ],
        "nbQuantities" : 12345,
        "nbCapacities" : 12345,
        "nbTimeWindows" : 12345,
        "nbExtraTravelPenalties" : 12345,
        "depotProperties" : [ "...", "..." ],
        "resourceProperties" : [ "...", "..." ],
        "orderProperties" : [ "...", "..." ],
        "options" : {
          "evaluation" : true,
          "startFromEvaluationInfo" : true,
          "maxOptimDuration" : "...",
          "reloadDuration" : "...",
          "noReload" : true,
          "distanceType" : "METERS",
          "teamId" : "...",
          "sendResultToWebhook" : "ORDERS",
          "countVisitCostOnceIfSameLocation" : true,
          "countDepotsInDeliveryCost" : "NONE",
          "excludeVisitCostIfMaxAdditionalCost" : true,
          "routingMethod" : "DISTANCE",
          "allowToll" : true,
          "allowTunnel" : true,
          "allowBridge" : true,
          "vehicleCode" : "...",
          "fuelCode" : "...",
          "speedPattern" : "...",
          "useForbiddenTransitAreas" : true,
          "useOTSolver" : true,
          "balanceType" : "HOURS",
          "balanceValue" : 12345
        }
      },
      "message" : "...",
      "status" : "ERROR"
    }
  }
```


```
GET /toursolver/status
```

Description

Get optimization status.

This service allows to know the status of an optimization.

Parameters

Type	Name	Description	Schema	Default
Query	taskId *optional*	the id of optimization task, as returned by the optimize service	string	

Responses

HTTP Code	Description	Schema
200	Success	json_OptimStatusResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200


```
{
  "application/json" : {
    "optimizeStatus" : "geocoding",
    "startTime" : 12345,
    "currentCo2" : 12345.0,
    "currentCost" : 12345,
    "currentCourierCost" : 12345,
    "currentDeliveredQuantity" : 12345.0,
    "currentDeliveryCost" : 12345,
    "currentDriveCost" : 12345,
    "currentDriveDistance" : 12345,
    "currentDriveTime" : 12345,
    "currentFixedCost" : 12345,
    "currentLateTime" : 12345,
    "currentNightsCost" : 12345,
    "currentOverWorkCost" : 12345,
    "currentOverWorkTime" : 12345,
    "currentPickUpQuantity" : 12345.0,
    "currentRestTime" : 12345,
    "currentUnplannedVisits" : 12345,
    "currentWaitTime" : 12345,
    "currentWorkCost" : 12345,
    "currentWorkTime" : 12345,
    "initialCo2" : 12345.0,
    "initialCost" : 12345,
    "initialCourierCost" : 12345,
    "initialDeliveredQuantity" : 12345.0,
    "initialDeliveryCost" : 12345,
    "initialDriveCost" : 12345,
    "initialDriveDistance" : 12345,
    "initialDriveTime" : 12345,
    "initialFixedCost" : 12345,
    "initialLateTime" : 12345,
    "initialNightsCost" : 12345,
    "initialOverWorkCost" : 12345,
    "initialOverWorkTime" : 12345,
    "initialPickUpQuantity" : 12345.0,
    "initialRestTime" : 12345,
```

```
    "initialUnplannedVisits" : 12345,
    "initialWaitTime" : 12345,
    "initialWorkCost" : 12345,
    "initialWorkTime" : 12345,
    "mileageChartRemainingTime" : 12345,
    "initialOpenTourNumber" : 12345,
    "currentOpenTourNumber" : 12345,
    "subOptimNb" : 12345,
    "subOptimWaitingNb" : 12345,
    "subOptimRunningNb" : 12345,
    "subOptimFinishedNb" : 12345,
    "subOptimErrorNb" : 12345,
    "subOptimAbortedNb" : 12345,
    "simulationId" : "...",
    "currentVisitsNb" : 12345,
    "initialPlannedVisits" : 12345,
    "currentPlannedVisits" : 12345,
    "message" : "...",
    "status" : "ERROR"
  }
}
```

```
POST /toursolver/stop
```

Description

Stop optimization.

Interrupt optimization task.

This stop is asynchronous. You should use the related status function to know when the action is truly stopped.

Parameters

Type	Name	Description	Schema	Default
Query	taskId *optional*	the id of optimization task, as returned by the optimize service	string	

Responses

HTTP Code	Description	Schema
201	Success	json_OptimStopResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 201

```
{
  "application/json" : {
    "firstStopAsked" : 12345,
    "message" : "...",
```

```
    "status" : "OK"
  }
}
```

```
GET /toursolver/toursResult
```

Description

Get optimization result organized by tours.

Get the result of planning optimization organized by tours (one tour per resource per day). Status of the optimization must be terminated.

Parameters

Type	Name	Description	Schema	Default
Query	taskId *optional*	the id of optimization task, as returned by the optimize service	string	

Responses

HTTP Code	Description	Schema
200	Success	json_OptimToursResult

Produces

- `application/xml`
- `application/json`
- `text/xml`

Example HTTP response

=.Response 200

```
{
  "application/json" : {
    "taskId" : "...",
    "tours" : [ {
      "dayId" : "...",
      "resourceId" : "...",
      "travelDistance" : 12345,
      "travelDuration" : "...",
      "resourceCapacities" : [ 12345.0, 12345.0 ],
      "usedCapacities" : [ 12345.0, 12345.0 ],
      "deliveryCost" : 12345.0,
      "plannedOrders" : [ {
        "dayId" : "...",
        "stopPosition" : 12345,
        "stopY" : 12345.0,
        "stopX" : 12345.0,
        "stopId" : "...",
        "stopType" : 12345,
        "stopDriveTime" : "...",
        "stopStartTime" : "...",
        "stopDuration" : "...",
        "stopStatus" : 12345,
        "stopDriveDistance" : 12345,
        "stopElapsedDistance" : 12345,
```

```
        "resourceId" : "..."
      }, {
        "dayId" : "...",
        "stopPosition" : 12345,
        "stopY" : 12345.0,
        "stopX" : 12345.0,
        "stopId" : "...",
        "stopType" : 12345,
        "stopDriveTime" : "...",
        "stopStartTime" : "...",
        "stopDuration" : "...",
        "stopStatus" : 12345,
        "stopDriveDistance" : 12345,
        "stopElapsedDistance" : 12345,
        "resourceId" : "..."
      } ],
      "additionalCost" : 12345.0,
      "totalCost" : 12345.0,
      "reloadNb" : 12345
    }, {
      "dayId" : "...",
      "resourceId" : "...",
      "travelDistance" : 12345,
      "travelDuration" : "...",
      "resourceCapacities" : [ 12345.0, 12345.0 ],
      "usedCapacities" : [ 12345.0, 12345.0 ],
      "deliveryCost" : 12345.0,
      "plannedOrders" : [ {
        "dayId" : "...",
        "stopPosition" : 12345,
        "stopY" : 12345.0,
        "stopX" : 12345.0,
        "stopId" : "...",
        "stopType" : 12345,
        "stopDriveTime" : "...",
        "stopStartTime" : "...",
        "stopDuration" : "...",
        "stopStatus" : 12345,
        "stopDriveDistance" : 12345,
        "stopElapsedDistance" : 12345,
        "resourceId" : "..."
      }, {
        "dayId" : "...",
        "stopPosition" : 12345,
        "stopY" : 12345.0,
        "stopX" : 12345.0,
        "stopId" : "...",
        "stopType" : 12345,
        "stopDriveTime" : "...",
        "stopStartTime" : "...",
        "stopDuration" : "...",
        "stopStatus" : 12345,
        "stopDriveDistance" : 12345,
        "stopElapsedDistance" : 12345,
        "resourceId" : "..."
      } ],
      "additionalCost" : 12345.0,
      "totalCost" : 12345.0,
      "reloadNb" : 12345
    } ],
    "warnings" : [ {
      "objectType" : "...",
      "id" : "...",
```

```
      "constraint" : 12345,
      "value" : "...",
      "message" : "...",
      "messageId" : 12345,
      "i18nMessageCode" : "...",
      "constraintName" : "..."
    }, {
      "objectType" : "...",
      "id" : "...",
      "constraint" : 12345,
      "value" : "...",
      "message" : "...",
      "messageId" : 12345,
      "i18nMessageCode" : "...",
      "constraintName" : "..."
    } ],
    "unplannedOrders" : [ {
      "stopID" : "...",
      "reason" : "..."
    }, {
      "stopID" : "...",
      "reason" : "..."
    } ],
    "message" : "...",
    "status" : "ERROR"
  }
}
```

Javascript tutorial

You can easily use the TsCloud API using only javascript. It is very easy to perform ajax requests with any popular javascript framework. We will show in this chapter how to do it with JQuery.

First of all, you will need to include Jquery and tsCloudApi.js :

```
<html>
        <head>
                <title>TsCloud Api Tester</title>
                <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
                <script src="https://geoservices.geoconcept.com/ToursolverCloud/combo?toursolver/
tsCloudApi.js"></script>
        </head>
        <body>
        ...
        </body>
</html>
```

Depending on your geographical area, the ToursolverCloud url may be different.

You will then use de tsCloudApi object to communicate with TsCloud servers. Before sending any request, you must initialize the object with your api key :

```
<script>
        var myApiKey = "this is a fake key";
        tsCloudApi.init(myApiKey);
</script>
```

Launch an optimisation

To start an optimization, you will have use the optimize method of the tsCloudApi object :

```
<script>
        var myApiKey = "this is a fake key";
        tsCloudApi.init(myApiKey);
        tsCloudApi.optimize(optimRequest,startOptimSuccessHandler,errorHandler);
</script>
```

tsCloudApi performs asynchronous ajax requests. Therefore, you have to implement handler functions that will be called when the response will be available (startOptimSuccessHandler and errorHandler in the above example). optimRequest is the object containing the list of depots (optional), the list of resources, the list of orders and the optimization options (please refer to the definitions chapter for more details).

Here is a minimal example with one resource and two orders :

```
<script>
        var optimRequest = {
                "depots": [],
                "resources": [{
                        "x": 2.33683,
                        "y": 48.86255,
                        "id": "Robert",
                        "workStartTime": "08:00",
                        "workEndTime": "18:00",
                        "lunch": {
                                "start": "12:00",
                                "end": "14:00",
                                "duration": "01:00",
                        },
                        "workingDays": "1-5",
                        "capacities": [1000.0],
                        "workPenalty": 20.0,
                        "overtimePenalties": [0.0],
                        "travelPenalty": 2.0,
                }],
                "orders": [{
                        "id": "ORDER-1",
                        "label": "HOSPITAL",
                        "quantities": [2.0],
                        "fixedVisitDuration": "00:30",
                        "timeWindows": [{
                                "beginTime": "08:00",
                                "endTime": "10:00"
                        }],
                        "x": 2.348433,
                        "y": 48.853661
                }, {
                        "id": "ORDER-2",
                        "label": "EMERGENCIES",
                        "quantities": [2.0],
                        "fixedVisitDuration": "00:25",
                        "timeWindows": [{
                                "beginTime": "10:00",
                                "endTime": "12:00"
                        }],
                        "x": 2.347802,
                        "y": 48.854687
```

```
                }],
                ,
                "options": {
                        "vehicleCode": "car",
                        "stopTime": "00:01",
                        "stopTimeWithoutImprovement": "00:01",
                        "maxOptimDuration": "00:01",
                        "stopCondition": 1,
                        "reloadDuration": "00:45"
                },
                "countryCode":"FR"
        };

</script>
```

Here is how you could write your error handler :

```
<script>
function errorHandler(jqxhr,status,errorThrown) {
        if (jqxhr.status == 403) {
                console.log("Bad key code ?");
        } else if (jqxhr.status == 429) {
                console.log("oups, slow down please : " + jqxhr.statusText);
        } else if (jqxhr.status != 200) {
                console.log("sorry, an error occured : " + jqxhr.statusText);
                try {
                        if (jqxhr.responseText) {
                                var resp = JSON.parse(jqxhr.responseText);
                                if (resp.message) {
                                        console.log(resp.message);
                                }
                        }
                } catch (e) {
                        console.log("could not find details about the error");
                }
        }
}
</script>
```

If your api key is not valid, you will get a 403 http code. As explained in authentication chapter, you may received a 429 http code if you send to many requests in a short time. If you misspelled an object attribut or for any other technical problem, you may receive a 500 http code.

Here is how you could write your success handler, which must retrieve the taskId and start the status polling process that we will detail in the next section :

```
<script>
        var taskId = null;

        function startOptimSuccessHandler(data,status,xhr) {
                if (data.status == "OK") {
                        taskId = data.taskId;
                        console.log("optim launched. taskId is " + taskId);
                        getStatus();
                } else {
                        console.log(data.message);
                }
        }
</script>
```

Optimization status polling

During the optimization process, you will be able to follow the cost evolution and the state of the optimization by calling the getStatus method of the tsCloudApi object :

```
<script>
	tsCloudApi.getStatus(taskId,getStatusSuccessHandler,errorHandler);
</script>
```

You can use the same errorHandler function we used for the optimization request. Your getStatusSuccessHandler should mainly check the optimization state, but it can be used to track the costs evolution. It is quite easy to build graphs to show this evolution. Here is an example using canvasjs. First of all, you must include canvasjs :

```
<html>
	<head>
		<title>TsCloud Api Tester</title>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
		<script src="https://cdnjs.cloudflare.com/ajax/libs/canvasjs/1.7.0/canvasjs.min.js"></
script>
		<script src="https://geoservices.geoconcept.com/ToursolverCloud/combo?toursolver/
tsCloudApi.js"></script>
	</head>
	<body>
	...
	</body>
</html>
```

Then, here is an example of getStatus command called every second, with the success handler used to build graphs, and to trigger the result download (that we will detail in a further section) :

```
<script>
	var getStatusTimeout = null;
	var graphIndex;
	var graphCost = null;
	var graphDist = null;


	function getStatus() {
		if (getStatusTimeout != null) {
			clearTimeout(getStatusTimeout);
		}
		getStatusTimeout = setTimeout(function() {
			tsCloudApi.getStatus(taskId,getStatusSuccessHandler,errorHandler);
		},1000);
	}

	function appendGraphData(data) {
		if (graphCost == null) {
			graphIndex = 0;
			var options = {
				title: {
					text: "Cost evolution"
				},
				animationEnabled: true,
				data: [
					{
						type: "spline", //change it to line, area, column, pie, etc
						dataPoints: [
```

```
                                                { x: graphIndex, y: data.currentCost }
                                        ]
                                }
                        ]
                };

                graphCost = new CanvasJS.Chart("chartContainerCost",options);

                options = {
                        title: {
                                text: "Distance evolution"
                        },
                        animationEnabled: true,
                        data: [
                                {
                                        type: "spline", //change it to line, area, column, pie, etc
                                        dataPoints: [
                                                { x: graphIndex, y: data.currentDriveDistance /
 1000 }
                                        ]
                                }
                        ]
                };

                graphDist = new CanvasJS.Chart("chartContainerDist",options);

        } else {
                graphIndex++;
                graphCost.options.data[0].dataPoints.push({ x: graphIndex, y: data.currentCost });
                graphDist.options.data[0].dataPoints.push({ x: graphIndex, y:
 data.currentDriveDistance/1000 });
        }
        graphCost.render();
        graphDist.render();
    }

    function getStatusSuccessHandler(data,status,xhr) {
        if (data.optimizeStatus == "ERROR") {
                console.log("sorry, an error occured");
        } else if (data.optimizeStatus == "TERMINATED") {
                console.log("optimization done");
                setTimeout("getResult()",10);
        } else {
                if (data.status == "OK") {
                        console.log(data.optimizeStatus +  " | current cost : " + data.currentCost +
 " | current distance : " + data.currentDriveDistance);
                        if (data.optimizeStatus == "RUNNING") {
                                appendGraphData(data);
                        }
                } else {
                        console.log("sorry, an error occured : " + data.message);
                }
                getStatus();
        }
    }

</script>
```

Stopping an optimization

Even if you must specify a maximum optimization time in the options objects sent in the optimization command, you can stop an optimization at any time. You can do this easily using the stop method of the tsCloudApi object :

```
<script>
        tsCloudApi.stop(taskId,stopSuccessHandler,errorHandler);
</script>
```

Stopping an optimization may take a few seconds and when stopSuccessHandler is called, it only means that your stopping request as been aknowleged. Therefore, you still have to call the getStatus method until the status says that the optimization is terminated.

```
<script>
        function stopSuccessHandler(data,status,xhr) {
                if (data.optimizeStatus == "ERROR") {
                        console.log("sorry, an error occured");
                } else {
                        getStatus();
                }
        }
</script>
```

Here we use the getStatus function detailed in the above section.

Retrieving the optimization result

Once your optimization is terminated, you can retrieve the result using the getResult method of the tsCloudApi object :

```
<script>
        function getResult() {
                tsCloudApi.getResult(taskId,showResult,errorHandler);
        }
</script>
```

The result will contain the resource and time assignment for each planned order, the list of unplanned orders and a list of warnings.

```
<script>
        var lastResult;
        function showResult(data,status,xhr) {
                if (data.status == "OK") {
      lastResult = data; //let's store the result for later ...
                        displayResult(data);
                } else {
                        appendTrace("sorry, an error occured : " + data.message);
                }
        }

        function displayResult(data) {
                $('#resultStats').append("<p>nb of unplanned orders : " + data.unplannedOrders.length+"</
p>");
                if (data.warnings && data.warnings.length > 0) {
```

```
                        for (var i=0;i<data.warnings.length;i++) {
                                $('#resultStats').append("<p>warning on " + data.warnings[i].id + " : " +
 data.warnings[i].message + " (" + data.warnings[i].value + ")"+"</p>");

                        }
                }

                var order = null;
                for (var i=0;i<data.plannedOrders.length;i++) {
                        order = data.plannedOrders[i];
                        $('#resultPlanning').append("<p>order " +  order.stopId + " as been planned on
 resource " + order.resourceId + " on day " + order.dayId + " at " + order.stopStartTime +"</p>");
                }

        }
</script>
```

Exporting result to operational planning

You can aslo export your optimization result to operational planning so that mobile resources could browse it on the field through a mobile app. Note that you must have previously created your mobile identifiers through TsCloud app. If you are trying to export to a period that already contains data, you will get an error and have to force the export, which will erase previous data first.

```
<script>
        function doExportToOperational(taskId,exportStartDate,force) {
                var operationalExportParams = {
                        resourceMapping : [
                                {
                                        id:"Robert",
                                        operationalId:"robert@mycompany.com"
                                }
                        ],
                        startDate : (exportStartDate)?exportStartDate:new Date(),
                        force : force,
                        taskId : taskId
                };


 tsCloudApi.exportToOperationalPlanning(operationalExportParams,showExportResult,errorHandler);
        }
</script>
```

Integrating Toursolver result page in your app

Once your optimization is finished, you can use Toursolver to show the result. To to this, you will have to retrieve a temporary login token and then open TsCloud in an iframe with this token and the simulationId found in the optimization result.

You can choose to show the full Toursolver UI, or only the result page just passing standalone=true as a parameter of the url.

