COVER PAGE

DELIVERABLE

Project Acronym: i-locate

Grant Agreement number: 621040

Project Title: Indoor/outdoor LOCation and Asset management Through open gEodata

D2.8 - Mobile client and portal infrastructure

(Accompanying report)

Revision: v.1.0

Authors:

Lucian Brancovean (INDSOFT), Danny Schumann (ZIGPOS), Catalin Popa (INDSOFT)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Author	Organisati	Description
v.01	05.05.2015	Lucian Brancovean	INDSOFT	Structure and content
v.02	19.06.2015	Danny Schumann	ZIGPOS	Mobile App.
v.03	23.06.2015	Catalin Popa	INDSOFT	Document Update
v.04	25.06.2015	Danny Schumann	ZIGPOS	Mobile App API update
v.05	26.06.2015	Catalin Popa	INDSOFT	Various updates and reformatting
v.1.0	30.06.2015	Giuseppe Conti, Gianni Rangoni	Trilogis	Final review and quality check

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

1 List of references

Number	Full reference
1	IndoorGML specification, OGC Specification. Available online at: http://www.opengeospatial.org/standards/indoorgml
2	
3	

2 Table of Acronyms

Acronym	Description
CityGML	City Geographic Markup Language
GeoJSON	Open standard format for encoding collections of simple geographical features using JSON
IndoorGML	Indoor Geography Markup Language
ID	Identifier
JSON	JavaScript Object Notation
OGC	Open Geospatial Consortium
REST	REpresentational State Transfer
SRID	ID of the reference system
URL	Uniform Resource Locator
WFS	Web Feature Service
WMS	Web Mapping Service
XML	Extensible Markup Language

3 Executive Abstract

The present document has been edited to be read as an installation guide for the i-locate portal and it also provides guidelines for the configuration of geoserver and of the mobile application used for editing and updating the staging maps for site locations.

The document is organised as follows: Chapter 6 describes the installation process steps for the tools used, while Chapter 7 gives a detailed approach to the geoserver configuration, with particular attention being paid to layers and styles. Finally, Chapter 8 describes the services and APIs used by the mobile application to ensure communication with the i-locate portal.

Table of content

1 LIST OF REFERENCES 3
2 TABLE OF ACRONYMS 4
3 EXECUTIVE ABSTRACT 5
4 INTRODUCTION 8
5 PREREQUISITES 9
6 INSTALLATION 10
6.1 Obtain Source Code 10
6.2 Install PostgreSQL 10
6.3 Configure Maven 10
6.4 Compile and package portal 10
6.5 Generate database structure 10
6.6 Install Apache Tomcat 11
6.7 Deploy in Apache Tomcat 11
7 GEOSERVER CONFIGURATION 12
7.1 Libraries 12
7.2 Workspace 12
7.3 Store 12
7.4 Styles 13
7.4.1 Style room 13
7.4.2 Style outline 14
7.4.3 Style wall 15
7.5 Layers 15
7.5.1 Layer indoor_line 16
7.5.2 Layer production 16
7.5.3 Layer staging 17
7.5.4 Layer production_o 17
7.5.5 Layer staging_o 18
7.5.6 Layer production_w 18
7.5.7 Layer staging_w 18
7.6 Layer Groups 19
7.6.1 Layer group production_g 19
7.6.2 Layer group staging_g 20
8 REST-API FOR CLIENT APPLICATIONS 21
8.1 Sites 21
8.2 Sections 21

8.2.1 List of Sections 21
8.2.2 Up/Download section maps 21
8.2.3 GeoJSON data format for sections 22
8.2.4 Geometry Types 22
8.2.5 Properties 22
8.2.6 Sample section with 2 rooms 23
8.3 Navigation Graph 23
8.3.1 Simple IndoorGML 24
8.3.2 Data Format 24

4 Introduction

This document is complements the accompanying report edited for the deliverable 2.7 (The i-locate “virtual hub”), providing information on the steps necessary to install the i-locate portal, starting from the available source code. The intended audience of this document is typically made of IT professionals, system engineers, network administrators. The present document is therefore very technical in nature and it is not intended for end users.

The i-locate web portal is a java web application, packaged as a “war” file, and can be run in a servlet container. Together with the portal, an instance of Geoserver must be run, which is also a Java web application, packaged as a “war” file.

