TMS IntraWeb WebOSMaps

DEVELOPERS GUIDE

April 2017

Table of contents

Introduction

The TMS TTIWWebOSMaps is a component that allows integration of the OpenStreetMaps road map control in IntraWeb applications. TTIWWebOSMaps offers pan, zoom and scale control.

In this document you will find an overview of the TTIWWebOSMaps component and its features, code snippets to quickly start using the component and overviews of properties, methods and events.

Availability

TTIWWebOSMaps component is available as IntraWeb component for Delphi and C++Builder.

TTIWWebOSMaps can be used for Win32 and Win64 application development * and is available for
Delphi XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle,10.1 Berlin,10.2 Tokyo
C++Builder XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle,10.1 Berlin,10.2 Tokyo

TTIWWebOSMaps is compatible with the IntraWeb 11, 12, 14.

TTIWWebOSMaps has been designed for and tested with: Windows Vista, Windows 2008, Windows 7, Windows 8 and Windows 10.

*Win64 development requires RAD Studio XE2 or newer versions.

Terms of use

With the purchase of TTIWWebOSMaps, you are entitled to our consulting and support services to integrate the OpenStreetMaps service in IntraWeb applications and with this consulting and support comes the full source code needed to do this integration. As TTIWWebOSMaps uses the OpenStreetMaps and OpenLayers services, you're bound to the terms of these services that can be found at:

http://www.openstreetmap.org/copyright

TMS software is not responsible for the use of TTIWWebOSMaps. The purchase of TTIWWebOSMaps does not include any license fee that you might possibly be required to pay to OpenStreetMaps/OpenLayers. It will depend on your type of usage of the OpenStreetMaps/OpenLayers services whether a license fee needs to be paid to OpenStreetMaps/OpenLayers.
It is the sole responsibility of the user or company providing the application that integrates the OpenStreetMaps/OpenLayers services to respect the OpenStreetMaps/OpenLayers terms and conditions. TMS software does not take any responsibility nor indemnifies any party violating the OpenStreetMaps/OpenLayers services terms & conditions.

Limited warranty

TMS software cannot guarantee the current or future operation & uptime of the OpenStreetMaps/OpenLayers services. TMS software offers the consulting and support for TTIWWebOSMaps in good faith that the OpenStreetMaps/OpenLayers services are reliable and future-proof services. In no case, TMS software shall offer refunds or any other compensation in case the OpenStreetMaps/OpenLayers services terms/operation change or stop.

List of included components

TTIWWebOSMaps is the core map component.

Online references

TMS software website:
http://www.tmssoftware.com

TMS TTIWWebOSMaps page:
http://www.tmssoftware.com/site/IWWebOSMaps.asp

TTIWWebOSMaps

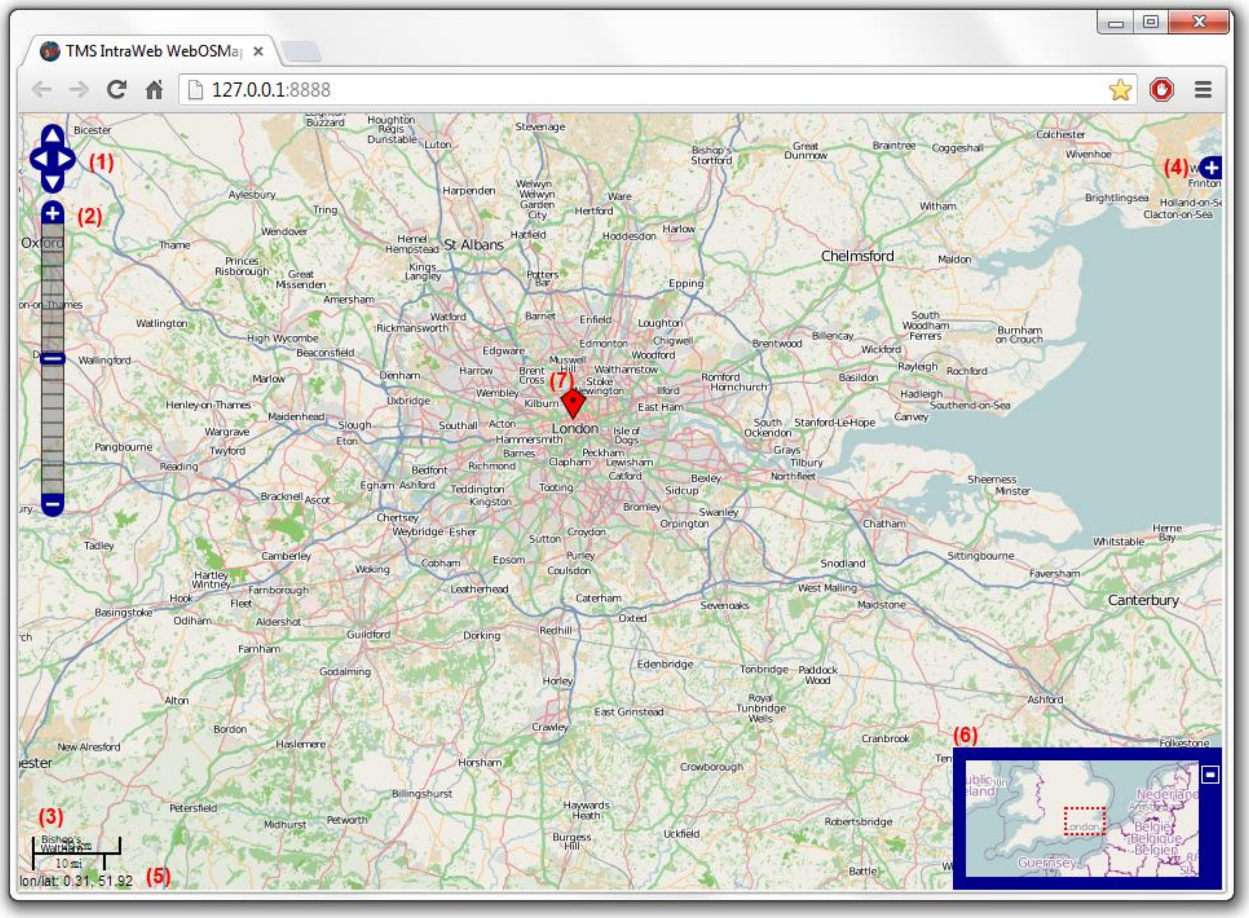
TTIWWebOSMaps description

The TMS TTIWWebOSMaps is a mapping component to integrate, display & control OpenStreetMaps in an IntraWeb application. It supports the default roadmap view. The TTIWWebOSMaps component offers pan, zoom and scale control. An overview-map is integrated for faster panning.
Markers can be added to the map via the longitude/latitude coordinates.
Polylines can be added to the map via a Path, which is a collection of longitude/latitude coordinates.
Polygons can be added to the map via longitude/latitude coordinates. Various polygon types exist: custom polygon, circle or rectangle.
Markers, Polylines and polygons can also be displayed with a custom label text.