```
<iframe id="integrationFrame" style="width:100%;height:800px;border:none;"></iframe>
<script>
    var doStandaloneIntegration = false;
    var lastResult; //we assume here that you have stored the result obtained earlier in this var
```

```
    function startIntegration(standaloneIntegration) {
        doStandaloneIntegration = standaloneIntegration;

        //Here we retrieve a temporary token
        tsCloudApi.getGatewayToken(startIntegrationHandleFunc, errorHandler);
    }

    function startIntegrationHandleFunc(data,status,xhr) {
        //We have received our token, let's build the url to display the simulation result
        let url = 'https://app.geoconcept.com/ToursolverCloud/ts/login?';
        url += 'token='+data.token;
        url += '&simulationId='+ lastResult.simulationId;
        if (doStandaloneIntegration) {
            url += '&standalone=true';
        }

        //Here we will set the src of an existing iframe (having id='integrationFrame') to display
 Toursolver result page in it
        $('#integrationFrame').attr('src', url)
    }
</script>
```

If you can also specify the day and the resource to be shown :

https://app.geoconcept.com/ToursolverCloud/ts/login?
token=xxx&standalone=true&day=1&resource=Robert

By default, the standalone mode also hides the resources/days selector above the orders list. You can get it back by setting the showSelector parameter :

https://app.geoconcept.com/ToursolverCloud/ts/login?
token=xxx&standalone=true&day=1&resource=Robert&showSelector=true

With new result page only, you can open a result in read only mode, all actions are disabled in this mode :

https://app.geoconcept.com/ToursolverCloud/ts/login?
token=xxx&standalone=true&day=1&resource=Robert&readOnly=true

For compatibility, you can still use the old integration url (https://geoservices.geoconcept.com/ToursolverCloud), in this case, use tsCloudApi.getLoginToken instead of tsCloudApi.getGatewayToken, but you should migrate to the new url.

Api clients

In the following sections, you will find some client libraries for different languages and tips that will help you to use our API. If you do not find a client for your favorite language, you will probably be able to generate your own one from our wsdl [https://geoservices.geoconcept.com/ToursolverCloud/api/ts/toursolverService?wsdl] or wadl [https://geoservices.geoconcept.com/ToursolverCloud/api/ts?_wadl].

C# Client Library

We do not provide C# client library but you can easily build one from the wdsl using the ServiceModel Metadata Utility Tool (Svcutil.exe) [https://docs.microsoft.com/en-us/dotnet/framework/wcf/

servicemodel-metadata-utility-tool-svcutil-exe] or Visual Studio's Add Service Reference Dialog [https://msdn.microsoft.com/en-us/library/bb386382.aspx].

C# Resource Example :

```
//read a resource from a REST url
Uri uri = new Uri(...);

XmlSerializer s = new XmlSerializer(
  typeof( byte[] )
);

  //Create the request object
WebRequest req = WebRequest.Create(uri);
WebResponse resp = req.GetResponse();
Stream stream = resp.GetResponseStream();
TextReader r = new StreamReader( stream );

byte[] result = (byte[]) s.Deserialize( r );

//handle the result as needed...
```

Java XML Client Library

The Java client-side library is used to provide the set of Java objects that can be serialized to/from XML using JAXB. This is useful for accessing the resources that are published by this application.

Download toursolver-xml-client.jar

Resources Example (Raw JAXB) :

```
java.net.URL url = new java.net.URL(baseURL + "/toursolver/optimize");
JAXBContext context = JAXBContext.newInstance( byte[].class, byte[].class );
java.net.URLConnection connection = url.openConnection();
connection.setDoOutput(true);
connection.connect();

Unmarshaller unmarshaller = context.createUnmarshaller();
Marshaller marshaller = context.createMarshaller();
marshaller.marshal(optimizeRequest, connection.getOutputStream());
OptimizeResult result = (OptimizeResult) unmarshaller.unmarshal( connection.getInputStream() );
//handle the result as needed...
```

Resources Example (Jersey client) :

```
javax.ws.rs.client.Client client = javax.ws.rs.client.ClientBuilder.newClient();

OptimizeResult result = client.target(baseUrl + "/toursolver/optimize")
  .post(javax.ws.rs.client.Entity.entity(optimizeRequest, "application/xml"), OptimizeResult.class);

//handle the result as needed...
```

JavaScript Client Library

The JavaScript client-side library defines classes that can be (de)serialized to/from JSON. This is useful for accessing the resources that are published by this application, but only those that produce a JSON representation of their resources (content type "application/json").

he library uses ES6 class syntax which has limited support. See MDN [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes#Browser_compatibility] and the ES6 Compatibility Table [https://kangax.github.io/compat-table/es6/] for more details.

The library contains a UMD loader which supports AMD, CommonJS and browser globals. The browser global variable name for this library is "javascriptClient".

Download toursolver-js.zip

JavaScript Example :

```
/read the resource in JSON:
var json = JSON.parse(jsonString);

//create an object
var object = new Object(json);

//retreive the json again
var newJson = object.toJSON();

//serialize the json
var newJsonString = JSON.stringify(newJson);
```

PHP JSON Client Library

The PHP JSON client-side library defines the PHP classes that can be (de)serialized to/from JSON. This is useful for accessing the resources that are published by this application, but only those that produce a JSON representation of their resources (content type "application/json").

This library requires the json_encode [http://php.net/manual/en/function.json-encode.php] function which was included in PHP versions 5.2.0+.

Download toursolver-php.zip

PHP JSON Example :

```
//read the resource in JSON:
$json = ...;

//read the json as an array.
$parsed = json_decode($json, true);

//read the json array as the object
$result = new Object($parsed);

//open a writer for the json
$json = $result->toJson();
```

PHP XML Client Library

The PHP client-side library defines the PHP classes that can be (de)serialized to/from XML. This is useful for accessing the resources that are published by this application, but only those that produce a XML representation of their resources.

This library leverages the XMLReader [http://php.net/manual/en/book.xmlreader.php] and XMLWriter [http://php.net/manual/en/book.xmlwriter.php] tools that were included in PHP versions 5.1.0+.

Download toursolver-php.zip

PHP XML Example :

```
//read the resource in XML form:
$xml = ...;

$reader = new \XMLReader();

if (!$reader->open($xml)) {
  throw new \Exception('Unable to open ' . $xml);
}
$result = new Object($reader);