5 Prerequisites

The web portal can run on any operating system where a Java servlet container can run (wherever Java Development Kit 7 is available). It has been developed and tested on Windows 7 and Ubuntu 14.04.

The following software is required:

- PostgreSQL database server, version 9.3 or newer.
- PostGIS extension for PostgreSQL, version 2.1 or newer.
- Java Development Kit, version 7.
- Apache Tomcat version 7 (or any Servlet 3.0 compatible container).
- Geoserver 2.5.1 or newer.
- Apache Maven 3 or newer.

6 Installation

In order to install the web portal, some steps are required for pre-configure, compile and run the whole infrastructure. The following paragraphs will explain what are the steps and a detailed guide on how to follow them.

6.1 Obtain Source Code

The source code of the portal is freely available online. Instructions on obtaining the source code can be found on the project website, at www.i-locate.eu. The code is hosted in a git environment; hence git knowledge is required to successfully obtain the sources.

6.2 Install PostgreSQL

Once the source code is downloaded, there should be already a working instance of PostgreSQL, with PostGIS extension. The details of the database installation are beyond the scope of this document, but can be easily found on the Internet from the official PostgreSQL/PostGIS websites.

6.3 Configure Maven

The following Maven profile needs to be defined, for example in the global *settings.xml*:

```
<profile>
    <id>ilocate</id>
    <properties>
         <db.jndiName>jdbc/ilocateDataSource</db.jndiName>
         <db.driver>org.postgresql.Driver</db.driver>
         <db.url>jdbc:postgresql://127.0.0.1:5432/ilocate</db.url>
<db.username>postgres</db.username>
         <db.password>postgres</db.password>
         <geoserver.baseUrl>http://portal.i-locate.eu:8080/geoserver</geoserver.baseUrl>
    </properties>
</profile>
```

The configuration details must be adapted to the local setup where the system will run, for example the database access details depend on the local installation and configuration of PostgreSQL.

6.4 Compile and package portal

In order to compile the source code, Maven has to be already installed and the following command must be provided at the portal source code root level folder. This will produce the *portal-web.war* file, which is the packaged portal, ready for the deployment in Tomcat:

mvn clean package –P ilocate

The resulting war file can be found in the *target* directory of the project.

6.5 Generate database structure

The following command will connect to the database using the settings specified in the Maven profile defined earlier (see section 6.3), and generate the required database structure:

mvn –P ilocate resources:resources liquibase:update

The result of this command will be a number of tables, views and a sequence in the database, which will be used by the portal for data storage.

6.6 Install Apache Tomcat

At this point there should already be a working installation of Apache Tomcat 7. The details of the servlet container installation are beyond the scope of this document, but can be easily found on the internet at the official Apache Tomcat webpage and depends on the OS on which the portal will run. Any other compatible servlet container can be used although not tested and/or not recommended.

6.7 Deploy in Apache Tomcat

In order to deploy the portal and the geoserver web archives in the servlet container to Apache Tomcat, copy the *portal-web.war* and the *geoserver.war* files into the *webapps* directory of the above mentioned servlet container. When Tomcat starts, both the portal and geoserver will be available on the path reflecting the *war* file names.

7 Geoserver Configuration

The default geoserver account is *user:admin* and *password:geoserver*. Log in to geoserver and create the configuration described below. Security configuration and additional configuration can be found at the official Geoserver website.

7.1 Libraries

Remove (or rename to .bak) the file:

webapps\geoserver\WEB-INF\lib\postgresql-8.4-701.jdbc3.jar

If it is available and not renamed, there will be a conflict with the postgres-jdbc and postgis jars provided by Apache Tomcat, and Geoserver will be unable to define and use PostGIS data stores.

7.2 Workspace

The Geoserver workspace will contain all i-locate related configuration. The following parameters must be provided in the Geoserver workspace section in order to create the i-locate workspace.

Name:	*ilocate*
Namespace URL:	*http://www.i-locate.eu/*
Services:	*WFS, WMS*

7.3 Store

The store represents the geoserver connections to the PostgreSQL/PostGIS database of the portal, hence it is important and required to setup store connection to the available database. The following parameters must be provided in order to create a working connection. User and password of the database depends on the configuration in place as in 6.3.

PostGIS Database

Workspace:	*ilocate*
Data Source Name:	*postgis-vm*
Enabled:	*yes*
host:	*localhost*
port:	*5432*
database:	*ilocate*
schema:	*public*
user:	*postgres*
password:	*postgres*

The other settings can be kept with their default values.