TTIWWebOSMaps features

- Position markers can be added to the maps. Markers can be default balloons or custom images.
- Markers is a collection of positions that are indicated on the map. Markers are based on longitude and latitude coordinates.
- A custom label text can optionally be displayed on top of a Marker, polyline or polygon or anywhere on the map.
- Polylines is a collection of lines that are displayed on the map. Polylines are based on a list of longitude and latitude coordinates.
- Polygons is a collection of closed lines with a filled region that are displayed on the map. Polygons are based on a list of longitude and latitude coordinates (for Polygons of type ptPath), a center point and radius (for Polygons of type ptCircle) or two longitude and latitude coordinates (for Polygons of type ptRectangle).
- Different controls are available and can be turned on or off. LayerSwitcher, OverViewMap control, PanZoom control, Scale control and MousePosition. The position on the screen of the control as well as the visibility can be defined.
- Different mouse and keyboard options are available: dragging of the map, enabling/disabling all controls, enabling/disabling zoom on double clicking the mouse, enabling/disabling the mouse scroll wheel and enabling/disabling the keyboard.

TTIWWebOSMaps architecture



The core part of the TMS IntraWeb WebOSMaps is the TTIWWebOSMaps control, an IntraWeb component exposing properties, methods and events to control OpenStreetMaps using the OpenLayers service. Additional OpenLayers controls can be optionally enabled on the map, i.e. a PanControl (1), a ZoomControl (2), a ScaleControl (3), a LayerSwitcherControl (4), a MousePosition (5) and an OverviewmapControl (6).

Different markers (7) can be added to display preferred locations. The marker can display a default balloon or when a valid URL is provided, an image or icon is displayed.

Various events are triggered when the user interacts with keyboard or mouse with the map.

TTIWWebOSMaps use

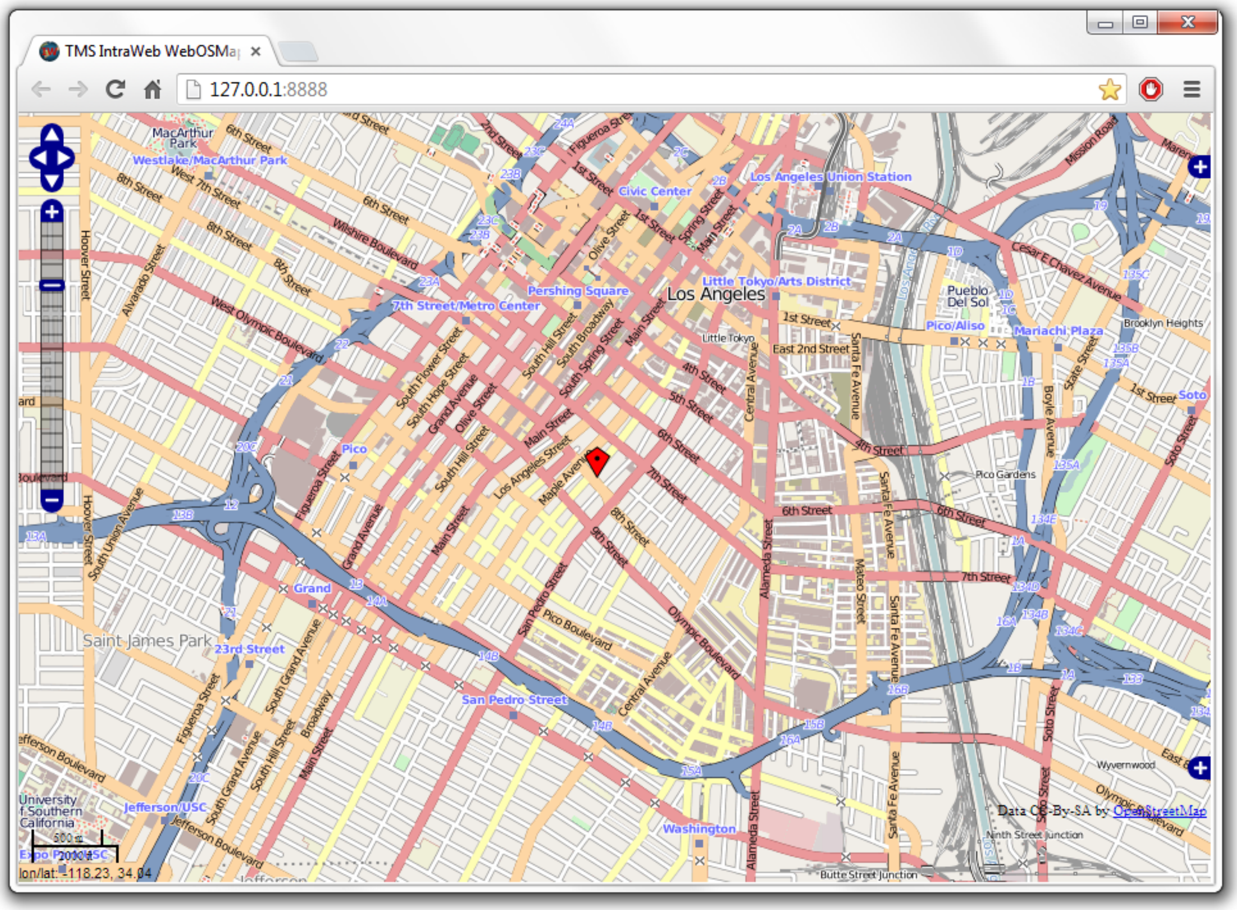
Getting started

From the component palette, select TTIWWebOSMaps and drop it on a form. The default center location displayed when an IntraWeb application is launched is set by: TIWWebOSMaps.MapOptions.DefaultLongitude, TIWWebOSMaps.MapOptions.DefaultLatitude.

Markers can be added to the map by adding a new entry to the collection TIWWebOSMaps.Markers and setting the Marker's properties Longitude & Latitude.