//open a writer for the xml
$out = ...;
$writer = new \XMLWriter();
$writer->openUri($out);
$writer->startDocument();
$writer->setIndent(4);
$result->toXml($writer);
$writer->flush();
```

Definitions

(JSON) AddVisitsRequest

Name	Description	Schema
id *optional*	Example : "null"	string
language *optional*	Example : "null"	string
orders *optional*		< json_TSOrder > array

(JSON) AddVisitsResult

Name	Description	Schema
message *optional*	Example : "null"	string
status *optional*	Example : "null"	string

(JSON) BalanceType

Type : enum (NONE, ORDERS, HOURS, QUANTITY)

(JSON) CostOperator

Type : enum (SUM, MAX, MIN, AVERAGE)

(JSON) DepotCostMode

Type : enum (NONE, ALL, FIRST, ALLBUTFIRST)

(JSON) DepotsResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
depots *optional*	List of depots	< json_TSDepot > array
message *optional*	error message Example : `"null"`	string
status *optional*	response status, OK or ERROR	json_Status

(JSON) DistanceType

Type : enum (KILOMETERS, MILES, METERS, FEET)

(JSON) EbookingWebhookFeature

Type : enum (ORDERSATISFACTION_CHANGED)

(JSON) EbookingWebhookRequest

Name	Description	Schema
feature *optional*		json_EbookingWebhookFeature
payload *optional*		object

(JSON) FulfillmentResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
lastKnownPosition *optional*	List of positions	< json_OperationalLastKnownPosition > array
message *optional*	error message Example : `"null"`	string
operationalOrderAchievements *optional*	List of orders	< json_OperationalOrderAchievement > array

Name	Description	Schema
status *optional*	response status, OK or ERROR	json_Status

(JSON) GPSStatus

<p>Classe Java pour GPSStatus.

<p>Le fragment de sch??ma suivant indique le contenu attendu figurant dans cette classe. <p> <pre> <simpleType name="GPSStatus"> <enumeration value="0"/> <enumeration value="1"/> <enumeration value="2"/> </restriction> </simpleType> </pre>

Type : enum (0, 1, 2)

(JSON) GeocodeInfos

Name	Description	Schema
address *optional*	Example : `"null"`	string
addressComplement *optional*	Example : `"null"`	string
city *optional*	Example : `"null"`	string
country *optional*	Example : `"null"`	string
geocodeAddressLine *optional*	Example : `"null"`	string
geocodeCity *optional*	Example : `"null"`	string
geocodePostalCode *optional*	Example : `"null"`	string
geocodeType *optional*		number
postcode *optional*	Example : `"null"`	string
region *optional*	Example : `"null"`	string
score *optional*		number

(JSON) LoginTokenResult

Result of the optimize service

Polymorphism : Composition

Name	Description	Schema
message	error message	string

Name	Description	Schema
optional	Example : `"null"`	
status *optional*	response status, OK or ERROR	json_Status
token *optional*	the token string Example : `"null"`	string
validUntil *optional*	The token validity end date	number

(JSON) OperationalExportRequest

Operational planning export parameters

Name	Description	Schema
dayNums *optional*		< number > array
force *optional*	if true, any existing operational planning will be replaced if false and if optimization period overlaps any existing operational planning, export will fail.	boolean
resourceMapping *optional*	List of MobileResourceMapping defining relation between resource identifier in optimize request and real mobile resource identifier	< json_OperationalResourceMapping > array
startDate *optional*	real date corresponding to day 1 of optimize request data	number
taskId *optional*	Task identifier. Must point to a completed optimization. Example : `"null"`	string

(JSON) OperationalLastKnownPosition

Polymorphism : Composition

Name	Description	Schema
accuracy *optional*	GPS positioning accuracy (radius in meters)	number
batteryLevel *optional*	Battery level of the device	number
date *optional*	last position recording date	number
gpsStatus *optional*	GPS status of the device	json_GPSStatus
id *optional*	Example : `"null"`	string
lat *optional*	Latitude	number
lon *optional*	Longitude	number
privateLife *optional*	Private life status in Mobile App. If true, it means that mobile App is currently in Private life mode, therefore this position is the last known position before the Private life switch.	boolean

(JSON) OperationalOrderAchievement

Polymorphism : Composition

Name	Description	Schema
achievementComment *optional*	Achievement comment Example : `"null"`	string
achievementEnd *optional*	Achievement end date and time	number
achievementEndLat *optional*	Achievement end latitude	number
achievementEndLon *optional*	Achievement end longitude	number
achievementStart *optional*	Achievement start date and time	number
achievementStartLat *optional*	Achievement start latitude	number
achievementStartLon *optional*	Achievement start longitude	number
apprauchSmsId *optional*	Example : `"null"`	string
apprauchSmsStatus *optional*	Example : `"null"`	string
data *optional*	fulfillment form data	< string, string > map
date *optional*	Planning day	number
end *optional*	Planned end date and time	number
feedbackSmsId *optional*	Example : `"null"`	string
feedbackSmsStatus *optional*	Example : `"null"`	string
geocode *optional*		json_GeocodeInfos
id *optional*	Example : `"null"`	string
lastSynchroStatusChange *optional*	Last change from mobile app	number
lat *optional*	Latitude	number
lon *optional*	Longitude	number
operationalResourceId *optional*	Mobile resource identifier (mobile app login) Example : `"null"`	string
order *optional*	Original order	json_TSOrder

Name	Description	Schema
pictures *optional*	List of picture relative urls. Url root for pictures is https://geoservices.geoconcept.com/TournsolverCloud/api/rest/otmobile/pictures/	< string > array
plannedOrder *optional*	Planned order	json_TSPlanned
signaturePicture *optional*	Signature, as a picture relative URL (a flavor of the reference *signatureSvg*) Is bound and synced from the *signaturesSvg* field. Is a relative URL, as also done for the *pictures* field (see its documentation for details). Example : `"null"`	string
signatureSvg *optional*	Signature svg Example : `"null"`	string
simulationDayId *optional*	day containing this order in the simulation used to fill the fulfillment planning Example : `"null"`	string
simulationId *optional*	identifier of the simulation used to fill the fulfillment planning Example : `"null"`	string
start *optional*	Planned start date and time	number
status *optional*	fulfillment status	json_OperationalOrderStatus
synchroStatus *optional*	Sync status	json_OperationalOrderSynchroStatus
timeWindowEnd *optional*		number
timeWindowSmsId *optional*	Example : `"null"`	string
timeWindowSmsStatus *optional*	Example : `"null"`	string
timeWindowStart *optional*		number
type *optional*	Event type	json_OperationalOrderType

(JSON) OperationalOrderStatus

Type : enum (CANDIDATE, FIXED, ACCEPTED, REFUSED, STARTED, FINISHED, CANCELLED, PAUSED, RESUMED, UNKNOWN)

(JSON) OperationalOrderSynchroStatus

Type : enum (PUBLISHED, SENT, UPDATED, UNKNOWN)

(JSON) OperationalOrderType

Type : enum (MISSION, RESTBREAK, LUNCHBREAK, WAITBREAK, RELOADBREAK, START, END, ENDBEFORENIGHT, STARTAFTERNIGHT, BRIEFING, DEBRIEFING, UNKNOWN)

(JSON) OperationalResourceMapping

The mobile resource mapping links the resource identifier used in optimized data and the mobile resource identifier used for operational planning export

Name	Description	Schema
id *optional*	resource identifier Example : `"null"`	string
operationalId *optional*	Mobile identifier Example : `"null"`	string

(JSON) OptimResultResult

Result of an optimization task.

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message Example : `"null"`	string
plannedOrders *optional*	the planned stops	< json_TSPlanned > array
simulationId *optional*	Id of the simulation associated to this task Example : `"null"`	string
status *optional*	response status, OK or ERROR	json_Status
taskId *optional*	the id of the optimization task (usefull if result is automatically sent through a webhook). Example : `"null"`	string
unplannedOrders *optional*	the orders which has not been planned because: it was sent by an other mail service it was not scheduled by any resource.	< json_TSUnplanned > array
warnings *optional*	the list of warning messages (and associated error codes) about possible Orders and Resources elements constraints misconfiguration.	< json_TSWarning > array

(JSON) OptimStatusResult

Result of status request

Polymorphism : Composition

Name	Description	Schema
currentCo2 *optional*	Remaining Kg CO2 of the current solution	number
currentCost *optional*	Cost of the current solution	number
currentCourierCost *optional*	Courier cost of the current solution	number
currentDeliveredQuantity *optional*	Total quantity delivered of the current solution	number
currentDeliveryCost	Delivery cost of the current solution	number

Name	Description	Schema
optional		
currentDriveCost *optional*	Drive cost of the current solution	number
currentDriveDistance *optional*	Drive distance of the current solution	number
currentDriveTime *optional*	Drive time in seconds of the current solution	number
currentFixedCost *optional*	Fixed cost of the current solution	number
currentLateTime *optional*	Late time in seconds of the current solution	number
currentNightsCost *optional*	Nights cost of the current solution	number
currentOpenTourNumber *optional*	initial number of open tours (tours with at least one visit)	number
currentOverWorkCost *optional*	Overwork cost of the current solution	number
currentOverWorkTime *optional*	Over work time in seconds of the current solution	number
currentPickUpQuantity *optional*	Total quantity picked-up of the current solution	number
currentPlannedVisits *optional*	Number of planned visits of the current solution (new engine only)	number
currentRestTime *optional*	Rest time in seconds of the current solution	number
currentUnplannedVisits *optional*	Number of visits unplanned or delivered by a courier of the current solution	number
currentVisitsNb *optional*		number
currentWaitTime *optional*	Wait time in seconds of the current solution	number
currentWorkCost *optional*	Work cost of the current solution	number
currentWorkTime *optional*	Work time in seconds of the current solution	number
initialCo2 *optional*	Remaining Kg CO2 of the initial solution	number
initialCost *optional*	Cost of the initial solution	number
initialCourierCost *optional*	Courier cost of the initial solution	number
initialDeliveredQuantity *optional*	Total quantity delivered of the initial solution	number
initialDeliveryCost *optional*	Delivery cost of the initial solution	number

Name	Description	Schema
initialDriveCost *optional*	Drive cost of the initial solution	number
initialDriveDistance *optional*	Drive distance of the initial solution	number
initialDriveTime *optional*	Drive time in seconds of the initial solution	number
initialFixedCost *optional*	Fixed cost of the initial solution	number
initialLateTime *optional*	Late time in seconds of the initial solution	number
initialNightsCost *optional*	Nights cost of the initial solution	number
initialOpenTourNumber *optional*	initial number of open tours (tours with at least one visit)	number
initialOverWorkCost *optional*	Overwork cost of the initial solution	number
initialOverWorkTime *optional*	Over work time in seconds of the initial solution	number
initialPickUpQuantity *optional*	Total quantity picked-up of the initial solution	number
initialPlannedVisits *optional*	Number of planned visits of the initial solution (new engine only)	number
initialRestTime *optional*	Rest time in seconds of the initial solution	number
initialUnplannedVisits *optional*	Number of visits unplanned or delivered by a courier of the initial solution	number
initialWaitTime *optional*	Wait time in seconds of the initial solution	number
initialWorkCost *optional*	Work cost of the initial solution	number
initialWorkTime *optional*	Work time in seconds of the current solution	number
message *optional*	error message Example : `"null"`	string
mileageChartRemainingTime *optional*	Mileage and travel time matrix computing remaining time. This information may not be available depending on the geographical area.	number
optimizeStatus *optional*	Current status of the optimization process	json_OptimizeStatus
simulationId *optional*	Id of the simulation associated to this task Example : `"null"`	string
startTime *optional*	Start time of the optimization	number
status *optional*	response status, OK or ERROR	json_Status
subOptimAbortedNb	Number of sub-optimizations in aborted state	number

Name	Description	Schema
optional		
subOptimErrorNb *optional*	Number of sub-optimizations in error state	number
subOptimFinishedNb *optional*	Number of sub-optimizations in finished state	number
subOptimNb *optional*	Number of sub-optimizations created (after pre-sectorization)	number
subOptimRunningNb *optional*	Number of sub-optimizations in running state	number
subOptimWaitingNb *optional*	Number of sub-optimizations in waiting state	number

(JSON) OptimStopResult

Result of status request

Polymorphism : Composition

Name	Description	Schema
firstStopAsked *optional*	First stop demand stamp	number
message *optional*	error message Example : `"null"`	string
status *optional*	response status, OK or ERROR	json_Status

(JSON) OptimToursResult

Result of an optimization task.

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message Example : `"null"`	string
status *optional*	response status, OK or ERROR	json_Status
taskId *optional*	the id of the optimization task (usefull if result is automatically sent through a webhook). Example : `"null"`	string
tours *optional*	List of tours (one tour per resource per day)	< json_TSTour > array
unplannedOrders *optional*	the orders which has not been planned because: it was sent by an other mail service it was not scheduled by any resource.	< json_TSUnplanned > array
warnings *optional*	the list of warning messages (and associated error codes) about possible Orders and Resources elements constraints misconfiguration.	< json_TSWarning > array

(JSON) OptimizeRequest

Optimization request

Name	Description	Schema
beginDate *optional*	This date will be used if you try to export the optimization result through TsCloud GUI. Format : YYYY-MM-DD Default : day+1.	number
countryCode *optional*	Main country code used to route the optimization to the good optimization server farm. Example : `"null"`	string
depots *optional*	list of depots (for reloading or starting tours)	< json_TSDepot > array
language *optional*	Language to use for message localization Example : `"null"`	string
options *optional*	the optimize task options	json_TSOptions
orders *optional*	list of orders (visits to do)	< json_TSOrder > array
organization *optional*	In multi-user content, you generally create various organizations that allows to define who sees what. If you set the organization here, only users that can see this organization will see the result of this optimization in the UI. Note that if you specified a teamId instead of sending the resources in the optimization data, the organization will be guessed from the team. If no organization is set, all users will see the result. Example : `"null"`	string
resources *optional*	collection of resources, the elements which will perform deliveries, pick-ups, commercial visit, etc.	< json_TSResource > array
simulationName *optional*	Simulation name Optional : generated automatically if not provided Example : `"null"`	string
userLogin *optional*	In multi-user content, you can specify a user login you. The simulation will be owned by this user. By default, the owner of the simulation is the default user of the account. Example : `"null"`	string

(JSON) OptimizeResult

Result of the optimize service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message Example : `"null"`	string
status	response status, OK or ERROR	json_Status

Name	Description	Schema
optional		
taskId *optional*	the id of the optimization task that was launched Example : `"null"`	string

(JSON) OptimizeStatus

Type : enum (undefined, waiting, geocoding, mileageChartBuilding, running, aborted, terminated, error, sectorizationWaiting, sectorizationRunning, sectorizationFinished, sectorizationAborted)

(JSON) PersistentObject

Name	Description	Schema
id *optional*	Example : `"null"`	string

(JSON) PhoneNumberType

Type : enum (MOBILE, OFFICE, HOME, OFFICE_FAX, HOME_FAX)

(JSON) ResourcesResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message Example : `"null"`	string
resources *optional*	List of resources	< json_TSResource > array
status *optional*	response status, OK or ERROR	json_Status

(JSON) RoutingMethod

Type : enum (TIME, DISTANCE)

(JSON) SectorizationMethod

Type : enum (TIME, DISTANCE)

(JSON) SimulationResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message Example : `"null"`	string
simulation		json_TSSimulation

Name	Description	Schema
optional		
status *optional*	response status, OK or ERROR	json_Status

(JSON) Status

Type : enum (OK, ERROR)

(JSON) TSAdditionalCost

Name	Description	Schema
type *optional*	TSOrder type (stored in a custom data) Example : `"null"`	string
value *optional*	Cost for this type	number

(JSON) TSDepot

The Depots object stores depots data. They are composed of X,Y coordinates, identity tags and a set of constraints.

If you want to specify depot locations or allow a resource to reload at several depots, use the Depots object to add and configure your depot data. The Depots object contains zero or more elements. If it contains no element, the base location of each resource acts as a depot location. Once a depot is associated with a resource its base location is no longer used as a depot location.

Polymorphism : Composition

Name	Description	Schema
allProductsRequired *optional*	Indicates whether a resource must provide all required products or one at least. Set it to True to indicate that a resource must provide all the depot required products. Set it to False to indicate that a resource must provide at least one of the depot required products. Type : boolean Default : True	boolean
availability *optional*	Indicates whether a depot is included in the planning process or not. This constraint enables you to easily modify your optimization problem configuration without having to add nor delete your original data. * Set it to True to include a depot element in the planning process. * Set it to False to ignore it: it will not be used in the planning.	boolean
deliveryQuantities *optional*	The available quantities of products available at the depot. You can specify up to 24 quantities Type : float array	< number > array
excludeResources *optional*	The list of resources excluded from the depot. Use this constraint to specify a list of resources that can not use the depot.	string

Name	Description	Schema
	Type : string array. Default: Empty array: no resource is excluded from the depot Example : `"null"`	
fixedLoadingDuration *optional*	The fixed duration for a resource to load at depot. Use this constraint to specify how long takes any resource reload at the depot. You can specify an additional duration according to the quantity to reload, using loadingDurationPerUnit. Type : "hh:mm:ss", DateTime. Default : "00:00:00". Example : tsDepot fixedVisitDuration = "00:30:00" indicates that 30 minutes are needed to load at the depot. tsDepot loadingDurationPerUnit = 120 indicates that 120 seconds are needed to load one unit. If the quantity to reload is 8, for instance, the variable part is 120*8. 16 minutes are required to load at the depot. Total load time = 30 minutes + 16 minutes = 46 minutes accounted for this reload break. Example : `"null"`	string
id *optional*	The unique identifier of the depot Example : `"null"`	string
loadingDurationPerUnit *optional*	The time needed to load a unit of product. This constraint is added to the fixed part of the loading duration: it depends on the total quantity to load. Type : "hh:mm:ss", DateTime, Integer (number of seconds). Default : 0. Example : `"null"`	string
openingDaysList *optional*	The depot days of opening, related to the nth depot time window. Use this constraint to specify the days of operation of a depot referring to a depot time window. It must be related to the tsResourceWorkDays constraint of the resources, as resources work days define the planning period. Depot may be opened on a 64-days long period max, from day 1 to day 64. Type : string value containing days separated with commas (like "1, 2, 5" or "4, 20/05/2010") or intervals (like "1-10", "2=>5" or "22/05/2010=>03/06/2010"). For day intervals, prefer the "=>" separator. 1 is the first day of the planning period, the corresponding date, if any, is defined by the oldest date, for all resources, in the tsResourceWorkDays constraint. No date in the depots days of opening description can precede the first day of the planning period defined by the resources (which is day 1). Default : 1=>64. Example : * Set it to "1" (or "14/05/2010) when a depot is opened on the first day of the planning period. Set it to "1, 2, 4, 5" (or "14/05/2010,15/05/2010,17/05/2010,18/05/2010") if depot may not be visited on the third day of a five-days planning period. * Set it to "1=>5" (or "14/05/2010=>18/05/2010") to define a 5-days long period during which the depot is opened.	< string > array
pickupQuantities *optional*	The available space for a product available at the depot. You can specify up to 24 values. Type : float array	< number > array
priority	Depot priority.	number

Name	Description	Schema
optional	Use this constraint to specify a priority on the depot. If two depots are nearby the system considers the one with the highest priority **Type : * integer	
requiredProducts *optional*	The list of the products a depots contains. Use this constraint when a depot must be affected to a specific kind of resource: these resources will have to provide the required required to be able to load at the depot. Type : string (as a list of products separated with commas). Default : none Example : `"null"`	string
resourceNames *optional*	Lists the resources that can use the depot. Type : string array. Default : Empty array: no resource can use the depot. Example : Specify ["Vehicle 1","Vehicle 2"] to allow only the resources with tags "Vehicle 1" and "Vehicle 2" to use the depot. Example : `"null"`	string
timeWindows *optional*	The depot time windows. Use this constraint to specify a depot time window during which the depot is opened. A depot time window is defined by its start time, its end time and days of opening. The start time of a depot time window is the first instant when a resource can enter the depot. Its end time is the last instant when a resource can enter the depot (but it may leave it afterward). You can specify up to 4 depot time windows for each depot. Type : "hh:mm:ss", DateTime. Default : "00:00:00".	< json_TSTimeWindow > array
travelTimeModifier *optional*	Indicates the value of the travel time modifier. When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location (tsDepotTravelTimeModifierValue), the portion of the travel time on which the modifier applies (tsDepotTravelTimeModifierLength) an offset to add to any travel duration leaving or reaching the location (tsDepotTravelTimeModifierOffSet). Example : * Set tsResource travelTimeModifierValue to 1.5, tsResource travelTimeModifierLength to 300 and tsResource travelTimeModifierOffSet to 60 for Resource 1 * Set tsDepot travelTimeModifierValue to 2, tsDepot travelTimeModifierLength to 420 and tsDepot travelTimeModifierOffSet to 0 for Depot 1 If the initial travel duration between Resource 1 and Depot 1 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660 Type : float, Default : 1	json_TSTravelTimeModifier
x	longitude WGS84 of depot	number

Name	Description	Schema
optional		
y *optional*	latitude WGS84 of depot	number

(JSON) TSEvaluationInfos

Name	Description	Schema
orderOriginalResourceID *optional*	The identity of the resource which visits the customer when evaluating an existing planning. * Use this constraint when you want to *evaluate* an existing planning. * Use the orderOriginalVisitDay constraint to specify the working day when the visit must be planned. * Use the orderPosition constraint to specify the position of a visit in the tour. Type : string (storing a unique resource tag). Default: : not used Example : `"null"`	string
orderOriginalVisitDay *optional*	Indicates the work day when the resource which visits the customer when evaluating an existing planning. * Use this constraint when you want to *evaluate* an existing planning. * Use the orderOriginalResourceID to specify the resource which visit the customer. * Use the orderPosition to specify the position of a visit in the tour. Type : integer between 1 and 64 or string representing a date. 1 is the first day of the planning period, the corresponding date, if any, is defined by the oldest date, for all resources, in the tsResourceWorkDays constraint. if the original visit day is in the date format, this date cannot precede the first day of the planning period defined by the resources (which is day 1). Default : not used if the tsOrderOriginalResourceIDis not used, 1 overwise Example : `"null"`	string
orderPosition *optional*	The position of a customer in a resource tour. Use this constraint when you want to evaluate a tour. Type : integer. Default : not used	number

(JSON) TSObject

Type : object

(JSON) TSOptions

Options for optimization request

Name	Description	Schema
allowBridge *optional*	Allow toll If false, all bridges will be avoided.	boolean

Name	Description	Schema
	Default : true.	
allowToll *optional*	Allow toll If false, all toll ways will be avoided. Default : true.	boolean
allowTunnel *optional*	Allow toll If false, all tunnels will be avoided. Default : true.	boolean
balanceType *optional*	Route plans balancing type * NONE : no balancing * ORDERS : balance the number of orders (to be used with balanceValue) * HOURS : balance the route plans duration * QUANTITY : balance the route plans delivered quantity Only available with OTSolver	json_BalanceType
balanceValue *optional*	Route plans balancing target (to be used with balanceType) Locked route plans are not considered. Only available with OTSolver	number
countDepotsInDeliveryCost *optional*	Depot cost configuration for tour delivery cost Specifies how depots stops are counted in tour delivery cost (using PenaltyPerVisit defined on resources). Possible values are : * NONE : depots stops are not counted in tour delivery cost * ALL : each depot stop counts as one visit in tour delivery cost * ALLBUTFIRST : first depot stop is ignored but each following depot stop will count as one visit in tour delivery cost * FIRST : only first depot stop is counted in tour delivery cost Default : NONE.	json_DepotCostMode
countVisitCostOnceIfSameLocation *optional*	If true, PenaltyPerVisit will counted only once in tour delivery cost if several visits at the same location are planned consecutively.	boolean
distanceType *optional*	Set the distance unit (meters, feet, km or miles), defaut is meters	json_DistanceType
evaluation *optional*	Enable this if you just want to evaluate (get cost, distances, etc) a set of tours. Therefore, you must fill the evaluationInfos objects in orders Default : false. (see TSEvaluationInfos)	boolean
excludeVisitCostIfMaxAdditionalCost *optional*	Exclude visit delivery cost for visits having the maximum additional cost	boolean
fuelCode *optional*	Vehicles fuel type. Possible values are : * diesel * undefined	string

Name	Description	Schema
	* unleaded Example : `"null"`	
maxOptimDuration *optional*	maximum optimization time. Default is one minute. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes) Example : `"null"`	string
noReload *optional*	Use it to forbid/allow reloads Default : false.	boolean
reloadDuration *optional*	Default reload duration Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes) Example : `"null"`	string
routingMethod *optional*	Routing method Specifies routing method used for travel time and distance matrix computation Possible values are : * TIME : routes minimizing travel time are used * DISTANCE : routes minimizing travel distance are used Default : TIME.	json_RoutingMethod