7.4 Styles

The styles section defines the appearance of rooms, walls and outline in the indoor map rendered by geoserver. Colors, the border thickness and the font used to display room names, are defined here.

7.4.1 Style room

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
 xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
 xmlns="http://www.opengis.net/sld"
 xmlns:ogc="http://www.opengis.net/ogc"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <!-- a Named Layer is the basic building block of an SLD document -->
  <NamedLayer>
    <Name>default_polygon</Name>
    <UserStyle>
    <!-- Styles can have names, titles and abstracts -->
      <Title>Default Polygon</Title>
      <Abstract>A sample style that draws a polygon</Abstract>
      <!-- FeatureTypeStyles describe how to render different features -->
      <!-- A FeatureTypeStyle for rendering polygons -->
      <FeatureTypeStyle>
        <Rule>
          <Name>zoomed_in</Name>
          <Title>Yellow Polygon with Red Outline</Title>
          <Abstract>A polygon with a yellow fill and a 1 pixel red outline</Abstract>
          <MaxScaleDenominator>600</MaxScaleDenominator>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#ffe4e5</CssParameter>
            </Fill>
            <Stroke>
              <CssParameter name="stroke">#ff8b90</CssParameter>
              <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
          </PolygonSymbolizer>

        </Rule>
        <Rule>
          <Name>zoomed_in</Name>
          <Title>Yellow Polygon with Red Outline</Title>
          <Abstract>A polygon with a yellow fill and a 1 pixel red outline</Abstract>
          <MaxScaleDenominator>300</MaxScaleDenominator>
          <TextSymbolizer>
            <Geometry>
              <ogc:Function name="centroid">
                <ogc:PropertyName>location</ogc:PropertyName>
              </ogc:Function>
            </Geometry>
            <Label>
              <ogc:PropertyName>room_name</ogc:PropertyName>
            </Label>
            <Font>
```

```
            <CssParameter name="font-family">Arial</CssParameter>
            <CssParameter name="font-weight">bold</CssParameter>
            <CssParameter name="font-size">12</CssParameter>
          </Font>
          <LabelPlacement>
            <PointPlacement>
               <AnchorPoint>
                 <AnchorPointX>
                   0.5
                 </AnchorPointX>
                 <AnchorPointY>
                   0.5
                 </AnchorPointY>
               </AnchorPoint>
            </PointPlacement>
          </LabelPlacement>
          <Halo>
            <Radius>2</Radius>
            <Fill>
              <CssParameter name="fill">#FFFFFF</CssParameter>
            </Fill>
          </Halo>
          <!--<VendorOption name="group">yes</VendorOption>-->
          <VendorOption name="autoWrap">100</VendorOption>
          <!--<VendorOption name="goodnessOfFit">0.2</VendorOption>-->
          <!--<VendorOption name="conflictResolution">false</VendorOption>-->
          <VendorOption name="maxDisplacement">150</VendorOption>
        </TextSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

7.4.2 Style outline

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
 xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
 xmlns="http://www.opengis.net/sld"
 xmlns:ogc="http://www.opengis.net/ogc"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <!-- a Named Layer is the basic building block of an SLD document -->
  <NamedLayer>
    <Name>outline</Name>
    <UserStyle>
      <Title>Building outline</Title>
      <Abstract>A sample style that draws a polygon</Abstract>
      <!-- FeatureTypeStyles describe how to render different features -->
      <!-- A FeatureTypeStyle for rendering polygons -->
      <FeatureTypeStyle>
        <Rule>
          <Name>rule1</Name>
          <Title>Yellow Polygon with Orange Outline</Title>
          <Abstract></Abstract>
          <MaxScaleDenominator>4000</MaxScaleDenominator>
          <PolygonSymbolizer>
```

```
            <Fill>
              <CssParameter name="fill">#f8f7ce</CssParameter>
            </Fill>
            <Stroke>
              <CssParameter name="stroke">#e0851a</CssParameter>
              <CssParameter name="stroke-width">2</CssParameter>
            </Stroke>
          </PolygonSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