This code snippet sets up the default view of the TTIWWebOSMaps to show the Los Angeles Theatre on Broadway at zoom level:

```
begin

// center the map at the coordinate
TIWWebOSMaps1.MapOptions.DefaultLatitude := 34.04;
TIWWebOSMaps1.MapOptions.DefaultLongitude := -118.25;

// Add a marker for the Los Angeles theatre
TIWWebOSMaps1.Markers.Add(TIWWebOSMaps1.MapOptions.DefaultLatitude,
TIWWebOSMaps1.MapOptions.DefaultLongitude, 'New Marker', '', false,
true);

// set zoom level
TIWWebOSMaps1.MapOptions.ZoomMap := 14;

end;
```


TMS IntraWeb WebOSMa
127.0.0.1:8888
Los Angeles
Data CC-By-SA by OpenStreetMap
lon/lat: -118.23, 34.04

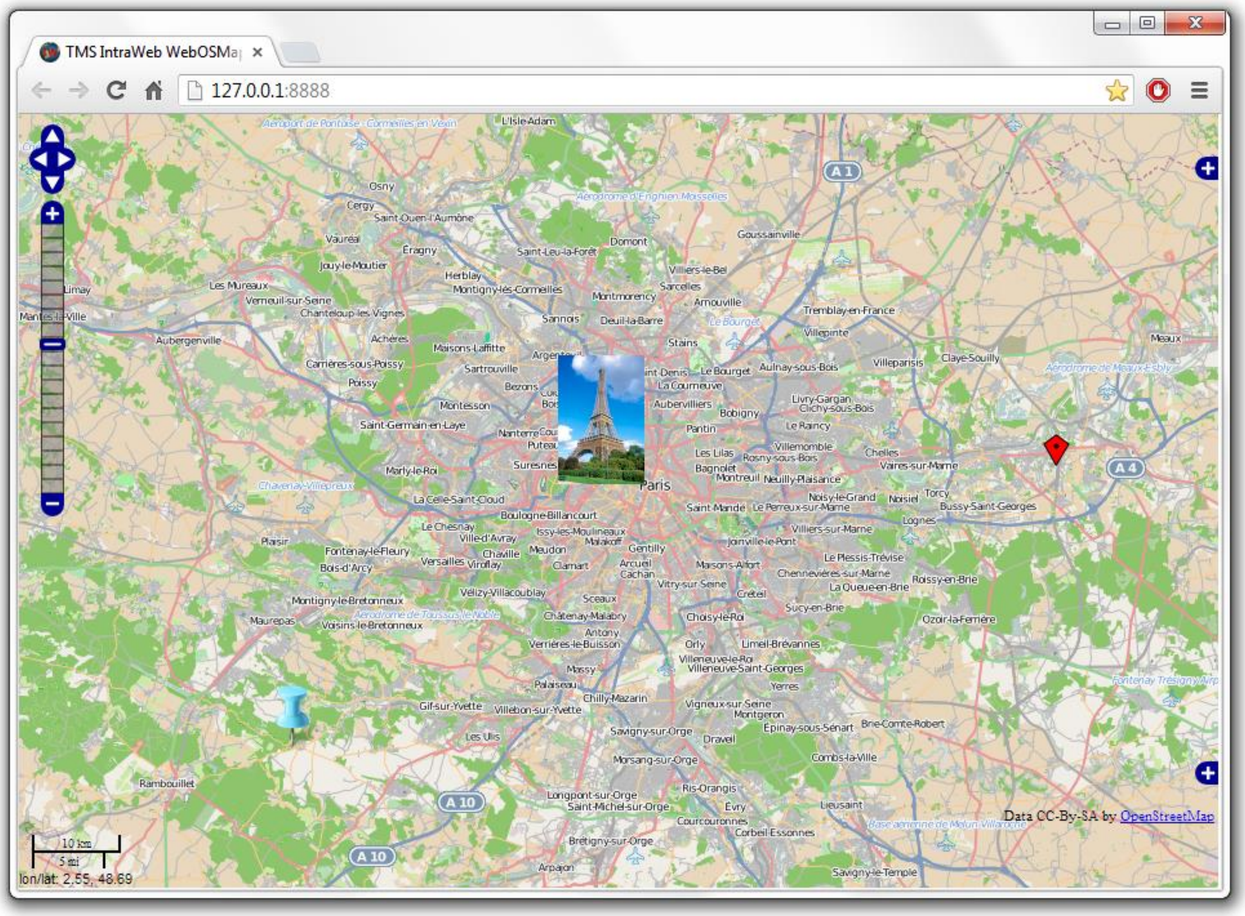
General map settings

TTIWWebOSMaps.MapOptions properties

- **DefaultLatitude:** Sets the latitude value for the default position when Launch is called.
- **DefaultLongitude:** Sets the longitude value for the default position when Launch is called.
- **DisableDoubleClickZoom:** When set to true, disables zoom functions when double-clicking.
- **Draggable:** When set to true, the entire map can be moved around in the control.
- **EnableKeyboard:** When set to true, enables the use of the keyboard for controlling panning in the map (or in street view mode).
- **ImageURL:** Defines the path to the image used for the OpenLayers controls. If no path is specified the default OpenLayers images are used.
- **LanguageURL:** Defines the path to the JavaScript file that contains the text values in a specific language. If no path is specified English is used as the default language.
- **ScriptURL:** Defines the path to the OpenLayers.js file used by the OpenLayers service. If no path is specified the default JavaScript file from the OpenLayers.org server is used.
- **Language:** Defines the language of the Copyright message, and the TTIWWebOSMaps.ControlsOptions.MapTypeControl displayed text.
- **ScrollWheel:** When set to true, enables the use of the scroll wheel. The scroll wheel can be used to zoom in and out on the map.
- **StyleURL:** Defines the path to Style.css file used by the OpenLayers service. If no path is specified the default JavaScript file from the OpenLayers.org server is used.
- **ZoomMap:** Is to be used to set the default zoom at startup. The zoom value is a value between 1 and 18 with 18 being the highest zoom level.

Map markers

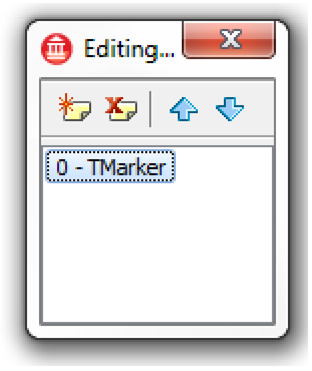
TMarkers is a collection of marker items giving the possibility to highlight certain locations on the map. A marker is either a default balloon or can be set to a custom icon by defining the URL for it. The example below shows a mix of pictures and a standard Google balloon marker. A sample on how to create a marker info window can be found in the samples paragraph.



On the left hand side, a PNG image is used as marker. In the middle section, a JPEG image is displayed as marker. On the right hand side, a standard OpenLayers balloon marker is used, as the marker was created with an empty icon property.