sendResultToWebhook *optional*	Enable result export to webhook If enabled, optimization result will be sent to the webook you defined in Toursolver Cloud configuration. Possible values are : * NONE : result will not be sent automatically when optimization finishes. You will have to call the result or toursResult service to retrieve it. * ORDERS : the "orders" result will be sent to the specified webhook. See OptimResultResult object for details. * TOURS : the "tours" result will be sent to the specified webhook. See OptimToursResult object for details. Default : NONE.	json_WebhookExportMode
speedPattern *optional*	Speed pattern name Specifies the speed pattern to be used for this optimization. "default_profile" is the internal name for the default speed pattern (its actual name in UI depends on your locale). You can use your own speed pattern (defined in Toursolver UI) but you will have to use its internal name : if you created a pattern called "Paris august speed pattern", its internal name will be "paris_august_speed_pattern". If set to null (or not defined), no speed pattern will be used at all (same speeds for whole day). Example : `"null"`	string
startFromEvaluationInfos *optional*	Enable this if you want the optimization to start from a specified initial configuration. Therefore, you must fill the evaluationInfos	boolean

Name	Description	Schema
	Default : false. objects in orders (see TSEvaluationInfos).	
teamId *optional*	If you did not specify any resource in your request, all resources defined in your tenant will be used for this optimization. If you specify a team identifier, only the resources of this team will be used. Example : `"null"`	string
useForbiddenTransitAreas *optional*	Whether to use forbidden transit areas Specifies true to use forbidden transit areas, and false to ignore forbidden transit areas If omitted, default to true, but has no effect if no forbidden transit areas are supplied	boolean
useOTSolver *optional*	Whether to use OTSolver instead of TSDK OTSolver is the new optimization engine behind Toursolver. It behaves better with big problems (more than 1000 visits). This is still a beta version and all the constraints supported by TSDK are not supported yet but they will be implemented as soon as possible. OTSolver must be enabled for your account before you can use it.	boolean
vehicleCode *optional*	Vehicles regulations. Possible values are : * bicycle * bus * car * deliveryIntermediateVehicle * deliveryLightCommercialVehicle * emergencyTruck * emergencyVehicle * intermediateVehicle * lightCommercialVehicle * pedestrian * taxi * truck If not specified, vehicle type defined in UI will be used Example : `"null"`	string

(JSON) TSOrder

The Order object stores order data.

Order is composed of X,Y coordinates, identity tags, a set of constraints and status reports.

Use the Orders object to specify your orders data. These elements can be considered as deliveries, pick-ups or commercial visits occuring at a specific locations.

The optimization process consists in assigning Orders elements to Resources elements in a way that respects the constraints of every element and minimizes the total cost of the planning.

Polymorphism : Composition

Name	Description	Schema
active *optional*	Indicates whether the order should scheduled or not Default : True.	boolean
allSkillsRequired *optional*	Indicates whether a resource must provide all required skills or one at least. * Set it to True to indicate that a resource must provide all the customer required skills. * Set it to False to indicate that a resource must provide at least one of the customer required skills. Default : True.	boolean
assignResources *optional*	The identities of the resources that can deliver a customer. Use this constraint when you want some customers to be delivered by specific resources. Type : string array containing resources tag ID Default : not used.	< string > array
courierPenalty *optional*	The cost of the order delivered by an independant transport device. Use this constraint when an order can be delivered independantly to specify the cost of the device. The solver engine will assign it to a courier device or integrate it in a tour according to the least cost. Type : float. Default : not used.	number
customDataMap *optional*	Private feature data. Use this feature when you need to keep accessible private data about customers. For instance if you need to export the address of an element in a planning report, store it there. These informations can also be exported to the mobile app. max 50 fields Default : empty.	< string, string > map
customerId *optional*	Example : `"null"`	string
delayPenaltyPerHour *optional*	The penalty per hour for being late at a customer. Use this constraint to allow a smarter control, with respect to time windows, than with the use of the tsOrderPunctuality constraint. Type : float. Default : not used.	number

Name	Description	Schema
email *optional*	Example : "null"	string
evaluationInfos *optional*	first order assignment (specifying resource, day and position in route). Use it to compare optimization result with a previous optimization Warning : used only if startFromEvaluationInfo is set to true in options.	json_TSEvaluationInfos
excludeResources *optional*	The identities of the resources which must not deliver a customer. Use this constraint when you want to prevent some customers from being delivered by specific resources.	< string > array
fixedVisitDuration *optional*	The fixed part of the total time spent visiting a customer. Use this constraint to specify the minimum time spent at a customer's. Use it to store the time needed by a vehicle to park, for instance, or the time planned for a commercial meeting. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used. Example : * tsOrderFixedVisitDuration = "00:30:00" indicates that the visit will be 30 minutes long. * tsOrderUnloadingDurationPerUnit = 120 indicates that 120 seconds are necessary to unload one unit. With a tsOrderQuantity equal to 8, for instance, the variable part is 120*8 (960 seconds or 16 minutes) to unload the customer ordered quantity. * Total delivery time = 30 minutes + 16 minutes = 46 minutes accounted for the customer delivery. Example : "null"	string
frequency *optional*	The visit occurrence within a defined period. Use this constraint to plan multiple visits at a customer during a period. Type: string value describing the number of visits over a period. Default: not used. Example : Frequency = "3/5" means a customer must be visited 3 times within a 5-days long period. The solver engine will create and optimize routes respecting an optimal duration between consecutive visits. Thus, back to the example, the computed period is 5/3 rounded = 2. The best solution will respect, as far as possible, a 2-days long period between each visit (ex: day1, day3, day5 is a good solution). Example : "null"	string
getNotifications *optional*		boolean
id *optional*	The unique identifier of the order Example : "null"	string
label *optional*	Example : "null"	string

Name	Description	Schema
minDuration *optional*	The min duration of the visit that enables partition. When the total duration of an order is very long, you may want to part it into multiple consecutive visits. First use this constraint to specify a duration threshold from which visit can be cut into multiple parts. Then use the tsOrderMinPartDuration to specify the min duration of one part. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used. Example : `"null"`	string
minPartDuration *optional*	The min duration of a part of a split visit. When the total duration of an order is very long, you may want to part it into multiple consecutive visits. First use the tsOrderMinDuration to specify a duration threshold from which the visit can be cut into multiple parts. Then use this constraint to specify the min duration of one part. The visit is split into parts of the set duration, and eventually a bigger one. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used. Example : `"null"`	string
phone *optional*	Example : `"null"`	string
possibleVisitDaysList *optional*	The possible visit days related to each visit time window. Use this constraint to specify the possible visit days of a customer referring to a visit time window. It must be related to the tsResourceWorkDays constraint of the resources, as resources work days define the planning period. You can plan visits on a 64-days-long period max. Type : array of string values containing visit days separated with commas (like "1, 2, 5" or "4, 20/05/2010") or intervals (like "1-10", "2=>5" or "22/05/2010=>03/06/2010"). For day intervals, prefer the "=>" separator. Be carrefull, if the separator of date is - then "1-5" corresponds to May the 1st of the current year. 1 is the first day of the planning period, the corresponding date, if any, is defined by the oldest date, for all resources, in the tsResourceWorkDays constraint. No date in the possible visit days description can precede the first day of the planning period defined by the resources (which is day 1). Default: 1=>64. You can set up to 4 sets of visit days in this array. If you define several time windows, each time window will apply to corresponding set of possible days : first time window applies to first possible days set, second time window applies to second possible days set, etc. Example : * Set it to "1" (or "14/05/2010) when a customer must be visited on the first day of the planning period. * Set it to "1, 2, 4, 5" (or "14/05/2010,15/05/2010,17/05/2010,18/05/2010") if the customer cannot be visited on the third day of a five days planning period. * Set it to "1=>5" (or "14/05/2010=>18/05/2010") to define a 5-days long period during which the customer can be visited.	< string > array

Name	Description	Schema
punctuality *optional*	The priority for a customer to be delivered on time. Use this constraint to introduce different priority levels for the respect of visit time windows (when defined). Specify a value from 1 to 5 to set the priority: customers with a punctuality set to 5 will have less risk of delayed deliveries than a customer with a punctuality set to 1. You can specify more precisely punctuality requirements to enforce delay control by using the tsOrderDelayPenaltyPerHour constraint instead of the tsOrderPunctuality constraint. Type : integer value from 1 to 5. Default : 1.	number
quantities *optional*	The requested product quantities for an order. Use this constraint to specify the product quantities ordered by a customer, for instance. Up to 24 quantities can be used to order different products. Useless when planning commercial tours. Type : float array (maximum of 3 significant digits after decimal separator). Default : 0 Max : 2,147,483. Example : * Use this constraint to store the number of ordered packages, when working with packages. * Use this constraint to store the number of ordered liters, when working with liquids. * Use multiple dimensions when working with multiple products: when delivering fuels for instance, you will use as many dimensions as types of fuels Warning : The quantity constraint of the *Orders* elements must be related to the corresponding capacity constraint of the *Resources* elements: thus, an order will not be planned whenever no resource with the related capacity has been defined.	< number > array
requiredSkills *optional*	The list of the skills a customer requires to be visited. Use this constraint when an order must be affected to a specific kind of resource: these resources will have to provide the required skills to be able to visit the customer. Type : string array. Default : none. Example : * Specify the word "Maintenance" as a required skill for a maintenance visit. Only the resources providing the "Maintenance" skill can perform the visit. * Specify "Small vehicle" as a required skill for a customer living down town.	< string > array
resourceCompatibility *optional*	Value for compatibility with resources. If the order has a compatibility constraint value, a resource is compatible with the order if its compatibility value is either non-existent either larger or equal to the order's one. See tsResourceOrderCompatibility.	number

Name	Description	Schema
	Example : * Set tsOrderResourceCompatibility to 254.18 and tsResourceOrderCompatibility to 260.12: order and resource are compatible * Set tsOrderResourceCompatibility to 254.18 and tsResourceOrderCompatibility to 200: the order can not be affected to the resource * Set tsOrderResourceCompatibility to 254.18 and tsResourceOrderCompatibility empty: order and resource are compatible * Set tsOrderResourceCompatibility empty and tsResourceOrderCompatibility to 260.12: order and resource are compatible Type : double Default : -1	
sequenceNumber *optional*		number
timeWindows *optional*	The visit time windows. Use this constraint to specify a visit time window during which the order can be delivered. A visit time window is defined by its start time, its end time and possible visit days (see tsOrderTimeWindow1_EndTime, tsOrderPossibleVisitDays1). You can specify up to 4 visit time windows for a customer. Type : array of time windows Default : "00:00:00". Example : A customer can only be delivered: * * on Monday from 11:00:00 to 12:00:00 AM and from 04:00:00 to 05:00:00 PM, * on Tuesday, Wednesday, Thursday from 08:00:00 to 09:00:00 AM and from 04:00:00 to 05:00:00 PM, * on Friday from 08:00:00 to 09:00:00 AM. Its visit time windows will be specified as follows: * first time window StartTime = "08:00:00", EndTime = "09:00:00", first set of possible days = "2=>5", * second time window StartTime = "11:00:00", EndTime = "12:00:00", second set of possible days = 1, * third time window StartTime = "16:00:00", EndTime = "17:00:00", third set of possible days = "1=>4".	< json_TSTimeWindow > array
travelTimeModifier *optional*	Indicates the value of the travel time modifier. When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location (tsOrderTravelTimeModifierValue), the portion of the travel time on which the modifier applies (tsOrderTravelTimeModifierLength)	json_TSTravelTimeModifier

Name	Description	Schema
	an offset to add to any travel duration leaving or reaching the location (tsOrderTravelTimeModifierOffSet). Example : * Set tsOrderTravelTimeModifierValue to 1.5, tsOrderTravelTimeModifierLength to 300 and tsOrderTravelTimeModifierOffSet to 60 for Order 1 * Set tsOrderTravelTimeModifierValue to 2, tsOrderTravelTimeModifierLength to 420 and tsOrderTravelTimeModifierOffSet to 0 for Order 2 If the initial travel duration between Order 1 and Order 2 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660 Type : float Default : 1	
tsOrderBefore *optional*	Example : "null"	string
tsOrderBeforeMaxTimeSpacing *optional*	Example : "null"	string
tsOrderBeforeMinTimeSpacing *optional*	Example : "null"	string
tsOrderFixed *optional*	If true force the use of the specified day, resource and start time	boolean
tsOrderLastVisit *optional*	The number of days before the beginning of the planning the last visit belonging to the order has been performed Use this constraint in case of a frequency order to specify from how many days the last visit was performed before starting the current planning. The default value indicates this constraint is to be unused in the planning. Type : negative integer Default : 0 Max : -2,147,483. Example : If *tsOrderLastVisit* = -2, the last visit of the same order has been performed two days ago. Warning : if the number of occurences of the frequency is more than one, the constraints tsOrderLastVisit and tsOrderMinimumSpacing are not considered.	number
tsOrderMaximumSpacing *optional*	The maximum number of days required between two visits of a same order. Use this constraint in case of frequency order to specify how many days at most should be left between two visits of the order. Type : integer Max : 2,147,483.	number
tsOrderMinimumSpacing *optional*	The minimum number of days required between two visits of a same order. Use this constraint in case of frequency order to specify how many days at least should be left between two visits of the order. Type : integer	number

Name	Description	Schema
	Default : 0 Max : 2,147,483. Warning : if the number of occurences of the frequency is more than one, the constraints tsOrderLastVisit and tsOrderMinimumSpacing are not considered.	
type *optional*	The type of visit at a customer. * 0 (Delivery): the resource will delivere related quantities. * 1 (PickUp): the resource will pick-up related quantities. * 2 (DeliveryFirst): the resource will not mix deliveries and pick-ups in a same rotation. It is possible to mix pickups and 0 type deliveries in the same rotation. However, the delivered products will come from the depot and the pickup products will be unloaded at the depot. No product delivered to a customer can come from a pickup at another customer's. Type : Integer. Default : 0 (Delivery).	number
unloadingDurationPerUnit *optional*	The time needed to deliver one unit of product. Use this constraint when the time spent at a customer depends on the quantity to deliver. It is the time needed to unload one unit of the quantity stored in the customer tsOrderQuantity constraint. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used. Example : `"null"`	string
wholeVisitInTimeWindow *optional*	Whole visit ends before time window end. Use this constraint if you want that a visit ends before the end of the time window. Default : false.	boolean
x *optional*	longitude of order location	number
y *optional*	latitude of order location	number

(JSON) TSPause

Name	Description	Schema
duration *optional*	Duration Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Example : `"60"`	string
end *optional*	maximum end time Type : Time ("hh:mm" or "hh:mm:ss") Example : `"14:00"`	string
start *optional*	minimum start time	string

Name	Description	Schema
	Type : Time ("hh:mm" or "hh:mm:ss") Example : "12:30"	

(JSON) TSPlanned

Details of a stop within a resource tour.

Name	Description	Schema
dayId *optional*	day of the stop Example : "null"	string
resourceId *optional*	Assigned resource identifier Example : "null"	string
stopDriveDistance *optional*	The drive distance from the previous to the current stop.	number
stopDriveTime *optional*	The drive time from the previous to the current stop. Type : Time ("hh:mm" or "hh:mm:ss") Example : "null"	string
stopDuration *optional*	The duration of the stop. Type : Time ("hh:mm" or "hh:mm:ss") Example : "null"	string
stopElapsedDistance *optional*	The tour cumulated driven distance at the stop.	number
stopId *optional*	the ID of the stop or order Example : "null"	string
stopPosition *optional*	The position of the stop in the resource tour.	number
stopStartTime *optional*	The time the stop starts at. Type : Time ("hh:mm" or "hh:mm:ss") Example : "null"	string
stopStatus *optional*	The status of the stop. An order can be on time, late, waited, impossible (non compliant). It can also be a courier service delivery and not planned. See StopStatusConstants for more information about stop status. Type : integer. * 0 = OnTime: The stop begins into the specified visit time window. * 1 = Late: The stop begins after the specified visit time window. * 2 = Waited: The resource waits at the stop for the specified stop start time. * 3 = Impossible: The stop cannot be planned. * 4 = BadDay: The stop is planned without respect of frequency. * NotPlanned: The courier service is in charge of a stop or the stop is not included in the planning process.	number

Name	Description	Schema
	* BadPrevious : When the visits are split, the parts should follow one another. This status is that of a part following another visit instead of the previous part of the split visit.	
stopType *optional*	The stop type A tour is composed of stops, which can be either customers visits or breaks, like reload breaks, lunch breaks, or even night breaks. StopTypeConstants are * 0 = Visit: The stop is an element of the Orders object. * 1 = Start: The stop is the initial stage (departure) of the tour, occurring at the resource's start location. * 2 = End: The stop is the the last stage (arrival) of the tour, occurring at the resource's stop location. * 3 = EndBeforeOvernight: The stop is the the last stage (arrival) of the tour, occurring where the resource stopped for the night. * 4 = ReloadBreak: The stop is a reload break within the tour, occurring at a depot location. * 5 = LunchBreak: The stop is the lunch break within the tour, occurring anywhere between two other located stops. * 6 = RestBreak: The stop is a rest break within the tour, occurring anywhere between two located stops. * 7 = WaitBreak: The stop is a wait break within the tour, occurring before a visit. * 8 = StartAfterOvernight: The stop is the last stage (arrival) of the tour, it occurs where the resource stopped for the night. tsStopDriveDistance * 9 = Briefing: The stop is a briefing break within the tour, occurring right after the departure of the tour. * 10 = DeBriefing: The stop is a wait break within the tour, occurring right before the arrival of the tour.	number
stopX *optional*	The X coordinate of the stop.	number
stopY *optional*	The Y coordinate of the stop.	number

(JSON) TSResource

The Resources object stores the collection of the elements which will perform deliveries, pick-ups, commercial visit, etc.

Each element can be seen as a driver/vehicle pair and their related driving and working constraints.

It is composed of an identity tag, and location coordinates (base or depot, departure, arrival location). Eventually, each element can be configured through a set of constraints which stores information like capacity, driver work time, penalties and so on.

Polymorphism : Composition

Name	Description	Schema
additionalCostOperator *optional*	Operator used to compute additional cost (see toursResult object) Possible values are : * SUM : tour additional cost will be the sum of additional costs of planned orders * MAX : tour additional cost will be the maximum value of additional costs of planned orders * MIN : tour additional cost will be the minimum value of additional costs of planned orders * AVERAGE : tour additional cost will be the average of additional costs of planned orders	json_CostOperator
additionalCostOrderCustomDataName *optional*	Name of TSOrder customData that will contain the TSOrder type used for additional cost computation (see toursResult object) Example : `"null"`	string
additionalCosts *optional*	List of additional cost (cost per order type)	< json_TSAdditionalCost > array
available *optional*	Indicates whether a resource is included in the planning process or not. This constraint enables you to easily modify your optimization problem configuration without having to add nor delete original data. * Set it to True to include a resource element in the planning process. * Set it to False to ignore it: it will not be used in the planning. Type : Boolean. Default : not used. Example : `true`	boolean
avgConsumption *optional*	Average consumption of the resource. Use this constraint to specify the average consumption of the resource. This consumption must be defined in liters per 100 distance unit (km or miles) Default : The consumption indicated in the VehicleProfile.Consumption property. Example : `8.0`	number
briefingDuration *optional*	An additional duration applied before the tour starts. Use this constraint to specify an additional duration before the resource begins the tour. For instance, to brief drivers before they leave or to simulate an initial vehicle loading time (no depot). Type : "hh:mm:ss", DateTime. Default : "00:00:00". Example : `"00:20:00"`	string
capacities *optional*	The total product capacity a resource can carry. Use this constraint to specify the amount of product a resource can deliver or pick-up. It must be linked with the customers related quantity constraint. 23 more dimensions can be used in order to specify various products. They must be linked with the customers related dimensions. Type : float (maximum of 3 significant digits after decimal separator). Default: None. Max : 2,147,483.	< number > array

Name	Description	Schema
	Example : * Use it to store the total number of packages a resource can deliver before reloading, when working with packages. * Use it to store the total number of litres a resource can deliver before reloading, when working with liquids. * Use multiple capacity dimensions when working with multiple products: when delivering fuels, you will use as many capacities as types of fuels.	
customDataMap *optional*	Private custom data. Use this feature when you need to keep attributes values of your own into the optimization context. For instance if you need the address of an element, store it there.	< string, string > map
dailyWorkTime *optional*	The maximum resource work duration over a day. Use this constraint to specify the maximum daily work duration (unless overtime periods have been defined). If the dailyWorkTime constraint is not set to a value, default daily work time is computed from resource time window and lunch duration (workStartTime, workEndTime and lunchDuration). Type : "hh:mm:ss", DateTime. Default : not used. Example : `"null"`	string
debriefingDuration *optional*	An additional duration applied at the end of the tour. Use this constraint to specify an additional duration after the resources stops at the end location. For instance, to debrief drivers after their tour or to account an ultimate vehicle unloading time, or a parking time. Type : "hh:mm:ss", DateTime. Default : "00:00:00". Example : `"null"`	string
driveRestAtCustomer *optional*	Indicates whether or not time spent at customer during deliveries can be considered as rest break. * Set it to True if a visit can be considered as a rest break, as soon as its duration is equal or higher than the legalMinRestDuration constraint. * Set it to False if not. Type : Boolean. Default : False.	boolean
driveRestAtDepot *optional*	Indicates whether time spent at depot during a reload break can be considered as drive break or not. Use this constraint to take into account drive time legislation in your planning results. * Set it to True if load breaks can be considered as drive breaks, as soon as their duration is equal or higher than the legalMinRestDuration constraint. * Set it to False if not. Type : Boolean. Default : False.	boolean
endX *optional*	longitude of resource arrival Specify it only if arrival and start are not the same.	number
endY *optional*	latitude of resource arrival	number

Name	Description	Schema
	Specify it only if arrival and start are not the same.	
extraTravelPenalties *optional*	The costs for a resource of driving for one distance unit.	< json_TSTravelPenalty > array
fixedLoadingDuration *optional*	The fixed duration of the resource load break. Use this constraint to specify how long takes a resource reload at depot. You can specify an additional duration according to the quantity to reload, using loadingDurationPerUnit. Type : "hh:mm:ss", DateTime. Default : "00:00:00". Example : * fixedVisitDuration = "00:30:00" indicates that 30 minutes are needed to load the resource. * loadingDurationPerUnit = 120 indicates that 120 sec are needed to load one unit. If the quantity to reload is 8, for instance, the variable part is 120*8. 16 minutes are required to load the resource. * Total load time = 30 minutes + 16 minutes = 46 minutes accounted for this reload break. Example : `"null"`	string
fixedUnloadingDuration *optional*	The fixed time needed for the resource to deliver/pick-up at the customer's place. This fixed resource's unloading duration is added to the resource's unloading duration depending on the quantity to deliver/pick-up and to the order's unloading duration depending on the quantity to deliver/pick-up. See also tsResource unloadingDurationPerUnit and tsOrder unloadingDurationPerUnit. Example : * Set tsOrderQuantity to 3. * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 6 mn * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration to 10 mn and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 10 mn * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration empty and tsResource unloadingPerUnit to 1 mn: the variable part of the order is 3 mn Type : "hh:mm:ss", DateTime. Default : "" Warning : the fixed/per unit unloading duration for resource is not compatible with the orders constraints: partition and whole visit in time window. Hence if one of the constraint at least tsResource fixedUnloadingDuration and tsResource unloadingDurationPerUnit is non null, partition and whole visit in time window are disabled. Example : `"null"`	string