7.4.3 Style wall

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
 xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
 xmlns="http://www.opengis.net/sld"
 xmlns:ogc="http://www.opengis.net/ogc"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <!-- a Named Layer is the basic building block of an SLD document -->
  <NamedLayer>
    <Name>default_polygon</Name>
    <UserStyle>
    <!-- Styles can have names, titles and abstracts -->
      <Title>Default Polygon</Title>
      <Abstract>A sample style that draws a polygon</Abstract>
      <!-- FeatureTypeStyles describe how to render different features -->
      <!-- A FeatureTypeStyle for rendering polygons -->
      <FeatureTypeStyle>
        <Rule>
          <Name>rule1</Name>
          <Title>Yellow Polygon with Red Outline</Title>
          <Abstract>A polygon with a brown fill and a 1 pixel brown outline</Abstract>
          <MaxScaleDenominator>600</MaxScaleDenominator>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#D96104</CssParameter>
            </Fill>
            <Stroke>
              <CssParameter name="stroke">#D96104</CssParameter>
              <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
          </PolygonSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

7.5 Layers

The layers section defines how the layers have to be created in the Geoserver configuration in order to run the i-locate portal with indoor maps.

7.5.1 Layer indoor_line

This is an internal layer used by the portal to provide the indoor navigation graph. The following parameters must be provided in order to configure the layer correctly

Name: *indoor_line*
Enabled: *yes*
Advertised: *yes*
Title: *indoor_line*
Native SRS: *EPSG:4326*
Declared SRS: *EPSG:4326*
SRS Handling: *Force declared*
Bounding Boxes: *-180 to +180*

7.5.2 Layer production

This layer displays the rooms in the production version of the map. Production version contains data that are publicly available to all users and promoted from draft version.

Create layer with "Create new SQL view"

Name: production
SQL statement: *SELECT id, location, room_name FROM production WHERE map_level = (%level% - 10)*

The (level - 10) expression is necessary to avoid negative parameters to parameterized layers. The level parameter is added 10 on the OpenLayers side, and subtracted 10 here in geoserver.

SQL view parameters:

Name: *level*
Default value: *10 (this actually represents level 0)*
Validation regexp: *^[\d]+$*

Attributes:

Name: *id*
Type: *BigDecimal*
Identifier: *yes*

Name: *location*
Type: *Polygon*
SRID: *4326*

Name: *room_name*

Bounding Boxes: *-180 to +180*

Publishing tab:

Default style: *room (as defined before on this document)*

7.5.3 Layer staging

This layer displays the rooms in the staging version of the map. Staging version contain data that is defined as ‘draft’ and not available to all users.

Create layer with "Create new SQL view"

Name: staging
SQL statement: *SELECT id, location, room_name FROM staging WHERE map_level = (%level% - 10)*

The (level - 10) expression is necessary to avoid negative parameters to parameterized layers. The level parameter is added 10 on the client side, and subtracted 10 in Geoserver.

SQL view parameters:

Name: *level*
Default value: *10 (this actually represents level 0)*
Validation regexp: *^[\d]+$*

Attributes:

Name: *id*
Type: *BigDecimal*
Identifier: *yes*

Name: *location*
Type: *Polygon*
SRID: *4326*

Name: *room_name*

Bounding Boxes: *-180 to +180*

Publishing tab:

Default style: *room (as defined before on this document)*

7.5.4 Layer production_o

This layer displays the outline in the production version of the map. The outline is unique for each pilot site and covers the whole area of all floors where the pilot site is.

Create from SQL view production_o.

Bounding Boxes: *-180 to +180*

Publishing tab:

Default style: *outline (as defined before on this document)*

7.5.5 Layer staging_o

This layer displays the outline in staging version of the map.

Create from SQL view staging_o

Bounding Boxes: *-180 to +180*

Publishing tab:

Default style: *outline (as defined higher on this page)*

7.5.6 Layer production_w

This layer displays the walls in the production version of the map.

Create layer with "Create new SQL view"

Name: *production_w*
SQL statement: *SELECT id, location FROM production_w WHERE map_level = (%level% - 10)*

The (level - 10) expression is necessary to avoid negative parameters to parameterized layers. The level parameter is added 10 on the client side, and subtracted 10 in geoserver.

SQL view parameters:

Name: *level*
Default value: *10 (this actually represents level 0)*
Validation regexp: *^[\d]+$*

Attributes:

Name: *id*
Type: *BigDecimal*
Identifier: *yes*

Name: *location*
Type: *Polygon*
SRID: *4326*

Name: *production_w*

Bounding Boxes: *-180 to +180*

Publishing tab:

Default style: *wall (as defined higher on this page)*

7.5.7 Layer staging_w

This layer displays the walls in the staging version of the map.