Adding markers

First open the markers collection editor by clicking the TTIWWebOSMaps.Markers property in the Object Inspector. From here, markers can be added or removed.



The equivalent in code is:

Adding a marker:

```
uses
  IWWebOSMapsMarkers;

var

  ADraggable, Avisible: Boolean
  ALongitude, ALatitude: double;
  ATitle: string;
  AIcon: string;

begin

  ADragable := true;
  AVisible := true;
  ALongitude := 18.45464;
  ALatitude := -23.7871;
  ATitle := ‘Some marker’;
  AIcon := http://...

  TIWWebOSMaps1.Markers.Add(aLatitude, aLongitude, aTitle, anIcon,
  aDraggable, aVisible);

end;
```

TTIWWebOSMaps.Markers properties

- **Draggable:** When set to true, the marker can be moved around the map when dragged.
- **Icon:** Allows the use of an image as marker. This can also be a picture when the url to that image is defined. An example can be found in the samples paragraph.
- **Latitude:** Sets the latitude value of the marker on the map.
- **Longitude:** Sets the longitude value of the marker on the map.
- **Tag:** Sets the tag for the marker.
- **Title:** Sets the title for the marker.
- **Visible:** When set to true, the marker is shown on the map.

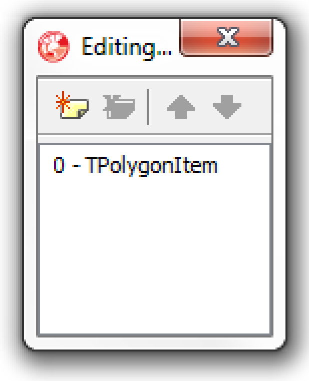
Map polygons

TIWWebOSMaps.Polygons is a collection of closed lines giving the possibility to highlight certain regions on the map. A text label can optionally be displayed on top of the polygon. The screenshot below shows a circle around Paris.

Paris
London
Dublin
France
España
Deutschland
Lisboa

<u>Adding polygons</u>

First open the polygons collection editor by clicking the TTIWWebOSMaps.Polygons property in the Object Inspector. From here, polygons can be added or removed.



The equivalent in code is:

Adding a polygon:

```
uses
  IWWebOSMapsPolygons;

var
  Circle: TMapPolygon;
  PolygonItem: TPolygonItem;

begin
  PolygonItem := TIWWebOSMaps1.Polygons.Add;
  Circle := PolygonItem.Polygon;
  Circle.PolygonType := ptCircle;
  Circle.BackgroundOpacity := 50;
  Circle.BorderWidth := 2;
  Circle.Radius := 75000;
  Circle.Center.Latitude := 48.86;
  Circle.Center.Longitude := 2.35;
  Circle.LabelText := 'Paris';
  Circle.LabelFont.Color := clRed;
  Circle.LabelFont.Size := 18;
  Circle.LabelFont.Style := [fsBold];
  Circle.LabelOffset.Y := 30;
  TIWWebOSMaps1.CreateMapPolygon(Circle);

end;
```

Editing a polygon:

```
TIWWebOSMaps1.Polygons[0].Polygon.Visible := not
TIWWebOSMaps1.Polygons[0].Polygon.Visible;

TIWWebOSMaps1.UpdateMapPolygon(TIWWebOSMaps1.Polygons[0].Polygon);
```

Removing a polygon:

```
TIWWebOSMaps1.DeleteMapPolygon(Index);
```

TTIWWebOSMaps.Polygons properties

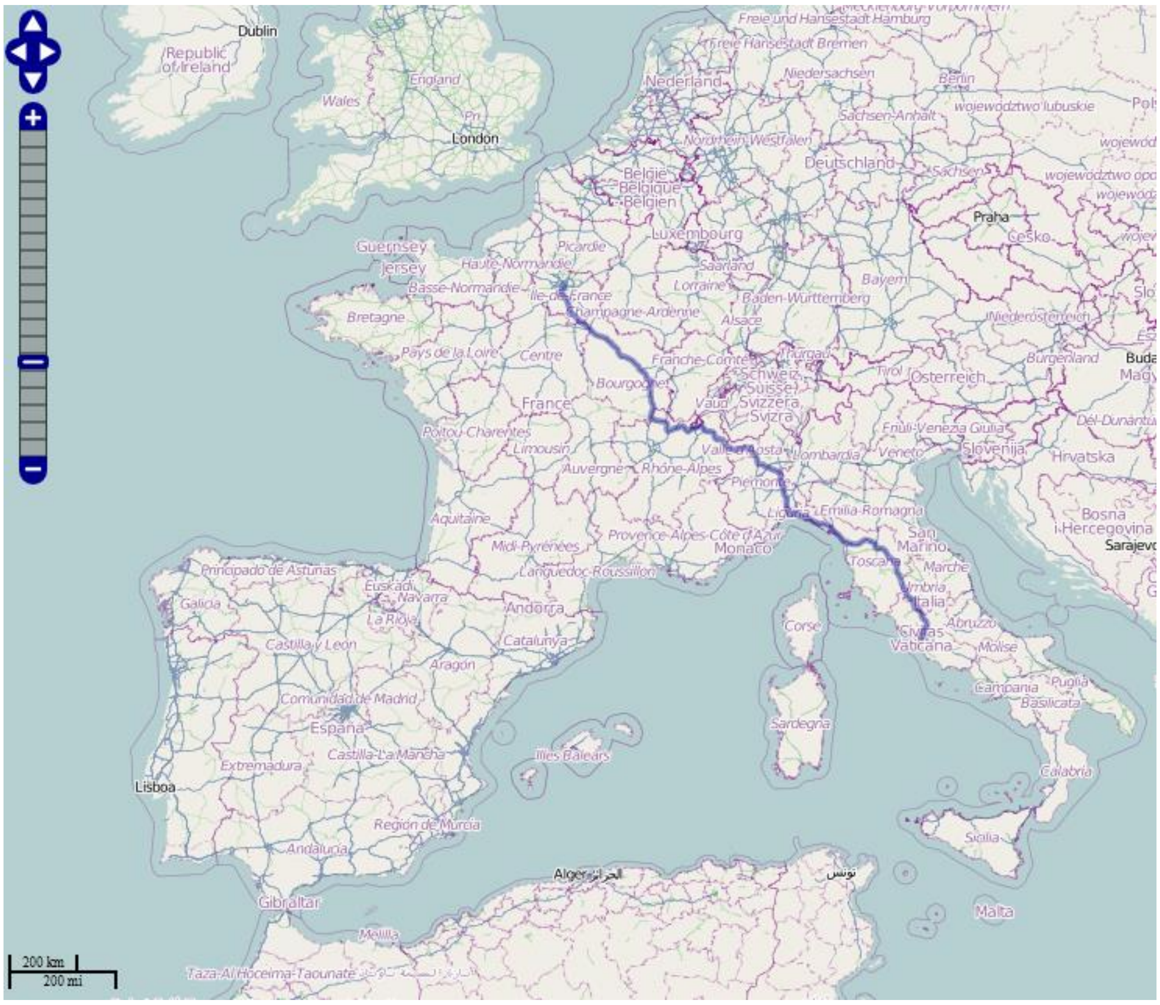
- **BackgroundColor:** The color of the polygon.
- **BackgroundOpacity:** The opacity of the polygon.
- **BorderColor:** The border color of the polygon.
- **BorderOpacity:** The border opacity of the polygon.
- **BorderWidth:** The width of the polygon border in pixels.
- **Bounds:** Sets the bounds of a polygon when PolygonType is set to ptRectangle.
    - **NorthEast:** Sets the latitude/longitude of the north east corner of the rectangle
        - **Latitude**
        - **Longitude**
    - **SouthWest:** Sets the latitude/longitude of the south west corner of the rectangle
        - **Latitude**
        - **Longitude**
- **Center:** Sets the latitude/longitude of the center point of the circle when PolygonType is set to ptCircle.
    - **Latitude**
    - **Longitude**
- **LabelFont:** The font name used for the label text. The label is displayed based on the LabelFont.Color, LabelFont.Name, LabelFont.Size and LabelFont.Style.fsBold settings.
- **LabelOffset:** The offset in pixels at which the label text is displayed.
    - **X:** The X-axis offset value in pixels.