fuelCode *optional*	Vehicles fuel type. Possible values are : * diesel * undefined	string

Name	Description	Schema
	* unleaded Example : "null"	
fuelType *optional*	The fuel type of the resource. Use this constraint to specify the fuel type of the resource. Available fuel types are : * 0: Unknown fuel type * 1: No fuel type (pedestrian) * 2: Diesel * 3: Unleaded fuel * 4: LPG Type : Integer. Default : The fuel type indicated in the VehicleProfile.FuelType property.	number
globalCapacity *optional*	Global capacity for a resource. This constraint unables to set a maximum capacity on a resource, whatever product is carried. If both global capacity and capacity per product are to be used, set True for the value of *useAllCapacities*. Type : Float Default : None Max : 2,147,483. Example : Set globalCapacity = 15 and useAllCapacities = False. The global capacity only is considered: the vehicle can carry 15 quantities of products, whatever product is to be carried.	number
id *optional*	The unique identifier of the resource. This id can not contain special characters like = or *:* Example : "null"	string
legalDailyDriveDuration *optional*	The legal daily duration a resource can drive before having a rest break. Use this constraint to take into account drive time legislation in your planning results. Specify the rest duration in the legalDailyRestDuration: rest breaks occur as soon as the resource has driven the legal daily driving duration or has completed its daily work time. Type : "hh:mm:ss", DateTime. Default : not used. Example : legalDailyDriveDuration = "09:00:00", legalDailyRestDuration = "11:00:00". The resource can cumulate a 9-hours daily drive time before having to rest during 11 hours. Example : "null"	string
legalDailyRestDuration *optional*	The legal rest duration a resource must have after the daily max drive duration. Use this constraint to take into account drive time legislation in your planning results. Rest breaks occur as soon as the resource has driven the legal max daily duration or has complete its daily work time. The use of both legalDailyDriveDuration and legalDailyRestDuration implies that the start and end time of tours are no longer defined by the workStartTime and workEndTime constraints, but may vary as long as the legalDailyRestDuration constraint is respected. Type : "hh:mm:ss", DateTime. Default : not used. Example : "null"	string

Name	Description	Schema
legalDriveRestDuration *optional*	The resource break duration after max drive duration. Use this constraint to take into account drive time legislation in your planning results. Type : "hh:mm:ss", DateTime. Default : not used. Example : `"null"`	string
legalMaxDriveDuration *optional*	The legal max duration a resource can drive without a break. Use this constraint to take into account drive time legislation in your planning results. Type : "hh:mm:ss", DateTime. Default: : not used. Example : * legalMaxDriveDuration = "04:30:00", legalDriveRestDuration = "00:45:00". The resource can drive for 4 hours and 30 minutes before having a 45 minutes break. Then it can start driving again for 4 hours and 30 minutes???etc. * If added driveRestAtCustomer = True and legalMinRestDuration = "00:15:00", rest time is cumulated as soon as customer visit duration is equal or higher than the specified min rest duration. If the 45 minutes rest break has not occured when the resource has driven for 4 hours and 30 minutes, it must stop to complete the break. Otherwise, it can drive again for 4 hours and 30 minutes. Example : `"00:30:00"`	string
legalMinRestDuration *optional*	The minimum duration a resource breaks for it to be considered as a drive break. Use this constraint to take into account drive time legislation in your planning results. When breaks occur at customer or at depot, they are considered as drive breaks if their duration is equal or higher than the specified min rest duration. Type : "hh:mm:ss", DateTime. Default: : not used. Example : `"null"`	string
loadBeforeDeparture *optional*	Indicates whether a resource must load before starting a tour or not. * Set this constraint to True to insert a load break at depot before the tour starts. * Set it to False to consider the vehicle loaded. Type : Boolean. Default: : False.	boolean
loadOnReturn *optional*	Indicates whether a resource must reload after a tour or not. * Set this constraint to True to insert a load break at depot after the tour ends. * Set it to False to not reload the vehicle. Type : Boolean. Default: : False.	boolean
loadingDurationPerUnit *optional*	The time needed to load a unit of product. This constraint is added to the fixed part of the loading duration: it depends on the total quantity to load. Type : "hh:mm:ss", DateTime, Integer (number of seconds). Default: : 0. Example : `"null"`	string
lunch *optional*	The lunch break Default: : no lunch break	json_TSPause
maxNightsOutPerJourney *optional*	The number of nights a resource can spend out to perform deliveries before coming back to base.	number

Name	Description	Schema
	Use this constraint to enable night breaks trip: it will store how many days a resource can drive away from depot. Unless the vehicle needs to reload, it will not come back before. Type : integer. Default: : not used. Example: maxNightsOutPerJourney = 4, workDays = 1=>5. The resource can perform deliveries during 5 days (and 4 nights) without coming back to depot. Starting on day 1, it must come back to depot on day 5.	
maximumDistance *optional*	The maximum distance the resource should travel per day. Use this constraint to specify the distance the resource should not exceed in one day Type : integer Default : -1 (no limit)	number
maximumReloads *optional*	Value for maximum number of reloads per day.	number
maximumReloadsPenalty *optional*	Value of the penalty if the resource exceeds the maximum number of reloads per day. If the resource exceeds the maximum number of reloads per day, it will be penalized by the maximum reloads penalty multiplied by the number of reloads' overtaking. The default value is 1024 if the constraint maximumReloads is set. See also maximumReloads.	number
maximumVisits *optional*	The maximum number of visits the resource can perform in one day. Type : integer Default : -1 (no limit) Max : 2,147,483.	number
minDriveDuration *optional*	The minimum laps of time between 2 consecutive visits. Use this constraint to specify a minimum drive time accounted when consecutive visits are very close, in order to compute a realistic timing. Type : "hh:mm:ss", DateTime. Default: : not used. Example : `"null"`	string
minimumQuantity *optional*	Value for minimum quantity to deliver. If the resource has a non null minimum quantity to deliver constraint value, the resource can visit on order only if the quantity to be picked-up or delivered at this order is higher than the constraint value. *Example : * * set order *quantity* to 3 and resource *minimumQuantity* to 2 : resource might visit the order * set order *quantity* to 3 and resource *minimumQuantity* to 5 : resource can not visit the order Type : float	number
mobileLogin *optional*	Mobile login to be specified only if you need to be able to export the optimization result to operation planning from TsCloud GUI. If you trigger operational planning export from the API, you will still need to specify the mapping between resource name and login. Example : `"null"`	string
nightPenalty	The night cost of a resource when planning night breaks.	number

Name	Description	Schema
optional	Use this constrainte to specify how much a night out costs (lunch, bed, breakfeast???) for a resource when night breaks are allowed. Type : float. Default: : 0	
noReload *optional*	Constraint providing resource to reload at depot during a tour. If none of the constraints loadBeforeDeparture and loadOnReturn is activated, the resource is never loading at depot. If one of these constraint is activated the resource is loading once per tour, either at the beginning of the tour or at the end of it. If both constraints are activated then the resource is loading twice a tour, at the beginning and at the end. With standard solver only : solver always works with solution cost comparison, so it may allow reloads even if you set noReload to true if the solution cost is really lower. The only way to totally prevent reloads is to set fixedLoadingDuration to a very long duration (20:00:00 for instance). The solver will warn you about this long duration but will not add reloads to the tours. Caution : Don't do this if you are using loadOnReturn or loadBeforeDeparture Default : False	boolean
nonUsePenalty *optional*	A cost paid for NOT using a resource in the computed planning. Use this constraint to penalize the fact that an available resource is not used in the planning. It proves usefull when you must use all the resources in the Resources collection: you will prevent the solver from optimizing the number of resources, as using them would be too expensive a solution. Type : float. Default: : not used. Example : Use it to specify parking taxes when a vehicle performs no tour.	number
openDistanceStart *optional*		boolean
openDistanceStop *optional*		boolean
openStart *optional*	Indicates whether or not a tour evaluation starts from the resource start location or from the first customer's place. * Set it to True to begin the tour at the first customer (no cost will be computed between the resource start location and the first customer). * Set it to False to begin the tour at the resource start location. Type : Boolean. Default: : False.	boolean
openStop *optional*	Indicates whether or not a tour evaluation ends at the last customer's place or at the resource end location. * Set it to True to finish the tour at the last customer location (no cost will be computed between the last customer and the resource end location). * Set it to False to finish the tour at the resource end location. Type : Boolean. Default: : False.	boolean
openTimeStart *optional*		boolean

Name	Description	Schema
openTimeStop *optional*		boolean
optimumStartTime *optional*	Adapts the resource start time to the best solution. * Set this constraint to False to start the tour at the time indicated by the workStartTime constraint. * Set this constraint to True to optimize the resource start time and enable the use of the dailyWorkTime constraint. Notice that, in both cases, the tour will not start before the time stored in the workStartTime constraint and that the use of the weeklyWorkTime constraint is enabled. Type : Boolean. Default: : False.	boolean
otherWorkEndTimes *optional*	The end time of the resource work time window for the "other working days" (see otherWorkDaysList). If you defined other working days, use this constraint to specify the time at which the tour must end. These times can be adjusted using the optimumStartTime parameter. You can define up to 3 different end times here, corresponding to the different "other working days" slots. Type : List of "hh:mm:ss", DateTime. Example : ["18:00:00","19:00:00] Example : `"null"`	string
otherWorkStartTimes *optional*	The start time of the resource work time window for the "other working days" (see otherWorkDaysList). If you defined other working days, use this constraint to specify the time at which the tour must start. These times can be adjusted using the optimumStartTime parameter. You can define up to 3 different start times here, corresponding to the different "other working days" slots. Type : List of "hh:mm:ss", DateTime. Example : ["08:00:00","09:00:00] Example : `"null"`	string
otherWorkingDays *optional*	Other working days (see workingDays) You can define up to 3 different working days slots. If you do so, you can then specify distinct work start and end times with otherWorkStartTimes and otherWorkEndTimes. Type : List of string (see workingDays format) Example : ["2","3=>5"]	< string > array
overnightMinDriving *optional*	The max duration a resource can drive to go back to base at the end of a tour when working with night breaks. Use this constraint when a resource can end a tour at base instead of having a night break on the road. Type : "hh:mm:ss". Default: : not used. Example : `"null"`	string
overtimeDurations *optional*	The duration of the resource overwork periods. (max 2 periods)	< object > array

Name	Description	Schema
	Use this constraint when you need to define 2 different overwork periods with regard to daily work time and their related surcharge. No overwork period can be defined with regard to weekly work time. Type : "hh:mm:ss", DateTime. Default: : not used (no second overtime period). Example : workPenalty = 10, first overtime duration = "02:00:00", first overtime penalty = 5,second overtime duration = "01:00:00", second overtime penalty = 10. The cost of the resource during the daily working time is 10 euros per hour. The resource can work during 3 more hours and each overworked hour will cost 10 + 5 euros for 2 hours and 10 + 10 euros for the last hour.	
overtimePenalties *optional*	A surcharge for a resource's overwork period (max 2 periods). Use this constraint to specify additional cost to workPenalty as overwork time. You can define up to 2 overwork periods with regard to daily work time and their related surcharge. No overwork period can be defined with regard to weekly work time. Type : float. Default: : not used. Example : workPenalty = 10, first overtime duration = "02:00:00", first overtime penalty = 5. The cost of the resource during the daily working time is 10 euros per hour. The resource can work during 2 hours after the daily working time and each overworked hour will cost 15 (10 + 5) euros.	< number > array
payWholeDay *optional*	Indicates whether or not a whole work day cost is computed as soon as a resource works even a few hours. * Set this constraint to True to account a whole day cost. * Set it to False to account only effective worked hours. Type : Boolean. Default: : False.	boolean
penaltyPerVisit *optional*	An additional fixed cost applied each time the resource performs a visit. Use this constraint when the cost of a resource may vary according to the number of visits it performs. Type : float. Default: : 0.	number
providedProducts *optional*	The list of products provided by a resource. This constraint is linked with tsDepot requiredProducts constraint: a depot with a required product can only be visited by a resource providing it. Type : string as a list of products separated with commas. Default : not used. Example : Specify "Oxygen" in the resource can provide oxygen. Example : `"null"`	string
providedSkills *optional*	The list of characteristics provided by a resource.	string

Name	Description	Schema
	This constraint is linked with the same order constraint: a customer with a required skill can only be delivered by a resource providing it. Type : string as a list of characteristics separated with commas. Default: : not used. Example : * Specify "Maintenance" in the provided skills of a resource designed to perform maintenance visits type. * Specify "Small vehicle" to the provided skills of a resource able to perform downtown visits. Example : `"null"`	
speedAdjustment *optional*	A factor to increase or decrease the vehicle speed. Whenever users observe a significant gap between estimated speeds and the real ones, they can adjust them by using this factor. It is expressed as a percentage of the used speed. Type : float. Default: : not used. Example : * speedAdjustment = 110: resource speeds will be 10% higher than the used speeds. * speedAdjustment = 80: resource speeds will be 20% lower than the used speeds.	number
startTravelTimeModifier *optional*	travel time modifier associated with the start location of the resource	json_TSTravelTimeModifier
startX *optional*	longitude of resource start (and arrival if no endX provided)	number
startY *optional*	latitude of resource start (and arrival if no endY provided)	number
stopTravelTimeModifier *optional*	travel time modifier associated with the stop location of the resource	json_TSTravelTimeModifier
tomTomWebFleetEnabled *optional*		boolean
tomTomWebFleetIdentifier *optional*	Example : `"null"`	string
travelPenalty *optional*	The cost for a resource of driving for one distance unit. Use this constraint to specify the average resource taxes (gazoline, wear,???) when driving one distance unit. Type : float Default : 1.5 Example : if travelPenalty = 0.5 (euro per distance unit) and the driven distance is about 100 unit (km or miles), the total distance cost is 0,5 * 100 = 50 euros.	number
travelTimeModifier *optional*	travel time modifiers When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel	json_TSTravelTimeModifier

Name	Description	Schema
	time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location travelTimeModifierValue), the portion of the travel time on which the modifier applies (travelTimeModifierLength) an offset to add to any travel duration leaving or reaching the location (travelTimeModifierOffSet). Example : * Set travelTimeModifierValue to 1.5, travelTimeModifierLength to 300 and travelTimeModifierOffSet to 60 for Resource 1 * Set travelTimeModifierValue to 2, travelTimeModifierLength to 420 and travelTimeModifierOffSet to 0 for Order 1 If the initial travel duration between Resource 1 and Order 1 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660	
unloadingDurationPerUnit *optional*	the time needed to the resource to deliver/pick-up one unit of product at the customer's place. This resource's duration is added to the fixed resource's unloading duration and to the order's unloading duration depending on quantity to deliver/pick-up. See also tsResource fixedUnloadingDuration and tsOrder unloadingDurationPerUnit. Example : * Set tsOrderQuantity to 3. * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 6 mn * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration to 10 mn and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 10 mn * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration empty and tsResource unloadingPerUnit to 1 mn: the variable part of the order is 3 mn Type : "hh:mm:ss", DateTime. Example : `"null"`	string
useAllCapacities *optional*	Determines if both global capacity and capacities per product should be considered in the optimization or only global capacity. Type : Boolean Default : False Example : Set globalCapacity = 15 and useAllCapacities = False. The global capacity only is considered: the vehicle can carry 15 quantities of products, whatever product is to be carried.	boolean
useInPlanningPenalty *optional*	Penalty value if the resource is used in the planning. Use this constraint to specify the penalty if the resource is used at least one day in the planning Type : float Default : 0	number
usePenalty *optional*	A cost paid for using the resource in the computed planning. Use this constraint to ponderate the cost of using a resource element of the Resources	number

Name	Description	Schema
	collection, when working with rented vehicle for instance. You can use it as well to reduce the number of resources used to perform the deliveries. Type : float. Default: : not used. Example : Add the cost of use to the distance and hourly costs when working with a service provider. The solver, aiming at cost reduction, will try to eliminate this resource first, as it is the most expensive.	
vehicleCode *optional*	Vehicles regulations. Possible values are : * bicycle * bus * car * deliveryIntermediateVehicle * deliveryLightCommercialVehicle * emergencyTruck * emergencyVehicle * intermediateVehicle * lightCommercialVehicle * pedestrian * taxi * truck If not specified, vehicle type defined in UI will be used Only new solver can handle optimization with several types of vehicle. If you use standard solver, you should specify the vehicle code only in the options object. Example : `"null"`	string
weeklyWorkTime *optional*	The maximum resource work duration over a week. Use this constraint to specify the maximum weekly work duration. Weeks are defined as follows: * year’s week, from monday to sunday, if a date has been specified in the workDays constraint * days 1 to 7 are the days of the first week and the second week start on day 8 if no date has been specified in the workDays constraint. No over work with regard to the weekly work duration is allowed. Type : "hh:mm:ss", DateTime. Default: : "168:00:00". Example : `"null"`	string
workEndTime *optional*	The end time of a resource work time window.	string

Name	Description	Schema
	Use this constraint to specify the time at which the tour must be over or the daily work duration. * If the optimumStartTime parameter is set to True and the dailyWorkTime or weeklyWorkTime constraint value is set by user, the tour cannot finish later than the value of the workEndTime parameter, whatever overtime may be allowed by the overtime1_Duration and overtime2_Duration constraints. * If the optimumStartTime parameter is set to True and the dailyWorkTime constraint value is not set, the end time of the tour can be adjusted to match the begining of the work time window. In this case, this parameter is used only to compute the resource daily work time and the tour can finish at midnight. * If the optimumStartTime parameter is set to False, then this parameter is used to compute the resource daily work time and the tour can finish no later than the specified end time plus the values of the first overtime duration and the second overtime duration constraints. If the daily work duration is specified by the workEndTime constraint's value, then it is equal to workEndTime minus workStartTime and lunchDuration. Example : For a vehicle to work for 6 hours between 7PM and 7AM, set * optimumStartTime=True * dailyWorkTime="06:00:00" * workStartTime="07:00:00" * workEndTime="19:00:00" For a vehicle to work for 8 hour between 8PM and 5AM with a one-hour break at 12:30, set * workStartTime="08:00:00" * workEndTime="17:00:00" * lunchDuration="01:00:00" * lunchTime="12:30:00" Type : "hh:mm:ss", DateTime. Default: : "24:00:00". Example : `"null"`	
workPenalty *optional*	The cost of a resource working for an hour. Use this constraint to specify the resource wages when working for an hour. Type : float. Default: : 9. Example : If workPenalty = 10 (euros per hour) and daily work time is about 8 hours, the total daily work cost is 10 * 8 = 80 euros.	number
workStartTime *optional*	The start time of the resource work time window. Use this constraint to specify the time at which the tour must start. This time can be adjusted using the optimumStartTime parameter. Type : "hh:mm:ss", DateTime. Default: : "00:00:00". Example : `"null"`	string

Name	Description	Schema
workingDays *optional*	The resource's work days. Use this constraint to specify the days a resource works in the planning period. This constraint is linked with the orders possible visit days constraints of Orders object elements and with the depot days of opening constraints of Depots object elements. Thus, customers and resources must have matching days for deliveries to be possible, and depots must be opened on resources working days to be used. A maximum of 64 days can be defined as work days. Working days can be written as integers (1,2???) or dates (14/05/2010, 15/05/2010, ???). If you are using dates, the oldest date in the workDays constraint defines day 1. Type : string values containing days separated with commas (like "1, 2, 5" or "14/05/2010, 15/05/2010, 18/05/2010" to specify day 1, day 2 and day 5) or intervals (like "1-10", "2=>5" or "14/05/2010=>24/05/2010") where 1 (or 14/05/2010) is the first day of the planning period. For day intervals, prefer the "=>" separator. Be careful, if the separator of date is - then "1-5" corresponds to May 1st of the current year. If you mix integer and date formats, beware that day 1 will all the same be defined by the oldest available date. Default: : "1". Example : You can define a single working day: Specify "1", "3" or "10" for the resource to work on day 1, day 3 or day 10 of the planning period. You can also define working periods. Specify "1-5" or "1=>5" or "14/05/2010=>18/05/2010" for the resource to work on a 5-days period (from day 1 to day 5 included). You can also mix single day and periods. Specify "1, 3=>5" for the resource to work on day 1 and from day 3 to 5 of the planning period. Example : `"null"`	string

(JSON) TSSimulation

Name	Description	Schema
depotProperties *optional*		< string > array
depots *optional*		< json_TSDepot > array
nbCapacities *optional*		number
nbExtraTravelPenalties *optional*		number
nbQuantities *optional*		number
nbTimeWindows *optional*		number
options *optional*		json_TSOptions
orderProperties *optional*		< string > array
orders *optional*		< json_TSOrder > array

Name	Description	Schema
resourceProperties *optional*		< string > array
resources *optional*		< json_TSResource > array

(JSON) TSTimeWindow

Time window

Name	Description	Schema
beginTime *optional*	the minimum begin time Type : Time ("hh:mm" or "hh:mm:ss") Example : `"08:00"`	string
endTime *optional*	the maximum end time Type : Time ("hh:mm" or "hh:mm:ss") Example : `"18:00"`	string

(JSON) TSTour

Details of a stop within a resource tour.

Name	Description	Schema
additionalCost *optional*	Additional cost Additional cost depends on additional cost configuration specified on resources and orders	number
dayId *optional*	day of the tour Example : `"null"`	string
deliveryCost *optional*	Total delivery cost of the tour Delivery cost depends on PenaltyPerVisit defined on resources and configuration specified in options (see countDepotsInDeliveryCost and countVisitCostOnceIfSameLocation)	number
plannedOrders *optional*	Orders planned in this tour	< json_TSPlanned > array
reloadNb *optional*	ReloadNb Number of reloads during this tour	number
resourceCapacities *optional*	List of resource capacities	< number > array
resourceId *optional*	Assigned resource identifier Example : `"null"`	string
totalCost *optional*	Total cost Total cost = (delivery cost) + (additional cost)	number
travelDistance *optional*	drive distance for this tour	number
travelDuration	drive duration for this tour	string

Name	Description	Schema
optional	Example : "null"	
usedCapacities *optional*	List of used capacities Tour used capacities can exceed resource capacities if the tour contains one or several stops to a depot.	< number > array

(JSON) TSTravelPenalty

Name	Description	Schema
distance *optional*	The driven distance (in the solver's measurement system unit) from which distance cost will be applied. Use this constraint when you need to specify a distance cost which can vary according to the covered distance. Up to 4 different distance costs can be specified. Each one must be related to the corresponding distance threshold, from which it will be applied	number
penalty *optional*	The cost for a resource of driving for one distance unit. Use this constraint to specify the average resource taxes (gazoline, wear,???) when driving one distance unit. Type : float Default : 1.5 Example : if penalty = 0.5 (euro per distance unit) and the driven distance is about 100 unit (km or miles), the total distance cost is 0,5 * 100 = 50 euros.	number

(JSON) TSTravelTimeModifier

Name	Description	Schema
length *optional*	Indicates the duration on which to apply the travel time modifier. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : 0 Example : "null"	string
offset *optional*	Indicates the offset of the travel time modifier. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : 0 Example : "null"	string
value *optional*	Indicates the value of the travel time modifier. When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location (tsDepotTravelTimeModifierValue), the portion of the travel time on which the modifier applies (length) an offset to add to any travel duration leaving or reaching the location (offSet). Example : * Set tsResource TravelTimeModifier Value to 1.5, tsResource TravelTime Modifier Length to 300 and tsResource TravelTimeModifier OffSet to 60 for Resource 1	number

Name	Description	Schema
	* Set tsDepot TravelTimeModifier Value to 2, tsDepot TravelTimeModifier Length to 420 and tsDepot TravelTimeModifier OffSet to 0 for Depot 1 If the initial travel duration between Resource 1 and Depot 1 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660 Type : float Default : 1	

(JSON) TSUnplanned

Unplanned order

Name	Description	Schema