  - **Y:** The Y-axis offset value in pixels.

- **LabelText:** The text displayed in the label. If this value is empty, no label is displayed.
- **Path:** The ordered sequence of coordinates of the polygon that forms a closed loop (when PolygonType is set to ptPath). Paths are closed automatically.
  - **Latitude:** Sets the latitude value of the polygon path item on the map.
  - **Longitude:** Sets the longitude value of the polygon path item on the map.
- **PolygonType:** Sets the type of polygon to be rendered.
  - **ptCircle:** Renders a circle based on the Radius and Center property values.
  - **ptPath:** Renders a polygon based on the list of Path coordinates.
  - **ptRectangle:** Renders a rectangle based on the Bounds property values.
- **Radius:** The radius of the polygon in meters. (When PolygonType is set to ptCircle)
- **Tag:** The tag of the polygon.
- **Visible:** When set to true, the polygon is shown on the map.

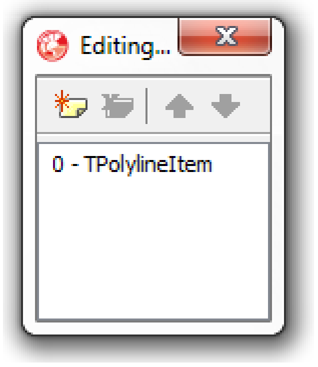
Map polylines

TTIWWebOSMaps.Polylines is a collection of lines giving the possibility to highlight certain routes on the map. A text label can optionally be displayed on top of the polygon. The screenshot below shows a route between a start location (Paris) and end location (Rome).



Adding polylines

First open the polylines collection editor by clicking the TTIWWebOSMaps.Polylines property in the Object Inspector. From here, polylines can be added or removed.



0 - TPolylineItem

The equivalent in code is:

Adding a polyline:

```
uses
  IWWebOSMapsPolylines;

var
  Poly: TPolyline;
  PolygonItem: TPolylineItem;
  pi: TPathItem;

begin

  PolylineItem := TIWWebOSMaps1.Polygons.Add;
  Poly := PolylineItem.Polyline;
  Poly := PItem.Polyline;
  Poly.Width := 4;
  Poly.Color := clWebDarkBlue;
  Poly.Opacity := 100;

  pi := Poly.Path.Add;
  pi.Latitude := 39.58108;
  pi.Longitude := -105.63535;

  pi := Poly.Path.Add;
  pi.Latitude := 40.315939;
  pi.Longitude := -105.440630;

  pi := Poly.Path.Add;
  pi.Latitude := 43.785890;
  pi.Longitude := -101.90175;

  TIWWebOSMaps1.CreateMapPolyline(Poly);
```

```
end;
```

Editing a polyline:

```
TIWWebOSMaps1.Polylines[0].Polyline.Visible := not
TIWWebOSMaps1.Polylines[0].Polyline.Visible;

TIWWebOSMaps1.UpdateMapPolyline(TIWWebOSMaps1.Polylines[0].Polyline);
```

Removing a polyline:

```
TIWWebOSMaps1.DeleteMapPolyline(Index);
```

TTIWWebOSMaps.Polylines properties

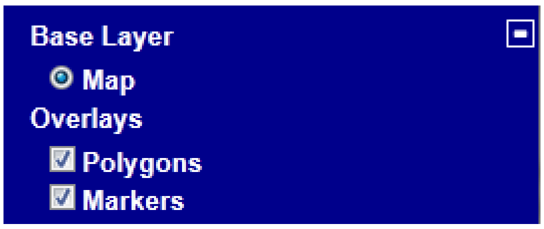
- **Color:** The color of the polyline.
- **LabelFont:** The font name used for the label text. The label is displayed based on the LabelFont.Color, LabelFont.Name, LabelFont.Size and LabelFont.Style.fsBold settings.
- **LabelOffset:** The offset in pixels at which the label text is displayed.
  - **X:** The X-axis offset value in pixels.
  - **Y:** The Y-axis offset value in pixels.
- **LabelText:** The text displayed in the label. If this value is empty, no label is displayed.
- **Opacity:** The opacity of the polyline.
- **Path:** The ordered sequence of coordinates of the polyline.
  - **Latitude:** Sets the latitude value of the polyline path item on the map.
  - **Longitude:** Sets the longitude value of the polyline path item on the map.
- **Tag:** The tag of the polygon.
- **Visible:** When set to true, the polyline is shown on the map.
- **Width:** The width of the polyline in pixels.

Maps ControlsOptions

The ControlsOptions class property bundles various settings for controlling the appearance and behaviour of various controls in the map.

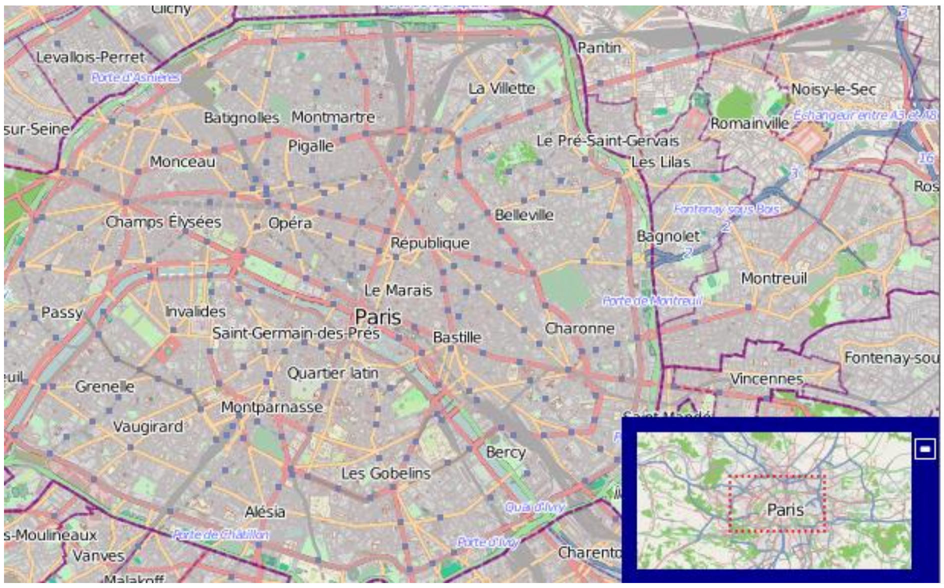
TTIWWebOSMaps.ControlsOptions properties

- **LayerSwitcherControl:** Defines the settings for layer switcher control. This control allows to enable or disable the Marker and Polygon/Polyline layer.



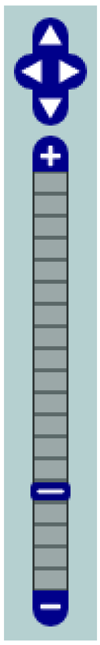
    - **Left:** The left position of the control in pixels. If the value is -1 the controls is displayed in its default position.
    - **Top:** The top position of the control in pixels. If the value is -1 the controls is displayed in its default position.
    - **Visible:** When set to true, the control is drawn on the map.

- **OverviewMapControl:** Defines the settings for the overview map control that shows a larger area, with the actual view displayed inside a red dotted line. When holding the mouse down in this area, and moving within the overviewmap control, the map can be panned to another location.



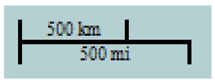
- **Left:** The left position of the control in pixels. If the value is -1 the controls is displayed in its default position.
- **Top:** The top position of the control in pixels. If the value is -1 the controls is displayed in its default position.
- **Visible:** When set to true, the control is drawn on the map.

- **PanZoomControl:**
  - Sets the pan control that allows panning the actual view of the map within the control, by clicking the arrow keys (up, down, left, right).
  - Defines the settings for the Zoom control that allows to zoom in on the actual view of the map, or to zoom out to a larger area. The center position on the screen is used as zooming location.



- **Left:** The left position of the control in pixels. If the value is -1 the controls is displayed in its default position.
- **Top:** The top position of the control in pixels. If the value is -1 the controls is displayed in its default position.
- **Visible:** When set to true, the control is drawn on the map.

- **ScaleControl:** Defines the settings for the Scale control that shows the actual scale of the view in the control.



- **Left:** The left position of the control in pixels. If the value is -1 the controls is displayed in its default position.
- **Top:** The top position of the control in pixels. If the value is -1 the controls is displayed in its default position.
- **Visible:** When set to true, the control is drawn on the map.

- **MousePosition:** Defines the settings for the MousePosition control that displays the longitude and latitude coordinates of the location that the mouse cursor is hovering.

  - **Left:** The left position of the control in pixels. If the value is -1 the controls is displayed in its default position.
  - **PrefixText:** The prefix text used in front of the mouse position coordinates.
  - **Top:** The top position of the control in pixels. If the value is -1 the controls is displayed in its default position.
  - **Visible:** When set to true, the control is drawn on the map.

Maps methods

- **function DeleteAllMapMarker: Boolean;**

  This function removes all previously created markers.

- **function CreateMapMarker(Marker: TMarker): Boolean;**

  The function adds a new marker in the markers collection.

- **function DeleteMapMarker(Id: Integer):Boolean;**

  The function removes a marker from the markers collection.

- **function GetMapBounds: Boolean;**

  This function retrieves the bounds coordinates of the currently displayed map. The bounds are returned via the OnBoundsRetrieved event.

- **function MapPanTo(Latitude, Longitude: Double):Boolean;**

  This function performs a pan to a location set by latitude and longitude coordinates. This is useful to set a certain position in the center of the control canvas.

- **function MapZoomTo(Bounds: TBounds): Boolean;**

This function performs a zoom to fit the map inside the given bounds coordinates.

- **function DegreesToLonLat(StrLon, StrLat: string; var Lon,Lat: double): boolean;**

  This function converts degrees to longitude / latitude coordinates.

TTIWWebOSMaps events

The TTIWWebOSMaps control contains three types of events:

- Normal events:
  Cause a full page refresh when fired.
- Asynchronous events:
  Cause an asynchronous page refresh when fired.
- Client events:
  Cause no page refresh when fired. Provide the possibility to execute custom JavaScript in the browser. These events are grouped under the ClientEvents class.

Events

- **OnAsyncBoundsRetrieved(Sender: TObject; EventParams: TStringList; Bounds: TBounds):**

  ClientEvents.BoundsRetrieved(nelat, nelon, swlat, swlon):

  Event triggered after the GetBounds function has been called. This event returns the bounds coordinates of the currently displayed map.

- **OnAsyncDownloadFinish(Sender: TObject; EventParams: TStringList;):**

  ClientEvents.DownloadFinish():

  Event triggered when the map download is finished.

- **OnMapClick(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer; Button: TMouseButton):**

  OnAsyncMapClick(Sender: TObject; EventParams: TStringList; Latitude, Longitude: Double; X, Y: Integer; Button: TMouseButton):

  ClientEvents.MapClick (lat, lon, x, y):

  Event triggered when the map is clicked. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel coordinates in the control window.