reason *optional*	the reason why it what not planned Example : "null"	string
stopID *optional*	The id of the stop. Example : "null"	string

(JSON) TSWarning

Warning message (and associated error code) about possible Orders and Resources elements constraints misconfiguration.

The Solver object checks automatically data configuration (if not done yet) before the optimization begins. Whenever values are detected as incoherent or invalid, warnings are emitted in the Warnings property of the involved element. Depending the warning, constraints value can be replaced by their default value or not.

Name	Description	Schema
constraint *optional*	id of constraint causing the warning	number
constraintName *optional*	Example : "null"	string
i18nMessageCode *optional*	Example : "null"	string
id *optional*	if of object causing the warning Example : "null"	string
message *optional*	warning message Example : "null"	string
messageId *optional*		number
objectType *optional*	type of object causing the warning Example : "null"	string
value *optional*	Example : "null"	string

(JSON) ToursolverServiceResult

generic result of service

Name	Description	Schema
message *optional*	error message Example : `"null"`	string
status *optional*	response status, OK or ERROR	json_Status

(JSON) WebhookExportMode

Type : enum (NONE, ORDERS, TOURS)

(XML) TSAdditionalCost

Polymorphism : Composition

Name	Description	Schema
type *optional*	TSOrder type (stored in a custom data)	string
value *optional*	Cost for this type	number

(XML) TSDepot

The Depots object stores depots data. They are composed of X,Y coordinates, identity tags and a set of constraints.

If you want to specify depot locations or allow a resource to reload at several depots, use the Depots object to add and configure your depot data. The Depots object contains zero or more elements. If it contains no element, the base location of each resource acts as a depot location. Once a depot is associated with a resource its base location is no longer used as a depot location.

Polymorphism : Composition

Name	Description	Schema
TSTimeWindow *optional*	The depot time windows. Use this constraint to specify a depot time window during which the depot is opened. A depot time window is defined by its start time, its end time and days of opening. The start time of a depot time window is the first instant when a resource can enter the depot. Its end time is the last instant when a resource can enter the depot (but it may leave it afterward). You can specify up to 4 depot time windows for each depot. Type : "hh:mm:ss", DateTime. Default : "00:00:00".	xml_ns0_TSTimeWindow
allProductsRequired *optional*	Indicates whether a resource must provide all required products or one at least. Set it to True to indicate that a resource must provide all the depot required products. Set it to False to indicate that a resource must provide at least one of the depot required products. Type : boolean Default : True	boolean
availability *optional*	Indicates whether a depot is included in the planning process or not.	boolean

Name	Description	Schema
	This constraint enables you to easily modify your optimization problem configuration without having to add nor delete your original data. * Set it to True to include a depot element in the planning process. * Set it to False to ignore it: it will not be used in the planning.	
deliveryQuantities *optional*	The available quantities of products available at the depot. You can specify up to 24 quantities Type : float array	number
excludeResources *optional*	The list of resources excluded from the depot. Use this constraint to specify a list of resources that can not use the depot. Type : string array. Default: Empty array: no resource is excluded from the depot	string
fixedLoadingDuration *optional*	The fixed duration for a resource to load at depot. Use this constraint to specify how long takes any resource reload at the depot. You can specify an additional duration according to the quantity to reload, using loadingDurationPerUnit. Type : "hh:mm:ss", DateTime. Default : "00:00:00". Example : tsDepot fixedVisitDuration = "00:30:00" indicates that 30 minutes are needed to load at the depot. tsDepot loadingDurationPerUnit = 120 indicates that 120 seconds are needed to load one unit. If the quantity to reload is 8, for instance, the variable part is 120*8. 16 minutes are required to load at the depot. Total load time = 30 minutes + 16 minutes = 46 minutes accounted for this reload break.	string
id *optional*	The unique identifier of the depot	string
loadingDurationPerUnit *optional*	The time needed to load a unit of product. This constraint is added to the fixed part of the loading duration: it depends on the total quantity to load. Type : "hh:mm:ss", DateTime, Integer (number of seconds). Default : 0.	string
openingDaysList *optional*	The depot days of opening, related to the nth depot time window. Use this constraint to specify the days of operation of a depot referring to a depot time window. It must be related to the tsResourceWorkDays constraint of the resources, as resources work days define the planning period. Depot may be opened on a 64-days long period max, from day 1 to day 64. Type : string value containing days separated with commas (like "1, 2, 5" or "4, 20/05/2010") or intervals (like "1-10", "2=>5" or "22/05/2010=>03/06/2010"). For day intervals, prefer the "=>" separator. 1 is the first day of the planning period, the corresponding date, if any, is defined by the oldest date, for all resources, in the tsResourceWorkDays constraint. No date in the depots days of opening description can precede the first day of the planning period defined by the resources (which is day 1). Default : 1=>64. Example : * Set it to "1" (or "14/05/2010) when a depot is opened on the first day of the planning period. Set it to "1, 2, 4, 5" (or	string

Name	Description	Schema
	"14/05/2010,15/05/2010,17/05/2010,18/05/2010") if depot may not be visited on the third day of a five-days planning period. * Set it to "1=>5" (or "14/05/2010=>18/05/2010") to define a 5-days long period during which the depot is opened.	
pickupQuantities *optional*	The available space for a product available at the depot. You can specify up to 24 values. Type : float array	number
priority *optional*	Depot priority. Use this constraint to specify a priority on the depot. If two depots are nearby the system considers the one with the highest priority **Type : * integer	number
requiredProducts *optional*	The list of the products a depots contains. Use this constraint when a depot must be affected to a specific kind of resource: these resources will have to provide the required required to be able to load at the depot. Type : string (as a list of products separated with commas). Default : none	string
resourceNames *optional*	Lists the resources that can use the depot. Type : string array. Default : Empty array: no resource can use the depot. Example : Specify ["Vehicle 1","Vehicle 2"] to allow only the resources with tags "Vehicle 1" and "Vehicle 2" to use the depot.	string
travelTimeModifier *optional*	Indicates the value of the travel time modifier. When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location (tsDepotTravelTimeModifierValue), the portion of the travel time on which the modifier applies (tsDepotTravelTimeModifierLength) an offset to add to any travel duration leaving or reaching the location (tsDepotTravelTimeModifierOffSet). Example : * Set tsResource travelTimeModifierValue to 1.5, tsResource travelTimeModifierLength to 300 and tsResource travelTimeModifierOffSet to 60 for Resource 1 * Set tsDepot travelTimeModifierValue to 2, tsDepot travelTimeModifierLength to 420 and tsDepot travelTimeModifierOffSet to 0 for Depot 1 If the initial travel duration between Resource 1 and Depot 1 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660 Type : float, Default : 1	xml_ns0_TSTravelTimeModifier
x	longitude WGS84 of depot	number

Name	Description	Schema
optional		
y *optional*	latitude WGS84 of depot	number

(XML) TSEvaluationInfos

Polymorphism : Composition

Name	Description	Schema
orderOriginalResourceId *optional*	The identity of the resource which visits the customer when evaluating an existing planning. * Use this constraint when you want to *evaluate* an existing planning. * Use the orderOriginalVisitDay constraint to specify the working day when the visit must be planned. * Use the orderPosition constraint to specify the position of a visit in the tour. Type : string (storing a unique resource tag). Default: : not used	string
orderOriginalVisitDay *optional*	Indicates the work day when the resource which visits the customer when evaluating an existing planning. * Use this constraint when you want to *evaluate* an existing planning. * Use the orderOriginalResourceID to specify the resource which visit the customer. * Use the orderPosition to specify the position of a visit in the tour. Type : integer between 1 and 64 or string representing a date. 1 is the first day of the planning period, the corresponding date, if any, is defined by the oldest date, for all resources, in the tsResourceWorkDays constraint. if the original visit day is in the date format, this date cannot precede the first day of the planning period defined by the resources (which is day 1). Default : not used if the tsOrderOriginalResourceIDis not used, 1 overwise	string
orderPosition *optional*	The position of a customer in a resource tour. Use this constraint when you want to evaluate a tour. Type : integer. Default : not used	number

(XML) TSObject

Polymorphism : Composition

No Content

(XML) TSOptions

Options for optimization request

Polymorphism : Composition

Name	Description	Schema
allowBridge *optional*	Allow toll If false, all bridges will be avoided. Default : true.	boolean
allowToll *optional*	Allow toll If false, all toll ways will be avoided. Default : true.	boolean
allowTunnel *optional*	Allow toll If false, all tunnels will be avoided. Default : true.	boolean
balanceType *optional*	Route plans balancing type * NONE : no balancing * ORDERS : balance the number of orders (to be used with balanceValue) * HOURS : balance the route plans duration * QUANTITY : balance the route plans delivered quantity Only available with OTSolver	xml_ns0_balanceType
balanceValue *optional*	Route plans balancing target (to be used with balanceType) Locked route plans are not considered. Only available with OTSolver	number
countDepotsInDeliveryCost *optional*	Depot cost configuration for tour delivery cost Specifies how depots stops are counted in tour delivery cost (using PenaltyPerVisit defined on resources). Possible values are : * NONE : depots stops are not counted in tour delivery cost * ALL : each depot stop counts as one visit in tour delivery cost * ALLBUTFIRST : first depot stop is ignored but each following depot stop will count as one visit in tour delivery cost * FIRST : only first depot stop is counted in tour delivery cost Default : NONE.	xml_ns0_depotCostMode
countVisitCostOnceIfSameLocation *optional*	If true, PenaltyPerVisit will counted only once in tour delivery cost if several visits at the same location are planned consecutively.	boolean
distanceType *optional*	Set the distance unit (meters, feet, km or miles), defaut is meters	xml_ns0_distanceType
evaluation *optional*	Enable this if you just want to evaluate (get cost, distances, etc) a set of tours. Therefore, you must fill the evaluationInfos objects in orders Default : false. (see TSEvaluationInfos)	boolean
excludeVisitCostIfMaxAdditionalCost *optional*	Exclude visit delivery cost for visits having the maximum additional cost	boolean
fuelCode	Vehicles fuel type. Possible values are :	string

Name	Description	Schema
optional	* diesel * undefined * unleaded	
maxOptimDuration *optional*	maximum optimization time. Default is one minute. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes)	string
noReload *optional*	Use it to forbid/allow reloads Default : false.	boolean
reloadDuration *optional*	Default reload duration Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes)	string
routingMethod *optional*	Routing method Specifies routing method used for travel time and distance matrix computation Possible values are : * TIME : routes minimizing travel time are used * DISTANCE : routes minimizing travel distance are used Default : TIME.	xml_ns0_routingMethod
sendResultToWebhook *optional*	Enable result export to webhook If enabled, optimization result will be sent to the webook you defined in Toursolver Cloud configuration. Possible values are : * NONE : result will not be sent automatically when optimization finishes. You will have to call the result or toursResult service to retrieve it. * ORDERS : the "orders" result will be sent to the specified webhook. See OptimResultResult object for details. * TOURS : the "tours" result will be sent to the specified webhook. See OptimToursResult object for details. Default : NONE.	xml_ns0_webhookExportMode
speedPattern *optional*	Speed pattern name Specifies the speed pattern to be used for this optimization. "default_profile" is the internal name for the default speed pattern (its actual name in UI depends on your locale). You can use your own speed pattern (defined in Toursolver UI) but you will have to use its internal name : if you created a pattern called "Paris august speed pattern", its internal name will be "paris_august_speed_pattern". If set to null (or not defined), no speed pattern will be used at all (same speeds for whole day).	string
startFromEvaluationInfo *optional*	Enable this if you want the optimization to start from a specified initial configuration. Therefore, you must fill the evaluationInfos Default : false.	boolean

Name	Description	Schema
	objects in orders (see TSEvaluationInfos).	
teamId *optional*	If you did not specify any resource in your request, all resources defined in your tenant will be used for this optimization. If you specify a team identifier, only the resources of this team will be used.	string
useForbiddenTransitAreas *optional*	Whether to use forbidden transit areas Specifies true to use forbidden transit areas, and false to ignore forbidden transit areas If omitted, default to true, but has no effect if no forbidden transit areas are supplied	boolean
useOTSolver *optional*	Whether to use OTSolver instead of TSDK OTSolver is the new optimization engine behind Toursolver. It behaves better with big problems (more than 1000 visits). This is still a beta version and all the constraints supported by TSDK are not supported yet but they will be implemented as soon as possible. OTSolver must be enabled for your account before you can use it.	boolean
vehicleCode *optional*	Vehicles regulations. Possible values are : * bicycle * bus * car * deliveryIntermediateVehicle * deliveryLightCommercialVehicle * emergencyTruck * emergencyVehicle * intermediateVehicle * lightCommercialVehicle * pedestrian * taxi * truck If not specified, vehicle type defined in UI will be used	string

(XML) TSOrder

The Order object stores order data.

Order is composed of X,Y coordinates, identity tags, a set of constraints and status reports.

Use the Orders object to specify your orders data. These elements can be considered as deliveries, pick-ups or commercial visits occuring at a specific locations.

The optimization process consists in assigning Orders elements to Resources elements in a way that respects the constraints of every element and minimizes the total cost of the planning.

Polymorphism : Composition

Name	Description	Schema
TSTimeWindow *optional*	The visit time windows. Use this constraint to specify a visit time window during which the order can be delivered. A visit time window is defined by its start time, its end time and possible visit days (see tsOrderTimeWindow1_EndTime, tsOrderPossibleVisitDays1). You can specify up to 4 visit time windows for a customer. Type : array of time windows Default : "00:00:00". Example : A customer can only be delivered: * * on Monday from 11:00:00 to 12:00:00 AM and from 04:00:00 to 05:00:00 PM, * on Tuesday, Wednesday, Thursday from 08:00:00 to 09:00:00 AM and from 04:00:00 to 05:00:00 PM, * on Friday from 08:00:00 to 09:00:00 AM. Its visit time windows will be specified as follows: * first time window StartTime = "08:00:00", EndTime = "09:00:00", first set of possible days = "2=>5", * second time window StartTime = "11:00:00", EndTime = "12:00:00", second set of possible days = 1, * third time window StartTime = "16:00:00", EndTime = "17:00:00", third set of possible days = "1=>4".	xml_ns0_TSTimeWindow
active *optional*	Indicates whether the order should scheduled or not Default : True.	boolean
allSkillsRequired *optional*	Indicates whether a resource must provide all required skills or one at least. * Set it to True to indicate that a resource must provide all the customer required skills. * Set it to False to indicate that a resource must provide at least one of the customer required skills. Default : True.	boolean
courierPenalty *optional*	The cost of the order delivered by an independant transport device. Use this constraint when an order can be delivered independantly to specify the cost of the device. The solver engine will assign it to a courier device or integrate it in a tour according to the least cost. Type : float. Default : not used.	number
customDataMap *optional*	Private feature data.	object

Name	Description	Schema
	Use this feature when you need to keep accessible private data about customers. For instance if you need to export the address of an element in a planning report, store it there. These informations can also be exported to the mobile app. max 50 fields Default : empty.	
customerId *optional*		string
delayPenaltyPerHour *optional*	The penalty per hour for being late at a customer. Use this constraint to allow a smarter control, with respect to time windows, than with the use of the tsOrderPunctuality constraint. Type : float. Default : not used.	number
email *optional*		string
evaluationInfos *optional*	first order assignment (specifying resource, day and position in route). Use it to compare optimization result with a previous optimization Warning : used only if startFromEvaluationInfo is set to true in options.	xml_ns0_TSEvaluationInfos
fixedVisitDuration *optional*	The fixed part of the total time spent visiting a customer. Use this constraint to specify the minimum time spent at a customer's. Use it to store the time needed by a vehicle to park, for instance, or the time planned for a commercial meeting. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used. Example : * tsOrderFixedVisitDuration = "00:30:00" indicates that the visit will be 30 minutes long. * tsOrderUnloadingDurationPerUnit = 120 indicates that 120 seconds are necessary to unload one unit. With a tsOrderQuantity equal to 8, for instance, the variable part is 120*8 (960 seconds or 16 minutes) to unload the customer ordered quantity. * Total delivery time = 30 minutes + 16 minutes = 46 minutes accounted for the customer delivery.	string
frequency *optional*	The visit occurrence within a defined period. Use this constraint to plan multiple visits at a customer during a period. Type: string value describing the number of visits over a period. Default: not used. Example : Frequency = "3/5" means a customer must be visited 3 times within a 5-days long period. The solver engine will create and optimize routes respecting an	string

Name	Description	Schema
	optimal duration between consecutive visits. Thus, back to the example, the computed period is 5/3 rounded = 2. The best solution will respect, as far as possible, a 2-days long period between each visit (ex: day1, day3, day5 is a good solution).	
getNotifications *optional*	Boolean.True if a visit can receive notification email/sms	boolean
id *optional*	The unique identifier of the order	string
label *optional*		string
minDuration *optional*	The min duration of the visit that enables partition. When the total duration of an order is very long, you may want to part it into multiple consecutive visits. First use this constraint to specify a duration threshold from which visit can be cut into multiple parts. Then use the tsOrderMinPartDuration to specify the min duration of one part. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used.	string
minPartDuration *optional*	The min duration of a part of a split visit. When the total duration of an order is very long, you may want to part it into multiple consecutive visits. First use the tsOrderMinDuration to specify a duration threshold from which the visit can be cut into multiple parts. Then use this constraint to specify the min duration of one part. The visit is split into parts of the set duration, and eventually a bigger one. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used.	string
phone *optional*		string
possibleVisitDaysList *optional*	The possible visit days related to each visit time window. Use this constraint to specify the possible visit days of a customer referring to a visit time window. It must be related to the tsResourceWorkDays constraint of the resources, as resources work days define the planning period. You can plan visits on a 64-days-long period max. Type : array of string values containing visit days separated with commas (like "1, 2, 5" or "4, 20/05/2010") or intervals (like "1-10", "2=>5" or "22/05/2010=>03/06/2010"). For day intervals, prefer the "=>" separator. Be carrefull, if the separator of date is - then "1-5" corresponds to May the 1st of the current year. 1 is the first day of the planning period, the corresponding date, if any, is defined by the oldest date, for all resources, in the tsResourceWorkDays constraint. No date in the possible visit days description can precede the first day of the planning period defined by the resources (which is day 1). Default: 1=>64. You can set up to 4 sets of visit days in this array. If you define several time windows, each time window will apply to corresponding set of possible days : first time window applies to first possible days set, second time window applies to second possible days set, etc.	string

Name	Description	Schema
	Example : * Set it to "1" (or "14/05/2010) when a customer must be visited on the first day of the planning period. * Set it to "1, 2, 4, 5" (or "14/05/2010,15/05/2010,17/05/2010,18/05/2010") if the customer cannot be visited on the third day of a five days planning period. * Set it to "1=>5" (or "14/05/2010=>18/05/2010") to define a 5-days long period during which the customer can be visited.	
punctuality *optional*	The priority for a customer to be delivered on time. Use this constraint to introduce different priority levels for the respect of visit time windows (when defined). Specify a value from 1 to 5 to set the priority: customers with a punctuality set to 5 will have less risk of delayed deliveries than a customer with a punctuality set to 1. You can specify more precisely punctuality requirements to enforce delay control by using the tsOrderDelayPenaltyPerHour constraint instead of the tsOrderPunctuality constraint. Type : integer value from 1 to 5. Default : 1.	number
quantity *optional*	The requested product quantities for an order. Use this constraint to specify the product quantities ordered by a customer, for instance. Up to 24 quantities can be used to order different products. Useless when planning commercial tours. Type : float array (maximum of 3 significant digits after decimal separator). Default : 0 Max : 2,147,483. Example : * Use this constraint to store the number of ordered packages, when working with packages. * Use this constraint to store the number of ordered liters, when working with liquids. * Use multiple dimensions when working with multiple products: when delivering fuels for instance, you will use as many dimensions as types of fuels Warning : The quantity constraint of the *Orders* elements must be related to the corresponding capacity constraint of the *Resources* elements: thus, an order will not be planned whenever no resource with the related capacity has been defined.	number
requiredSkill *optional*	The list of the skills a customer requires to be visited. Use this constraint when an order must be affected to a specific kind of resource: these resources will have to provide the required skills to be able to visit the customer. Type : string array. Default : none. Example :	string

Name	Description	Schema
	* Specify the word "Maintenance" as a required skill for a maintenance visit. Only the resources providing the "Maintenance" skill can perform the visit. * Specify "Small vehicle" as a required skill for a customer living down town.	
resource *optional*	The identities of the resources which must not deliver a customer. Use this constraint when you want to prevent some customers from being delivered by specific resources.	string
resourceCompatibility *optional*	Value for compatibility with resources. If the order has a compatibility constraint value, a resource is compatible with the order if its compatibility value is either non-existent either larger or equal to the order's one. See tsResourceOrderCompatibility. Example : * Set tsOrderResourceCompatibility to 254.18 and tsResourceOrderCompatibility to 260.12: order and resource are compatible * Set tsOrderResourceCompatibility to 254.18 and tsResourceOrderCompatibility to 200: the order can not be affected to the resource * Set tsOrderResourceCompatibility to 254.18 and tsResourceOrderCompatibility empty: order and resource are compatible * Set tsOrderResourceCompatibility empty and tsResourceOrderCompatibility to 260.12: order and resource are compatible Type : double Default : -1	number
sequenceNumber *optional*		number
travelTimeModifier *optional*	Indicates the value of the travel time modifier. When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location (tsOrderTravelTimeModifierValue), the portion of the travel time on which the modifier applies (tsOrderTravelTimeModifierLength) an offset to add to any travel duration leaving or reaching the location (tsOrderTravelTimeModifierOffSet). Example : * Set tsOrderTravelTimeModifierValue to 1.5, tsOrderTravelTimeModifierLength to 300 and tsOrderTravelTimeModifierOffSet to 60 for Order 1 * Set tsOrderTravelTimeModifierValue to 2, tsOrderTravelTimeModifierLength to 420 and tsOrderTravelTimeModifierOffSet to 0 for Order 2 If the initial travel duration between Order 1 and Order 2 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660 Type : float	xml_ns0_TSTravelTimeModifier

Name	Description	Schema
	Default : 1	
tsOrderBefore *optional*		string
tsOrderBeforeMaxTimeSpacing *optional*		string
tsOrderBeforeMinTimeSpacing *optional*		string
tsOrderFixed *optional*	If true force the use of the specified day, resource and start time	boolean
tsOrderLastVisit *optional*	The number of days before the beginning of the planning the last visit belonging to the order has been performed Use this constraint in case of a frequency order to specify from how many days the last visit was performed before starting the current planning. The default value indicates this constraint is to be unused in the planning. Type : negative integer Default : 0 Max : -2,147,483. Example : If *tsOrderLastVisit* = -2, the last visit of the same order has been performed two days ago. Warning : if the number of occurences of the frequency is more than one, the constraints tsOrderLastVisit and tsOrderMinimumSpacing are not considered.	number
tsOrderMaximumSpacing *optional*	The maximum number of days required between two visits of a same order. Use this constraint in case of frequency order to specify how many days at most should be left between two visits of the order. Type : integer Max : 2,147,483.	number
tsOrderMinimumSpacing *optional*	The minimum number of days required between two visits of a same order. Use this constraint in case of frequency order to specify how many days at least should be left between two visits of the order. Type : integer Default : 0 Max : 2,147,483. Warning : if the number of occurences of the frequency is more than one, the constraints tsOrderLastVisit and tsOrderMinimumSpacing are not considered.	number
type *optional*	The type of visit at a customer. * 0 (Delivery): the resource will delivere related quantities. * 1 (PickUp): the resource will pick-up related quantities. * 2 (DeliveryFirst): the resource will not mix deliveries and pick-ups in a same rotation. It is possible to mix pickups and 0 type deliveries in the same rotation. However, the delivered products will come from the depot and the pickup	number

Name	Description	Schema
	products will be unloaded at the depot. No product delivered to a customer can come from a pickup at another customer's. Type : Integer. Default : 0 (Delivery).	
unloadingDurationPerUnit *optional*	The time needed to deliver one unit of product. Use this constraint when the time spent at a customer depends on the quantity to deliver. It is the time needed to unload one unit of the quantity stored in the customer tsOrderQuantity constraint. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : not used.	string
wholeVisitInTimeWindow *optional*	Whole visit ends before time window end. Use this constraint if you want that a visit ends before the end of the time window. Default : false.	boolean
x *optional*	longitude of order location	number
y *optional*	latitude of order location	number

(XML) TSPause

Polymorphism : Composition

Name	Description	Schema
duration *optional*	Duration Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes).	string
end *optional*	maximum end time Type : Time ("hh:mm" or "hh:mm:ss")	string
start *optional*	minimum start time Type : Time ("hh:mm" or "hh:mm:ss")	string

(XML) TSPlanned

Details of a stop within a resource tour.

Polymorphism : Composition

Name	Description	Schema
dayId *optional*	day of the stop	string
resourceId *optional*	Assigned resource identifier	string
stopDriveDistance *optional*	The drive distance from the previous to the current stop.	number

Name	Description	Schema
stopDriveTime *optional*	The drive time from the previous to the current stop. Type : Time ("hh:mm" or "hh:mm:ss")	string
stopDuration *optional*	The duration of the stop. Type : Time ("hh:mm" or "hh:mm:ss")	string
stopElapsedDistance *optional*	The tour cumulated driven distance at the stop.	number
stopId *optional*	the ID of the stop or order	string
stopPosition *optional*	The position of the stop in the resource tour.	number
stopStartTime *optional*	The time the stop starts at. Type : Time ("hh:mm" or "hh:mm:ss")	string
stopStatus *optional*	The status of the stop. An order can be on time, late, waited, impossible (non compliant). It can also be a courier service delivery and not planned. See StopStatusConstants for more information about stop status. Type : integer. * 0 = OnTime: The stop begins into the specified visit time window. * 1 = Late: The stop begins after the specified visit time window. * 2 = Waited: The resource waits at the stop for the specified stop start time. * 3 = Impossible: The stop cannot be planned. * 4 = BadDay: The stop is planned without respect of frequency. * NotPlanned: The courier service is in charge of a stop or the stop is not included in the planning process. * BadPrevious : When the visits are split, the parts should follow one another. This status is that of a part following another visit instead of the previous part of the split visit.	number
stopType *optional*	The stop type A tour is composed of stops, which can be either customers visits or breaks, like reload breaks, lunch breaks, or even night breaks. StopTypeConstants are * 0 = Visit: The stop is an element of the Orders object. * 1 = Start: The stop is the initial stage (departure) of the tour, occurring at the resource’s start location. * 2 = End: The stop is the the last stage (arrival) of the tour, occurring at the resource’s stop location. * 3 = EndBeforeOvernight: The stop is the the last stage (arrival) of the tour, occurring where the resource stopped for the night.	number

Name	Description	Schema
	* 4 = ReloadBreak: The stop is a reload break within the tour, occurring at a depot location. * 5 = LunchBreak: The stop is the lunch break within the tour, occurring anywhere between two other located stops. * 6 = RestBreak: The stop is a rest break within the tour, occurring anywhere between two located stops. * 7 = WaitBreak: The stop is a wait break within the tour, occurring before a visit. * 8 = StartAfterOvernight: The stop is the last stage (arrival) of the tour, it occurs where the resource stopped for the night. tsStopDriveDistance * 9 = Briefing: The stop is a briefing break within the tour, occurring right after the departure of the tour. * 10 = DeBriefing: The stop is a wait break within the tour, occurring right before the arrival of the tour.	
stopX *optional*	The X coordinate of the stop.	number
stopY *optional*	The Y coordinate of the stop.	number

(XML) TSResource

The Resources object stores the collection of the elements which will perform deliveries, pick-ups, commercial visit, etc.

Each element can be seen as a driver/vehicle pair and their related driving and working constraints.

It is composed of an identity tag, and location coordinates (base or depot, departure, arrival location). Eventually, each element can be configured through a set of constraints which stores information like capacity, driver work time, penalties and so on.

Polymorphism : Composition

Name	Description	Schema
additionalCostOperator *optional*	Operator used to compute additional cost (see toursResult object) Possible values are : * SUM : tour additional cost will be the sum of additional costs of planned orders * MAX : tour additional cost will be the maximum value of additional costs of planned orders * MIN : tour additional cost will be the minimum value of additional costs of planned orders * AVERAGE : tour additional cost will be the average of additional costs of planned orders	xml_ns0_costOperator
additionalCostOrderCustomDataTSOrder *optional*	Name of the order customData that will contain the TSOrder type used for additional cost computation (see toursResult object)	string

Name	Description	Schema
additionalCosts *optional*	List of additional cost (cost per order type)	xml_ns0_TSAdditionalCost
available *optional*	Indicates whether a resource is included in the planning process or not. This constraint enables you to easily modify your optimization problem configuration without having to add nor delete original data. * Set it to True to include a resource element in the planning process. * Set it to False to ignore it: it will not be used in the planning. Type : Boolean. Default : not used.	boolean
avgConsumption *optional*	Average consumption of the resource. Use this constraint to specify the average consumption of the resource. This consumption must be defined in liters per 100 distance unit (km or miles) Default : The consumption indicated in the VehicleProfile.Consumption property.	number
briefingDuration *optional*	An additional duration applied before the tour starts. Use this constraint to specify an additional duration before the resource begins the tour. For instance, to brief drivers before they leave or to simulate an initial vehicle loading time (no depot). Type : "hh:mm:ss", DateTime. Default : "00:00:00".	string
capacity *optional*	The total product capacity a resource can carry. Use this constraint to specify the amount of product a resource can deliver or pick-up. It must be linked with the customers related quantity constraint. 23 more dimensions can be used in order to specify various products. They must be linked with the customers related dimensions. Type : float (maximum of 3 significant digits after decimal separator). Default: None. Max : 2,147,483. Example : * Use it to store the total number of packages a resource can deliver before reloading, when working with packages. * Use it to store the total number of litres a resource can deliver before reloading, when working with liquids. * Use multiple capacity dimensions when working with multiple products: when delivering fuels, you will use as many capacities as types of fuels.	number
customDataMap *optional*	Private custom data. Use this feature when you need to keep attributes values of your own into the optimization context. For instance if you need the address of an element, store it there.	object
dailyWorkTime *optional*	The maximum resource work duration over a day. Use this constraint to specify the maximum daily work duration (unless overtime periods have been defined). If the dailyWorkTime constraint is not set to a value, default daily work time is computed from resource time window and lunch duration (workStartTime, workEndTime and lunchDuration). Type : "hh:mm:ss", DateTime. Default : not used.	string

Name	Description	Schema
debriefingDuration *optional*	An additional duration applied at the end of the tour. Use this constraint to specify an additional duration after the resources stops at the end location. For instance, to debrief drivers after their tour or to account an ultimate vehicle unloading time, or a parking time. Type : "hh:mm:ss", DateTime. Default : "00:00:00".	string
driveRestAtCustomer *optional*	Indicates whether or not time spent at customer during deliveries can be considered as rest break. * Set it to True if a visit can be considered as a rest break, as soon as its duration is equal or higher than the legalMinRestDuration constraint. * Set it to False if not. Type : Boolean. Default : False.	boolean
driveRestAtDepot *optional*	Indicates whether time spent at depot during a reload break can be considered as drive break or not. Use this constraint to take into account drive time legislation in your planning results. * Set it to True if load breaks can be considered as drive breaks, as soon as their duration is equal or higher than the legalMinRestDuration constraint. * Set it to False if not. Type : Boolean. Default : False.	boolean
endX *optional*	longitude of resource arrival Specify it only if arrival and start are not the same.	number
endY *optional*	latitude of resource arrival Specify it only if arrival and start are not the same.	number
extraTravelPenalty *optional*	The costs for a resource of driving for one distance unit.	xml_ns0_TSTravelPenalty
fixedLoadingDuration *optional*	The fixed duration of the resource load break. Use this constraint to specify how long takes a resource reload at depot. You can specify an additional duration according to the quantity to reload, using loadingDurationPerUnit. Type : "hh:mm:ss", DateTime. Default : "00:00:00". Example : * fixedVisitDuration = "00:30:00" indicates that 30 minutes are needed to load the resource. * loadingDurationPerUnit = 120 indicates that 120 sec are needed to load one unit. If the quantity to reload is 8, for instance, the variable part is 120*8. 16 minutes are required to load the resource. * Total load time = 30 minutes + 16 minutes = 46 minutes accounted for this reload break.	string
fixedUnloadingDuration *optional*	The fixed time needed for the resource to deliver/pick-up at the customer's place.	string

Name	Description	Schema
	This fixed resource's unloading duration is added to the resource's unloading duration depending on the quantity to deliver/pick-up and to the order's unloading duration depending on the quantity to deliver/pick-up. See also tsResource unloadingDurationPerUnit and tsOrder unloadingDurationPerUnit. Example : * Set tsOrderQuantity to 3. * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 6 mn * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration to 10 mn and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 10 mn * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration empty and tsResource unloadingPerUnit to 1 mn: the variable part of the order is 3 mn Type : "hh:mm:ss", DateTime. Default : "" Warning : the fixed/per unit unloading duration for resource is not compatible with the orders constraints: partition and whole visit in time window. Hence if one of the constraint at least tsResource fixedUnloadingDuration and tsResource unloadingDurationPerUnit is non null, partition and whole visit in time window are disabled.	
fuelCode *optional*	Vehicles fuel type. Possible values are : * diesel * undefined * unleaded	string
fuelType *optional*	The fuel type of the resource. Use this constraint to specify the fuel type of the resource. Available fuel types are : * 0: Unknown fuel type * 1: No fuel type (pedestrian) * 2: Diesel * 3: Unleaded fuel * 4: LPG Type : Integer. Default : The fuel type indicated in the VehicleProfile.FuelType property.	number
globalCapacity *optional*	Global capacity for a resource. This constraint unables to set a maximum capacity on a resource, whatever product is carried. If both global capacity and capacity per product are to be used, set True for the value of *useAllCapacities*. Type : Float Default : None Max : 2,147,483.	number

Name	Description	Schema
	Example : Set globalCapacity = 15 and useAllCapacities = False. The global capacity only is considered: the vehicle can carry 15 quantities of products, whatever product is to be carried.	
id *optional*	The unique identifier of the resource. This id can not contain special characters like = or *:*	string
legalDailyDriveDuration *optional*	The legal daily duration a resource can drive before having a rest break. Use this constraint to take into account drive time legislation in your planning results. Specify the rest duration in the legalDailyRestDuration: rest breaks occur as soon as the resource has driven the legal daily driving duration or has completed its daily work time. Type : "hh:mm:ss", DateTime. Default : not used. Example : legalDailyDriveDuration = "09:00:00", legalDailyRestDuration = "11:00:00". The resource can cumulate a 9-hours daily drive time before having to rest during 11 hours.	string
legalDailyRestDuration *optional*	The legal rest duration a resource must have after the daily max drive duration. Use this constraint to take into account drive time legislation in your planning results. Rest breaks occur as soon as the resource has driven the legal max daily duration or has complete its daily work time. The use of both legalDailyDriveDuration and legalDailyRestDuration implies that the start and end time of tours are no longer defined by the workStartTime and workEndTime constraints, but may vary as long as the legalDailyRestDuration constraint is respected. Type : "hh:mm:ss", DateTime. Default : not used.	string
legalDriveRestDuration *optional*	The resource break duration after max drive duration. Use this constraint to take into account drive time legislation in your planning results. Type : "hh:mm:ss", DateTime. Default : not used.	string
legalMaxDriveDuration *optional*	The legal max duration a resource can drive without a break. Use this constraint to take into account drive time legislation in your planning results. Type : "hh:mm:ss", DateTime. Default: : not used. Example : * legalMaxDriveDuration = "04:30:00", legalDriveRestDuration = "00:45:00". The resource can drive for 4 hours and 30 minutes before having a 45 minutes break. Then it can start driving again for 4 hours and 30 minutes???etc. * If added driveRestAtCustomer = True and legalMinRestDuration = "00:15:00", rest time is cumulated as soon as customer visit duration is equal or higher than the specified min rest duration. If the 45 minutes rest break has not occured when the resource has driven for 4 hours and 30 minutes, it must stop to complete the break. Otherwise, it can drive again for 4 hours and 30 minutes.	string
legalMinRestDuration *optional*	The minimum duration a resource breaks for it to be considered as a drive break. Use this constraint to take into account drive time legislation in your planning results. When breaks occur at customer or at depot, they are considered as drive breaks if their duration is equal or higher than the specified min rest duration.	string

Name	Description	Schema
	Type : "hh:mm:ss", DateTime. Default: : not used.	
loadBeforeDeparture *optional*	Indicates whether a resource must load before starting a tour or not. * Set this constraint to True to insert a load break at depot before the tour starts. * Set it to False to consider the vehicle loaded. Type : Boolean. Default: : False.	boolean
loadOnReturn *optional*	Indicates whether a resource must reload after a tour or not. * Set this constraint to True to insert a load break at depot after the tour ends. * Set it to False to not reload the vehicle. Type : Boolean. Default: : False.	boolean
loadingDurationPerUnit *optional*	The time needed to load a unit of product. This constraint is added to the fixed part of the loading duration: it depends on the total quantity to load. Type : "hh:mm:ss", DateTime, Integer (number of seconds). Default: : 0.	string
lunch *optional*	The lunch break Default: : no lunch break	xml_ns0_TSPause
maxNightsOutPerJourney *optional*	The number of nights a resource can spend out to perform deliveries before coming back to base. Use this constraint to enable night breaks trip: it will store how many days a resource can drive away from depot. Unless the vehicle needs to reload, it will not come back before. Type : integer. Default: : not used. Example: maxNightsOutPerJourney = 4, workDays = 1=>5. The resource can perform deliveries during 5 days (and 4 nights) without coming back to depot. Starting on day 1, it must come back to depot on day 5.	number
maximumDistance *optional*	The maximum distance the resource should travel per day. Use this constraint to specify the distance the resource should not exceed in one day Type : integer Default : -1 (no limit)	number
maximumReloads *optional*	Value for maximum number of reloads per day.	number
maximumReloadsPenalty *optional*	Value of the penalty if the resource exceeds the maximum number of reloads per day. If the resource exceeds the maximum number of reloads per day, it will be penalized by the maximum reloads penalty multiplied by the number of reloads' overtaking. The default value is 1024 if the constraint maximumReloads is set. See also maximumReloads.	number
maximumVisits *optional*	The maximum number of visits the resource can perform in one day. Type : integer Default : -1 (no limit) Max : 2,147,483.	number
minDriveDuration *optional*	The minimum laps of time between 2 consecutive visits.	string

Name	Description	Schema
	Use this constraint to specify a minimum drive time accounted when consecutive visits are very close, in order to compute a realistic timing. Type : "hh:mm:ss", DateTime. Default: : not used.	
minimumQuantity *optional*	Value for minimum quantity to deliver. If the resource has a non null minimum quantity to deliver constraint value, the resource can visit on order only if the quantity to be picked-up or delivered at this order is higher than the constraint value. *Example : * * set order *quantity* to 3 and resource *minimumQuantity* to 2 : resource might visit the order * set order *quantity* to 3 and resource *minimumQuantity* to 5 : resource can not visit the order Type : float	number
mobileLogin *optional*	Mobile login to be specified only if you need to be able to export the optimization result to operation planning from TsCloud GUI. If you trigger operational planning export from the API, you will still need to specify the mapping between resource name and login.	string
nightPenalty *optional*	The night cost of a resource when planning night breaks. Use this constrainte to specify how much a night out costs (lunch, bed, breakfeast???) for a resource when night breaks are allowed. Type : float. Default: : 0	number
noReload *optional*	Constraint providing resource to reload at depot during a tour. If none of the constraints loadBeforeDeparture and loadOnReturn is activated, the resource is never loading at depot. If one of these constraint is activated the resource is loading once per tour, either at the beginning of the tour or at the end of it. If both constraints are activated then the resource is loading twice a tour, at the beginning and at the end. With standard solver only : solver always works with solution cost comparison, so it may allow reloads even if you set noReload to true if the solution cost is really lower. The only way to totally prevent reloads is to set fixedLoadingDuration to a very long duration (20:00:00 for instance). The solver will warn you about this long duration but will not add reloads to the tours. Caution : Don’t do this if you are using loadOnReturn or loadBeforeDeparture Default : False	boolean
nonUsePenalty *optional*	A cost paid for NOT using a resource in the computed planning. Use this constraint to penalize the fact that an available resource is not used in the planning. It proves usefull when you must use all the resources in the Resources collection: you will prevent the solver from optimizing the number of resources, as using them would be too expensive a solution. Type : float. Default: : not used. Example : Use it to specify parking taxes when a vehicle performs no tour.	number

Name	Description	Schema
openDistanceStart *optional*		boolean
openDistanceStop *optional*		boolean
openStart *optional*	Indicates whether or not a tour evaluation starts from the resource start location or from the first customer's place. * Set it to True to begin the tour at the first customer (no cost will be computed between the resource start location and the first customer). * Set it to False to begin the tour at the resource start location. Type : Boolean. Default: : False.	boolean
openStop *optional*	Indicates whether or not a tour evaluation ends at the last customer's place or at the resource end location. * Set it to True to finish the tour at the last customer location (no cost will be computed between the last customer and the resource end location). * Set it to False to finish the tour at the resource end location. Type : Boolean. Default: : False.	boolean
openTimeStart *optional*		boolean
openTimeStop *optional*		boolean
optimumStartTime *optional*	Adapts the resource start time to the best solution. * Set this constraint to False to start the tour at the time indicated by the workStartTime constraint. * Set this constraint to True to optimize the resource start time and enable the use of the dailyWorkTime constraint. Notice that, in both cases, the tour will not start before the time stored in the workStartTime constraint and that the use of the weeklyWorkTime constraint is enabled. Type : Boolean. Default: : False.	boolean
otherWorkEndTime *optional*	The end time of the resource work time window for the "other working days" (see otherWorkDaysList). If you defined other working days, use this constraint to specify the time at which the tour must end. These times can be adjusted using the optimumStartTime parameter. You can define up to 3 different end times here, corresponding to the different "other working days" slots. Type : List of "hh:mm:ss", DateTime. Example : ["18:00:00","19:00:00]	string
otherWorkStartTime *optional*	The start time of the resource work time window for the "other working days" (see otherWorkDaysList).	string

Name	Description	Schema
	If you defined other working days, use this constraint to specify the time at which the tour must start. These times can be adjusted using the optimumStartTime parameter. You can define up to 3 different start times here, corresponding to the different "other working days" slots. Type : List of "hh:mm:ss", DateTime. Example : ["08:00:00","09:00:00]	
otherWorkingDay *optional*	Other working days (see workingDays) You can define up to 3 different working days slots. If you do so, you can then specify distinct work start and end times with otherWorkStartTimes and otherWorkEndTimes. Type : List of string (see workingDays format) Example : ["2","3=>5"]	string
overnightMinDriving *optional*	The max duration a resource can drive to go back to base at the end of a tour when working with night breaks. Use this constraint when a resource can end a tour at base instead of having a night break on the road. Type : "hh:mm:ss". Default: : not used.	string
overtimeDuration *optional*	The duration of the resource overwork periods. (max 2 periods) Use this constraint when you need to define 2 different overwork periods with regard to daily work time and their related surcharge. No overwork period can be defined with regard to weekly work time. Type : "hh:mm:ss", DateTime. Default: : not used (no second overtime period). Example : workPenalty = 10, first overtime duration = "02:00:00", first overtime penalty = 5,second overtime duration = "01:00:00", second overtime penalty = 10. The cost of the resource during the daily working time is 10 euros per hour. The resource can work during 3 more hours and each overworked hour will cost 10 + 5 euros for 2 hours and 10 + 10 euros for the last hour.	string
overtimePenalty *optional*	A surcharge for a resource’s overwork period (max 2 periods). Use this constraint to specify additional cost to workPenalty as overwork time. You can define up to 2 overwork periods with regard to daily work time and their related surcharge. No overwork period can be defined with regard to weekly work time. Type : float. Default: : not used. Example : workPenalty = 10, first overtime duration = "02:00:00", first overtime penalty = 5. The cost of the resource during the daily working time is 10 euros per hour. The resource can work during 2 hours after the daily working time and each overworked hour will cost 15 (10 + 5) euros.	number

Name	Description	Schema
payWholeDay *optional*	Indicates whether or not a whole work day cost is computed as soon as a resource works even a few hours. * Set this constraint to True to account a whole day cost. * Set it to False to account only effective worked hours. Type : Boolean. Default: : False.	boolean
penaltyPerVisit *optional*	An additional fixed cost applied each time the resource performs a visit. Use this constraint when the cost of a resource may vary according to the number of visits it performs. Type : float. Default: : 0.	number
providedProducts *optional*	The list of products provided by a resource. This constraint is linked with tsDepot requiredProducts constraint: a depot with a required product can only be visited by a resource providing it. Type : string as a list of products separated with commas. Default : not used. Example : Specify "Oxygen" in the resource can provide oxygen.	string
providedSkills *optional*	The list of characteristics provided by a resource. This constraint is linked with the same order constraint: a customer with a required skill can only be delivered by a resource providing it. Type : string as a list of characteristics separated with commas. Default: : not used. Example : * Specify "Maintenance" in the provided skills of a resource designed to perform maintenance visits type. * Specify "Small vehicle" to the provided skills of a resource able to perform downtown visits.	string
speedAdjustment *optional*	A factor to increase or decrease the vehicle speed. Whenever users observe a significant gap between estimated speeds and the real ones, they can adjust them by using this factor. It is expressed as a percentage of the used speed. Type : float. Default: : not used. Example : * speedAdjustment = 110: resource speeds will be 10% higher than the used speeds. * speedAdjustment = 80: resource speeds will be 20% lower than the used speeds.	number
startTravelTimeModifier *optional*	travel time modifier associated with the start location of the resource	xml_ns0_TSTravelTimeModifier
startX *optional*	longitude of resource start (and arrival if no endX provided)	number
startY	latitude of resource start (and arrival if no endY provided)	number

Name	Description	Schema
optional		
stopTravelTimeModifier *optional*	travel time modifier associated with the stop location of the resource	xml_ns0_TSTravelTimeModifier
tomTomWebFleetEnabled *optional*		boolean
tomTomWebFleetIdentifier *optional*		string
travelPenalty *optional*	The cost for a resource of driving for one distance unit. Use this constraint to specify the average resource taxes (gazoline, wear,???) when driving one distance unit. Type : float Default : 1.5 Example : if travelPenalty = 0.5 (euro per distance unit) and the driven distance is about 100 unit (km or miles), the total distance cost is 0,5 * 100 = 50 euros.	number