- **OnAsyncMapMouseEnter(Sender: TObject; EventParams: TStringList; Latitude, Longitude: Double; X, Y: Integer):**

  ClientEvents.MapMouseEnter(lat,lon,x,y):

  Event triggered when the mouse cursor enters the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnAsyncMapMouseExit(Sender: TObject; EventParams: TStringList; Latitude, Longitude: Double; X, Y: Integer);**

  ClientEvents.MapMouseExit(lat,lon,x,y):

  Event triggered when the mouse cursor exits the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnAsyncMapMouseMove(Sender: TObject; EventParams: TStringList; Latitude, Longitude: Double; X, Y: Integer);**

  ClientEvents.MapMouseMove(lat,lon,x,y):

  Event triggered when the mouse is moved within the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnAsyncMapMove(Sender: TObject; EventParams: TStringList);**

  ClientEvents. MapMove():

  Event triggered when the entire map is moved (left mouse and drag) within the control.

- **OnAsyncMapMoveEnd(Sender: TObject; EventParams: TStringList)**

  ClientEvents.MapMoveEnd(lat, lon):

  Event triggered at the end of an entire map move (left mouse and drag) within the control.

- **OnAsyncMapMoveStart(Sender: TObject; EventParams: TStringList);**

  ClientEvents.MapMoveStart():

  Event triggered at the start of an entire map move (left mouse and drag) within the control.

- **OnAsyncMapZoomChange(Sender: TObject; EventParams: TStringList; NewLevel: Integer):**

  ClientEvents.MapZoomChange(zoomlevel):

  Event triggered when the zoom level is changed via the zoom control. The event returns the selected zoom level.

- **OnMarkerClick(Sender: TObject; IdMarker: Integer):**

  AsyncMarkerClick(Sender: TObject; EventParams: TStringList; IdMarker: Integer):

  ClientEvents.MarkerClick(id)

  Event triggered when a marker is clicked. Returns the marker id.

- **OnAsyncMarkerDrag(Sender: TObject; EventParams: TStringList; IdMarker: Integer; Latitude, Longitude: Double);**

  ClientEvents.MarkerDrag(id,lat,lon):

  Event triggered when a marker is dragged around the control. The event returns the marker id, latitude and longitude coordinates of the selected marker.

- **OnAsyncMarkerDragEnd(Sender: TObject; EventParams: TStringList; IdMarker: Integer; Latitude, Longitude: Double);**

  ClientEvents.MarkerDragEnd(id,lat,lon):

  Event triggered at the end of when a marker is dragged in the control. The event returns the marker id, latitude and longitude coordinates of the selected marker.

- **OnAsyncMarkerDragStart(Sender: TObject; EventParams: TStringList; IdMarker: Integer; Latitude, Longitude: Double);**

  ClientEvents.MarkerDragStart(id,lat,lon):

  Event triggered at the start of when a marker is dragged in the control. The event returns the marker id, latitude and longitude coordinates of the selected marker.

- **OnAsyncMarkerMouseDown(Sender: TObject; EventParams: TStringList; IdMarker: Integer; Latitude, Longitude: Double);**

  ClientEvents.MarkerMouseDown(id)

  Event triggered when the mouse cursor is over a marker and a mouse button is pressed. The event returns the marker id.

- **OnAsyncMarkerMouseEnter(Sender: TObject; EventParams: TStringList; IdMarker: Integer; Latitude, Longitude: Double);**

  ClientEvents.MarkerMouseEnter(id):

  Event triggered when the mouse cursor enters a marker. The event returns the marker id.

- **OnAsyncMarkerMouseExit(Sender: TObject; EventParams: TStringList; IdMarker: Integer; Latitude, Longitude: Double);**

  ClientEvents.MarkerMouseExit(id):

Event triggered when the mouse cursor leaves the marker. The event returns the marker id.

- **OnAsyncMarkerMouseUp(Sender: TObject; EventParams: TStringList; IdMarker: Integer; Latitude, Longitude: Double);**

  ClientEvents.MarkerMouseUp(id)

  Event triggered when the mouse cursor is over a marker and a mouse button is released. The event returns the marker id.

- **OnPolygonClick(Sender: TObject; IdPolygon: Integer):**

  OnAsyncPolygonClick(Sender: TObject; EventParams: TStringList; IdPolygon: Integer);

  ClientEvents.PolygonClick(id):

  Event triggered when a polygon is clicked. The event returns the polygon id.

- **OnAsyncPolygonDrag(Sender: TObject; EventParams: TStringList; IdPolygon: Integer; Coordinates: TCoordinates);**

  ClientEvents.PolygonDrag(id,co):

  Event triggered when a polygon is dragged around the control. The event returns the polygon id and coordinates of the selected polygon.

- **OnAsyncPolygonDragEnd(Sender: TObject; EventParams: TStringList; IdPolygon: Integer; var Allow: Boolean; Coordinates: TCoordinates);**

  ClientEvents.PolygonDragEnd(id,co):

  Event triggered at the end of when a polygon is dragged around the control. The event returns the polygon id and coordinates of the selected polygon.

- **AsyncPolygonDragStart(Sender: TObject; EventParams: TStringList; IdPolygon: Integer);**

  ClientEvents.PolygonDragStart(id):

  Event triggered at the start of when a polygon is dragged around the control. The event returns the polygon id and coordinates of the selected polygon.

- **OnAsyncPolygonMouseEnter(Sender: TObject; EventParams: TStringList; IdPolygon: Integer);**

  ClientEvents.PolygonMouseEnter(id):

  Event triggered when the mouse cursor enters a polygon. The event returns the polygon id.

- **OnAsyncPolygonMouseExit(Sender: TObject; EventParams: TStringList; IdPolygon: Integer);**

  ClientEvents.PolygonMouseExit(id):

  Event triggered when the mouse cursor leaves a polygon. The event returns polygon id.

- **OnPolylineClick(Sender: TObject; IdPolyline: Integer):**

  OnAsyncPolylineClick(Sender: TObject; EventParams: TStringList; IdPolyline: Integer);

  ClientEvents.PolylineClick(id):

  Event triggered when a polyline is clicked. The event returns the polyline id.

- **OnAsyncPolylineDrag(Sender: TObject; EventParams: TStringList; IdPolyline: Integer; Coordinates: TCoordinates);**

  ClientEvents.PolylineDrag(id,co):

  Event triggered when a polyline is dragged around the control. The event returns the polyline id and coordinates of the selected polyline.

- **OnAsyncPolylineDragEnd(Sender: TObject; EventParams: TStringList; IdPolyline: Integer; var Allow: Boolean; Coordinates: TCoordinates);**

  ClientEvents.PolylineDragEnd(id,co):

  Event triggered at the end of when a polyline is dragged around the control. The event returns the polyline id and coordinates of the selected polyline.

- **AsyncPolylineDragStart(Sender: TObject; EventParams: TStringList; IdPolyline: Integer);**

  ClientEvents.PolylineDragStart(id):

  Event triggered at the start of when a polyline is dragged around the control. The event returns the polyline id and coordinates of the selected polyline.

- **OnAsyncPolylineMouseEnter(Sender: TObject; EventParams: TStringList; IdPolyline: Integer);**

  ClientEvents.PolylineMouseEnter(id):

  Event triggered when the mouse cursor enters a polyline. The event returns the polyline id.

- **OnAsyncPolylineMouseExit(Sender: TObject; EventParams: TStringList; IdPolyline: Integer);**

  ClientEvents.PolylineMouseExit(id):

Event triggered when the mouse cursor leaves a polyline. The event returns polyline id.

- **AsyncWebOSMapsError(Sender: TObject; EventParams: TStringList; ErrorType: TErrorType);**

  ClientEvents.WebOSMapsError(id):

  Event triggered when an error is received. This event returns the error type.

TTIWWebOSMaps Keyboard

When TTIWWebOSMaps.MapOptions.EnableKeyboard is set to true, keyboard support is enabled.

Below is a list of keys that will allow you to navigate through the map without using the mouse.

Panning the map

Arrow key up, Arrow key down, Arrow key left, Arrow key right, move the map in the corresponding directions a pixel at a time.

Page up key, Page down key, Home key and End key move the map up, down, left and right within the control a page at a time.

Zooming the map

The Plus (+) key functions as a click on the plus button of the zoom control, and performs a zoom in the map with one step.

The Minus (-) key performs the same as a click of the minus button in the zoom control, and performs a zoom out in the map with one step.

TTIWWebOSMaps Sample code

Sample 1

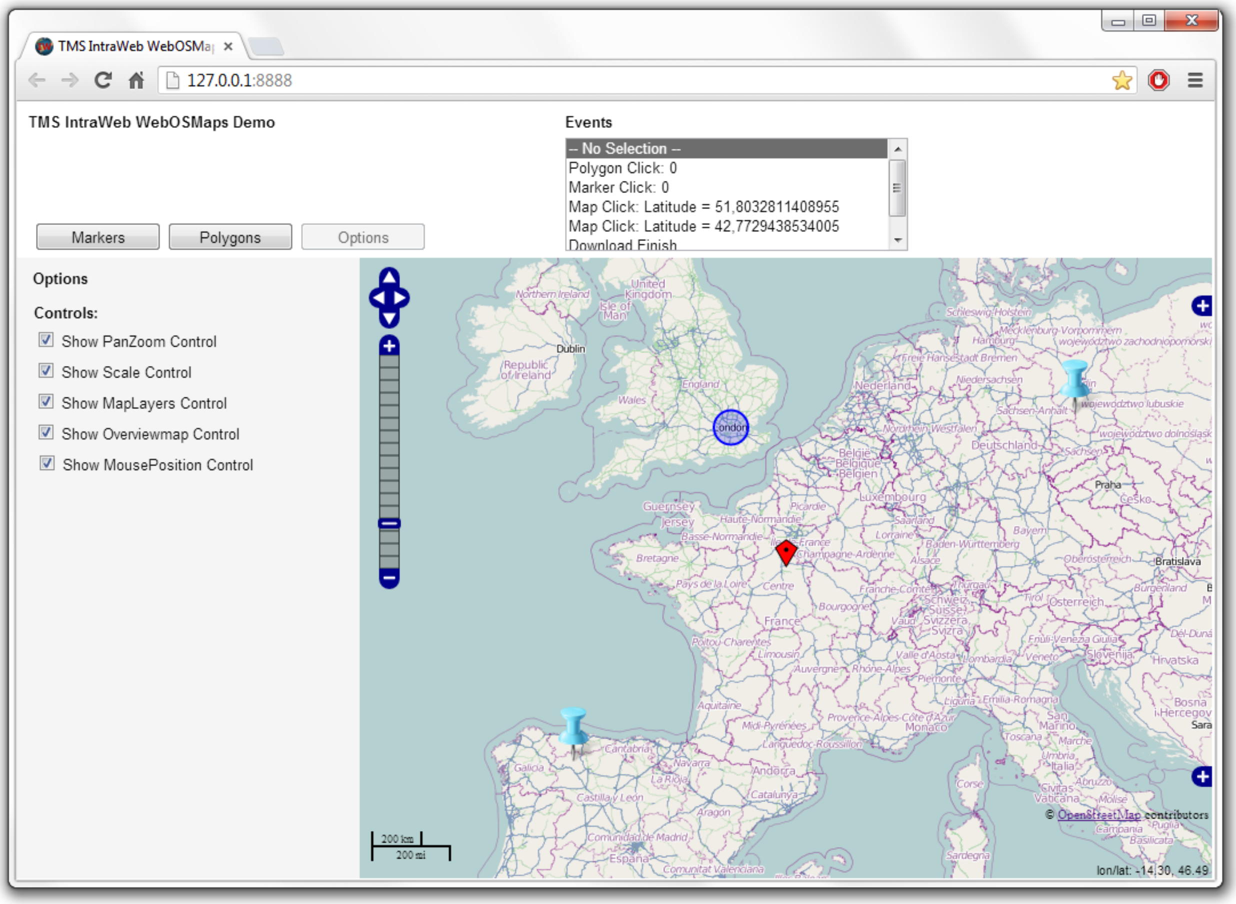
This example shows how to format an icon URL string for proper image loading.

```
Var
  MarkerIcon: string;

begin

  MarkerIcon := StringReplace('File://' + GlobalPath +
'Files\Thumbnails\EiffelTower.png', '\', '/',[rfReplaceAll, rfIgnoreCase]);

end;
```

TTIWWebOSMaps demo

The TMS IntraWeb WebOSMaps Demo program shows the various configuration possibilities of the TTIWWebOSMaps component. It allows to interactively set various properties and the changes will be immediately reflected in the map.

Main screen:



TMS IntraWeb WebOSMaps Demo

Events

-- No Selection --
Polygon Click: 0
Marker Click: 0
Map Click: Latitude = 51,8032811408955
Map Click: Latitude = 42,7729438534005
Download Finish

Options

Controls:

- ☑ Show PanZoom Control
- ☑ Show Scale Control
- ☑ Show MapLayers Control
- ☑ Show Overviewmap Control
- ☑ Show MousePosition Control

Note that a selection of asynchronous events that occur are displayed in the top right of the browser screen.

Markers menu

From here a marker can be set at specific latitude/longitude coordinates or via a mouseclick on the map. All markers can be deleted.

Options menu

From here the visibility of the various controls on the WebOSMaps can be set.

Polygons menu

From here a Line, Circle or Square polygon can be added at a latitude/longitude coordinate. All polygons can be deleted.