travelTimeModifier *optional*	travel time modifiers When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location travelTimeModifierValue), the portion of the travel time on which the modifier applies (travelTimeModifierLength) an offset to add to any travel duration leaving or reaching the location (travelTimeModifierOffSet). Example : * Set travelTimeModifierValue to 1.5, travelTimeModifierLength to 300 and travelTimeModifierOffSet to 60 for Resource 1 * Set travelTimeModifierValue to 2, travelTimeModifierLength to 420 and travelTimeModifierOffSet to 0 for Order 1 If the initial travel duration between Resource 1 and Order 1 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660	xml_ns0_TSTravelTimeModifier
unloadingDurationPerUnit *optional*	the time needed to the resource to deliver/pick-up one unit of product at the customer’s place. This resource’s duration is added to the fixed resource’s unloading duration and to the order’s unloading duration depending on quantity to deliver/pick-up. See also tsResource fixedUnloadingDuration and tsOrder unloadingDurationPerUnit. Example : * Set tsOrderQuantity to 3. * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 6 mn * Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration to 10 mn and tsResource unloadingPerUnit to 0 or empty: the variable part of the order is 10 mn	string

Name	Description	Schema
	* Set tsOrder unloadingDurationPerUnit to 2 mn and tsResource fixedUnloadingDuration empty and tsResource unloadingPerUnit to 1 mn: the variable part of the order is 3 mn Type : "hh:mm:ss", DateTime.	
useAllCapacities *optional*	Determines if both global capacity and capacities per product should be considered in the optimization or only global capacity. Type : Boolean Default : False Example : Set globalCapacity = 15 and useAllCapacities = False. The global capacity only is considered: the vehicle can carry 15 quantities of products, whatever product is to be carried.	boolean
useInPlanningPenalty *optional*	Penalty value if the resource is used in the planning. Use this constraint to specify the penalty if the resource is used at least one day in the planning Type : float Default : 0	number
usePenalty *optional*	A cost paid for using the resource in the computed planning. Use this constraint to ponderate the cost of using a resource element of the Resources collection, when working with rented vehicle for instance. You can use it as well to reduce the number of resources used to perform the deliveries. Type : float. Default: : not used. Example : Add the cost of use to the distance and hourly costs when working with a service provider. The solver, aiming at cost reduction, will try to eliminate this resource first, as it is the most expensive.	number
vehicleCode *optional*	Vehicles regulations. Possible values are : * bicycle * bus * car * deliveryIntermediateVehicle * deliveryLightCommercialVehicle * emergencyTruck * emergencyVehicle * intermediateVehicle * lightCommercialVehicle * pedestrian * taxi * truck If not specified, vehicle type defined in UI will be used	string

Name	Description	Schema
	Only new solver can handle optimization with several types of vehicle. If you use standard solver, you should specify the vehicle code only in the options object.	
weeklyWorkTime *optional*	The maximum resource work duration over a week. Use this constraint to specify the maximum weekly work duration. Weeks are defined as follows: * year's week, from monday to sunday, if a date has been specified in the workDays constraint * days 1 to 7 are the days of the first week and the second week start on day 8 if no date has been specified in the workDays constraint. No over work with regard to the weekly work duration is allowed. Type : "hh:mm:ss", DateTime. Default: : "168:00:00".	string
workEndTime *optional*	The end time of a resource work time window. Use this constraint to specify the time at which the tour must be over or the daily work duration. * If the optimumStartTime parameter is set to True and the dailyWorkTime or weeklyWorkTime constraint value is set by user, the tour cannot finish later than the value of the workEndTime parameter, whatever overtime may be allowed by the overtime1_Duration and overtime2_Duration constraints. * If the optimumStartTime parameter is set to True and the dailyWorkTime constraint value is not set, the end time of the tour can be adjusted to match the begining of the work time window. In this case, this parameter is used only to compute the resource daily work time and the tour can finish at midnight. * If the optimumStartTime parameter is set to False, then this parameter is used to compute the resource daily work time and the tour can finish no later than the specified end time plus the values of the first overtime duration and the second overtime duration constraints. If the daily work duration is specified by the workEndTime constraint's value, then it is equal to workEndTime minus workStartTime and lunchDuration. Example : For a vehicle to work for 6 hours between 7PM and 7AM, set * optimumStartTime=True * dailyWorkTime="06:00:00" * workStartTime="07:00:00" * workEndTime="19:00:00" For a vehicle to work for 8 hour between 8PM and 5AM with a one-hour break at 12:30, set * workStartTime="08:00:00" * workEndTime="17:00:00" * lunchDuration="01:00:00" * lunchTime="12:30:00"	string

Name	Description	Schema
	Type : "hh:mm:ss", DateTime. Default: : "24:00:00".	
workPenalty *optional*	The cost of a resource working for an hour. Use this constraint to specify the resource wages when working for an hour. Type : float. Default: : 9. Example : If workPenalty = 10 (euros per hour) and daily work time is about 8 hours, the total daily work cost is 10 * 8 = 80 euros.	number
workStartTime *optional*	The start time of the resource work time window. Use this constraint to specify the time at which the tour must start. This time can be adjusted using the optimumStartTime parameter. Type : "hh:mm:ss", DateTime. Default: : "00:00:00".	string
workingDays *optional*	The resource's work days. Use this constraint to specify the days a resource works in the planning period. This constraint is linked with the orders possible visit days constraints of Orders object elements and with the depot days of opening constraints of Depots object elements. Thus, customers and resources must have matching days for deliveries to be possible, and depots must be opened on resources working days to be used. A maximum of 64 days can be defined as work days. Working days can be written as integers (1,2???) or dates (14/05/2010, 15/05/2010, ???). If you are using dates, the oldest date in the workDays constraint defines day 1. Type : string values containing days separated with commas (like "1, 2, 5" or "14/05/2010, 15/05/2010, 18/05/2010" to specify day 1, day 2 and day 5) or intervals (like "1-10", "2=>5" or "14/05/2010=>24/05/2010") where 1 (or 14/05/2010) is the first day of the planning period. For day intervals, prefer the "=>" separator. Be careful, if the separator of date is - then "1-5" corresponds to May 1st of the current year. If you mix integer and date formats, beware that day 1 will all the same be defined by the oldest available date. Default: : "1". Example : You can define a single working day: Specify "1", "3" or "10" for the resource to work on day 1, day 3 or day 10 of the planning period. You can also define working periods. Specify "1-5" or "1=>5" or "14/05/2010=>18/05/2010" for the resource to work on a 5-days period (from day 1 to day 5 included). You can also mix single day and periods. Specify "1, 3=>5" for the resource to work on day 1 and from day 3 to 5 of the planning period.	string

(XML) TSSimulation

Polymorphism : Composition

Name	Description	Schema
depots *optional*		xml_ns0_TSDepot
nbCapacities *optional*		number

Name	Description	Schema
nbExtraTravelPenalties *optional*		number
nbQuantities *optional*		number
nbTimeWindows *optional*		number
options *optional*		xml_ns0_TSOptions
orders *optional*		xml_ns0_TSOrder
resources *optional*		xml_ns0_TSResource

(XML) TSTimeWindow

Time window

Polymorphism : Composition

Name	Description	Schema
beginTime *optional*	the minimum begin time Type : Time ("hh:mm" or "hh:mm:ss")	string
endTime *optional*	the maximum end time Type : Time ("hh:mm" or "hh:mm:ss")	string

(XML) TSTour

Details of a stop within a resource tour.

Polymorphism : Composition

Name	Description	Schema
additionalCost *optional*	Additional cost Additional cost depends on additional cost configuration specified on resources and orders	number
dayId *optional*	day of the tour	string
deliveryCost *optional*	Total delivery cost of the tour Delivery cost depends on PenaltyPerVisit defined on resources and configuration specified in options (see countDepotsInDeliveryCost and countVisitCostOnceIfSameLocation)	number
plannedOrders *optional*	Orders planned in this tour	xml_ns0_TSPlanned
reloadNb *optional*	ReloadNb Number of reloads during this tour	number
resourceCapacities	List of resource capacities	number

Name	Description	Schema
optional		
resourceId *optional*	Assigned resource identifier	string
totalCost *optional*	Total cost Total cost = (delivery cost) + (additional cost)	number
travelDistance *optional*	drive distance for this tour	number
travelDuration *optional*	drive duration for this tour	string
usedCapacities *optional*	List of used capacities Tour used capacities can exceed resource capacities if the tour contains one or several stops to a depot.	number

(XML) TSTravelPenalty

Polymorphism : Composition

Name	Description	Schema
distance *optional*	The driven distance (in the solver's measurement system unit) from which distance cost will be applied. Use this constraint when you need to specify a distance cost which can vary according to the covered distance. Up to 4 different distance costs can be specified. Each one must be related to the corresponding distance threshold, from which it will be applied	number
penalty *optional*	The cost for a resource of driving for one distance unit. Use this constraint to specify the average resource taxes (gazoline, wear,???) when driving one distance unit. Type : float Default : 1.5 Example : if penalty = 0.5 (euro per distance unit) and the driven distance is about 100 unit (km or miles), the total distance cost is 0,5 * 100 = 50 euros.	number

(XML) TSTravelTimeModifier

Polymorphism : Composition

Name	Description	Schema
length *optional*	Indicates the duration on which to apply the travel time modifier. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : 0	string
offset *optional*	Indicates the offset of the travel time modifier. Type : Time ("hh:mm" or "hh:mm:ss") or Integer (number of minutes). Default : 0	string
value	Indicates the value of the travel time modifier.	number

Name	Description	Schema
optional	When reaching a location situated in a large city, one may want to take into account driving difficulties, such as narrow streets and congestion. The travel time modifier enables to increase the travel times around a location. It is describes by three values. The value by which multiply the travel times around the location (tsDepotTravelTimeModifierValue), the portion of the travel time on which the modifier applies (length) an offset to add to any travel duration leaving or reaching the location (offSet). Example : * Set tsResource TravelTimeModifier Value to 1.5, tsResource TravelTime Modifier Length to 300 and tsResource TravelTimeModifier OffSet to 60 for Resource 1 * Set tsDepot TravelTimeModifier Value to 2, tsDepot TravelTimeModifier Length to 420 and tsDepot TravelTimeModifier OffSet to 0 for Depot 1 If the initial travel duration between Resource 1 and Depot 1 was 1000, one obtains a travel time 360 * 1.5 + 60 + 280 + 420 * 2 + 0 = 1660 Type : float Default : 1	

(XML) TSUnplanned

Unplanned order

Polymorphism : Composition

Name	Description	Schema
reason *optional*	the reason why it what not planned	string
stopID *optional*	The id of the stop.	string

(XML) TSWarning

Warning message (and associated error code) about possible Orders and Resources elements constraints misconfiguration.

The Solver object checks automatically data configuration (if not done yet) before the optimization begins. Whenever values are detected as incoherent or invalid, warnings are emitted in the Warnings property of the involved element. Depending the warning, constraints value can be replaced by their default value or not.

Polymorphism : Composition

Name	Description	Schema
constraint *optional*	id of constraint causing the warning	number
constraintName *optional*		string
i18nMessageCode *optional*		string

Name	Description	Schema
id *optional*	if of object causing the warning	string
message *optional*	warning message	string
messageId *optional*		number
objectType *optional*	type of object causing the warning	string
value *optional*		string

(XML) addVisitsRequest

Polymorphism : Composition

Name	Description	Schema
id *optional*		string
language *optional*		string
orders *optional*		xml_ns0_TSOrder

(XML) addVisitsResult

Polymorphism : Composition

Name	Description	Schema
message *optional*		string
status *optional*		string

(XML) balanceType

Type : enum (NONE, ORDERS, HOURS, QUANTITY)

(XML) costOperator

Type : enum (SUM, MAX, MIN, AVERAGE)

(XML) depotCostMode

Type : enum (NONE, ALL, FIRST, ALLBUTFIRST)

(XML) depotsResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
depots *optional*	List of depots	xml_ns0_TSDepot
message *optional*	error message	string
status *optional*	response status, OK or ERROR	xml_ns0_status

(XML) distanceType

Type : enum (KILOMETERS, MILES, METERS, FEET)

(XML) fulfillmentResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
lastKnownPosition *optional*	List of positions	xml_ns0_operationalLastKnownPosition
message *optional*	error message	string
operationalOrderAchievements *optional*	List of orders	xml_ns0_operationalOrderAchievement
status *optional*	response status, OK or ERROR	xml_ns0_status

(XML) geocodeInfos

Polymorphism : Composition

Name	Description	Schema
address *optional*		string
addressComplement *optional*		string
city *optional*		string
country *optional*		string
geocodeAddressLine *optional*		string
geocodeCity *optional*		string
geocodePostalCode *optional*		string
geocodeType *optional*		number

Name	Description	Schema
postcode *optional*		string
region *optional*		string
score *optional*		number

(XML) loginTokenResult

Result of the optimize service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message	string
status *optional*	response status, OK or ERROR	xml_ns0_status
token *optional*	the token string	string
validUntil *optional*	The token validity end date	string

(XML) operationalExportRequest

Operational planning export parameters

Polymorphism : Composition

Name	Description	Schema
dayNums *optional*		number
force *optional*	if true, any existing operational planning will be replaced if false and if optimization period overlaps any existing operational planning, export will fail.	boolean
resourceMapping *optional*	List of MobileResourceMapping defining relation between resource identifier in optimize request and real mobile resource identifier	xml_ns0_operationalResourceMapping
startDate *optional*	real date corresponding to day 1 of optimize request data	string
taskId *optional*	Task identifier. Must point to a completed optimization.	string

(XML) operationalLastKnownPosition

Polymorphism : Composition

Name	Description	Schema
accuracy *optional*	GPS positioning accuracy (radius in meters)	number
batteryLevel	Battery level of the device	number

Name	Description	Schema
optional		
date *optional*	last position recording date	string
gpsStatus *optional*	GPS status of the device	xml_ns1_GPSStatus
id *optional*		string
lat *optional*	Latitude	number
lon *optional*	Longitude	number
privateLife *optional*	Private life status in Mobile App. If true, it means that mobile App is currently in Private life mode, therefore this position is the last known position before the Private life switch.	boolean

(XML) operationalOrderAchievement

Polymorphism : Composition

Name	Description	Schema
achievementComment *optional*	Achievement comment	string
achievementEnd *optional*	Achievement end date and time	string
achievementEndLat *optional*	Achievement end latitude	number
achievementEndLon *optional*	Achievement end longitude	number
achievementStart *optional*	Achievement start date and time	string
achievementStartLat *optional*	Achievement start latitude	number
achievementStartLon *optional*	Achievement start longitude	number
apprauchSmsId *optional*		string
apprauchSmsStatus *optional*		string
data *optional*	fulfillment form data	object
date *optional*	Planning day	string
end *optional*	Planned end date and time	string
feedbackSmsId *optional*		string

Name	Description	Schema
feedbackSmsStatus *optional*		string
geocode *optional*		xml_ns0_geocodeInfos
id *optional*		string
lastSynchroStatusChange *optional*	Last change from mobile app	string
lat *optional*	Latitude	number
lon *optional*	Longitude	number
operationalResourceId *optional*	Mobile resource identifier (mobile app login)	string
order *optional*	Original order	xml_ns0_TSOrder
pictures *optional*	List of picture relative urls. Url root for pictures is https://geoservices.geoconcept.com/ToursolverCloud/api/rest/otmobile/pictures/	string
plannedOrder *optional*	Planned order	xml_ns0_TSPlanned
signaturePicture *optional*	Signature, as a picture relative URL (a flavor of the reference *signatureSvg*) Is bound and synced from the *signaturesSvg* field. Is a relative URL, as also done for the *pictures* field (see its documentation for details).	string
signatureSvg *optional*	Signature svg	string
simulationDayId *optional*	day containing this order in the simulation used to fill the fulfillment planning	string
simulationId *optional*	identifier of the simulation used to fill the fulfillment planning	string
start *optional*	Planned start date and time	string
status *optional*	fulfillment status	xml_ns0_operationalOrderStatus
synchroStatus *optional*	Sync status	xml_ns0_operationalOrderSynchroStatus
timeWindowEnd *optional*		string
timeWindowSmsId *optional*		string
timeWindowSmsStatus *optional*		string
timeWindowStart *optional*		string
type *optional*	Event type	xml_ns0_operationalOrderType

(XML) operationalOrderStatus

Type : enum (CANDIDATE, FIXED, ACCEPTED, REFUSED, STARTED, FINISHED, CANCELLED, PAUSED, RESUMED, UNKNOWN)

(XML) operationalOrderSynchroStatus

Type : enum (PUBLISHED, SENT, UPDATED, UNKNOWN)

(XML) operationalOrderType

Type : enum (MISSION, RESTBREAK, LUNCHBREAK, WAITBREAK, RELOADBREAK, START, END, ENDBEFORENIGHT, STARTAFTERNIGHT, BRIEFING, DEBRIEFING, UNKNOWN)

(XML) operationalResourceMapping

The mobile resource mapping links the resource identifier used in optimized data and the mobile resource identifier used for operational planning export

Polymorphism : Composition

Name	Description	Schema
id *optional*	resource identifier	string
operationalId *optional*	Mobile identifier	string

(XML) optimResultResult

Result of an optimization task.

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message	string
plannedOrders *optional*	the planned stops	xml_ns0_TSPlanned
simulationId *optional*	Id of the simulation associated to this task	string
status *optional*	response status, OK or ERROR	xml_ns0_status
taskId *optional*	the id of the optimization task (usefull if result is automatically sent through a webhook).	string
unplannedOrders *optional*	the orders which has not been planned because: it was sent by an other mail service it was not scheduled by any resource.	xml_ns0_TSUnplanned
warnings *optional*	the list of warning messages (and associated error codes) about possible Orders and Resources elements constraints misconfiguration.	xml_ns0_TSWarning

(XML) optimStatusResult

Result of status request

Polymorphism : Composition

Name	Description	Schema
currentCo2 *optional*	Remaining Kg CO2 of the current solution	number
currentCost *optional*	Cost of the current solution	number
currentCourierCost *optional*	Courier cost of the current solution	number
currentDeliveredQuantity *optional*	Total quantity delivered of the current solution	number
currentDeliveryCost *optional*	Delivery cost of the current solution	number
currentDriveCost *optional*	Drive cost of the current solution	number
currentDriveDistance *optional*	Drive distance of the current solution	number
currentDriveTime *optional*	Drive time in seconds of the current solution	number
currentFixedCost *optional*	Fixed cost of the current solution	number
currentLateTime *optional*	Late time in seconds of the current solution	number
currentNightsCost *optional*	Nights cost of the current solution	number
currentOpenTourNumber *optional*	initial number of open tours (tours with at least one visit)	number
currentOverWorkCost *optional*	Overwork cost of the current solution	number
currentOverWorkTime *optional*	Over work time in seconds of the current solution	number
currentPickUpQuantity *optional*	Total quantity picked-up of the current solution	number
currentPlannedVisits *optional*	Number of planned visits of the current solution (new engine only)	number
currentRestTime *optional*	Rest time in seconds of the current solution	number
currentUnplannedVisits *optional*	Number of visits unplanned or delivered by a courier of the current solution	number
currentVisitsNb *optional*		number
currentWaitTime *optional*	Wait time in seconds of the current solution	number

Name	Description	Schema
currentWorkCost *optional*	Work cost of the current solution	number
currentWorkTime *optional*	Work time in seconds of the current solution	number
initialCo2 *optional*	Remaining Kg CO2 of the initial solution	number
initialCost *optional*	Cost of the initial solution	number
initialCourierCost *optional*	Courier cost of the initial solution	number
initialDeliveredQuantity *optional*	Total quantity delivered of the initial solution	number
initialDeliveryCost *optional*	Delivery cost of the initial solution	number
initialDriveCost *optional*	Drive cost of the initial solution	number
initialDriveDistance *optional*	Drive distance of the initial solution	number
initialDriveTime *optional*	Drive time in seconds of the initial solution	number
initialFixedCost *optional*	Fixed cost of the initial solution	number
initialLateTime *optional*	Late time in seconds of the initial solution	number
initialNightsCost *optional*	Nights cost of the initial solution	number
initialOpenTourNumber *optional*	initial number of open tours (tours with at least one visit)	number
initialOverWorkCost *optional*	Overwork cost of the initial solution	number
initialOverWorkTime *optional*	Over work time in seconds of the initial solution	number
initialPickUpQuantity *optional*	Total quantity picked-up of the initial solution	number
initialPlannedVisits *optional*	Number of planned visits of the initial solution (new engine only)	number
initialRestTime *optional*	Rest time in seconds of the initial solution	number
initialUnplannedVisits *optional*	Number of visits unplanned or delivered by a courier of the initial solution	number
initialWaitTime *optional*	Wait time in seconds of the initial solution	number
initialWorkCost *optional*	Work cost of the initial solution	number
initialWorkTime	Work time in seconds of the current solution	number

Name	Description	Schema
optional		
message *optional*	error message	string
mileageChartRemainingTime *optional*	Mileage and travel time matrix computing remaining time. This information may not be available depending on the geographical area.	number
optimizeStatus *optional*	Current status of the optimization process	xml_ns0_optimizeStatus
simulationId *optional*	Id of the simulation associated to this task	string
startTime *optional*	Start time of the optimization	string
status *optional*	response status, OK or ERROR	xml_ns0_status
subOptimAbortedNb *optional*	Number of sub-optimizations in aborted state	number
subOptimErrorNb *optional*	Number of sub-optimizations in error state	number
subOptimFinishedNb *optional*	Number of sub-optimizations in finished state	number
subOptimNb *optional*	Number of sub-optimizations created (after pre-sectorization)	number
subOptimRunningNb *optional*	Number of sub-optimizations in running state	number
subOptimWaitingNb *optional*	Number of sub-optimizations in waiting state	number

(XML) optimStopResult

Result of status request

Polymorphism : Composition

Name	Description	Schema
firstStopAsked *optional*	First stop demand stamp	string
message *optional*	error message	string
status *optional*	response status, OK or ERROR	xml_ns0_status

(XML) optimToursResult

Result of an optimization task.

Polymorphism : Composition

Name	Description	Schema
message	error message	string

Name	Description	Schema
optional		
status *optional*	response status, OK or ERROR	xml_ns0_status
taskId *optional*	the id of the optimization task (usefull if result is automatically sent through a webhook).	string
tours *optional*	List of tours (one tour per resource per day)	xml_ns0_TSTour
unplannedOrders *optional*	the orders which has not been planned because: it was sent by an other mail service it was not scheduled by any resource.	xml_ns0_TSUnplanned
warnings *optional*	the list of warning messages (and associated error codes) about possible Orders and Resources elements constraints misconfiguration.	xml_ns0_TSWarning

(XML) optimizeRequest

Optimization request

Polymorphism : Composition

Name	Description	Schema
beginDate *optional*	This date will be used if you try to export the optimization result through TsCloud GUI. Format : YYYY-MM-DD Default : day+1.	string
countryCode *optional*	Main country code used to route the optimization to the good optimization server farm.	string
depots *optional*	list of depots (for reloading or starting tours)	xml_ns0_TSDepot
language *optional*	Language to use for message localization	string
options *optional*	the optimize task options	xml_ns0_TSOptions
orders *optional*	list of orders (visits to do)	xml_ns0_TSOrder
organization *optional*	In multi-user content, you generally create various organizations that allows to define who sees what. If you set the organization here, only users that can see this organization will see the result of this optimization in the UI. Note that if you specified a teamId instead of sending the resources in the optimization data, the organization will be guessed from the team. If no organization is set, all users will see the result.	string
resources *optional*	collection of resources, the elements which will perform deliveries, pick-ups, commercial visit, etc.	xml_ns0_TSResource
simulationName *optional*	Simulation name Optional : generated automatically if not provided	string

Name	Description	Schema
userLogin *optional*	In multi-user content, you can specify a user login you. The simulation will be owned by this user. By default, the owner of the simulation is the default user of the account.	string

(XML) optimizeResult

Result of the optimize service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message	string
status *optional*	response status, OK or ERROR	xml_ns0_status
taskId *optional*	the id of the optimization task that was launched	string

(XML) optimizeStatus

Type : enum (undefined, waiting, geocoding, mileageChartBuilding, running, aborted, terminated, error, sectorizationWaiting, sectorizationRunning, sectorizationFinished, sectorizationAborted)

(XML) persistentObject

Polymorphism : Composition

Name	Description	Schema
id *optional*		string

(XML) phoneNumberType

Type : enum (MOBILE, OFFICE, HOME, OFFICE_FAX, HOME_FAX)

(XML) resourcesResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message	string
resources *optional*	List of resources	xml_ns0_TSResource
status *optional*	response status, OK or ERROR	xml_ns0_status

(XML) routingMethod

Type : enum (TIME, DISTANCE)

(XML) sectorizationMethod

Type : enum (TIME, DISTANCE)

(XML) simulationResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message	string
simulation *optional*		xml_ns0_TSSimulation
status *optional*	response status, OK or ERROR	xml_ns0_status

(XML) status

Type : enum (OK, ERROR)

(XML) toursolverServiceResult

generic result of service

Polymorphism : Composition

Name	Description	Schema
message *optional*	error message	string
status *optional*	response status, OK or ERROR	xml_ns0_status

(XML) webhookExportMode

Type : enum (NONE, ORDERS, TOURS)

xml_ns1_GPSStatus

<p>Classe Java pour GPSStatus.

<p>Le fragment de sch??ma suivant indique le contenu attendu figurant dans cette classe. <p> <pre> <simpleType name="GPSStatus"> <enumeration value="0"/> <enumeration value="1"/> <enumeration value="2"/> </restriction> </simpleType> </pre>

Type : enum (0, 1, 2)