Create layer with "Create new SQL view"

Name: *staging_w*
SQL statement: *SELECT id, location FROM staging_w WHERE map_level = (%level% - 10)*

The (level - 10) expression is necessary to avoid negative parameters to parameterized layers. The level parameter is added 10 on the client side, and subtracted 10 in geoserver.

SQL view parameters:

Name: *level*
Default value: *10 (this actually represents level 0)*
Validation regexp: *^[\d]+$*

Attributes:

Name: *id*
Type: *BigDecimal*
Identifier: *yes*

Name: *location*
Type: *Polygon*
SRID: *4326*

Name: *staging_w*

Bounding Boxes: *-180 to +180*

Publishing tab:

Default style: *wall (as defined higher on this page)*

7.6 Layer Groups

Layer groups are used in geoserver to group logically and show multiple layers at a time, reducing client side complexity and number of network request.

7.6.1 Layer group production_g

This layer groups together everything needed for the production version of the map.

Create a layer group with name `production_g`.

Workspace: *ilocate*

Bounds: *-180 to 180*

Add layers in order:

1. `production_o` style: `outline`
2. `production` style: `room`

3. `production_w` style: `wall`

7.6.2 Layer group staging_g

This layer groups together everything needed for the staging version of the map.

Create a layer group with name staging_g.

Workspace: *ilocate*

Bounds: *-180 to 180*

Add layers in order:

1. `staging_o` style: `outline`
2. `staging` style: `room`
3. `staging_w` style: `wall`

8 REST-API for Client applications

The i-locate portal provides a REST-API that can be used by client applications, for instance by the Mobile client. The API provides read and write access to the i-locate specific data of the portal, like indoor maps and indoor navigation graphs. The content data is formatted as JSON.

Each i-locate location is represented by a site. Each site is composed of sections which define the rooms, walls and the outline of a building on a specific level. The site may also have an indoor navigation graph an IndoorGML format.

The following sections describe the details of the REST resources. The so called BASE_URL is the URL of the i-locate server where the service is running.

8.1 Sites

It is possible to query the site descriptors of all available sites. Each site has a unique ID, a name and a global position. The following parameters describe how to query for obtaining a list of sites.

Rest method and URL: GET http://BASE_URL/sites/

Return Type and body: JSON-String array with all map descriptors

```
[
        { “id”: 1, “name”: “Brukenthal”, “position” : { “latitude”: 45, “longitude”: 14} },
        { “id”: 2, “name”: “HTW”, “position” : { “latitude”: 46, “longitude”: 12} }
]
```

8.2 Sections

A list of sections of each site is available to be queried (obtain the list of sections), upload the GeoJson formatted maps for each section or download them.

8.2.1 List of Sections

One can get the list of sections available for a pilot. Each section has some properties; ID, name, level and type. Sections can have one of the three types that are room, wall and outline.

Example REST method and URL: *GET http://BASE_URL/sites/<site-id>/sections*

Return Type and body:

```
[
        { “id”: 1, “name”: “Rooms 0”, “level” : 0, “type” : ”room”},
        { “id”: 2, “name”: “Walls 0”, “level” : 0, “type” : ”wall”},
        { “id”: 3, “name”: “Level Outline”, “level” : null, “type” : ”outline”}
]
```

8.2.2 Up/Download section maps

Each section can be described and produced as GeoJSON, containing all floor plan data for each selected section. It is possible retrieve and update each section layout in the following manner:

Retrieve section data

Rest method and URL: *GET http://BASE_URL/sites/<site-id>/section/<section-id>*

Request Type and body: GeoJSON formatted section

Update section data

Request method and URL: *POST http://BASE_URL/sites/<site-id>/section/<section-id>*

- Header: Content-type: application/json
- Body Data: GeoJSON formatted section

8.2.3 GeoJSON data format for sections

The GeoJSON data format for describing section is following the IndoorJSON extension of GeoJSON with modifications for being compliant with the i-locate portal and the IndoorGML standard.

8.2.4 Geometry Types

The GeoJSON object may be represented in one of the following geometry types:

GeoJSON geometry type	Usage
Polygon	Rooms, Outline
Geometry collection	combination of the previous types

8.2.5 Properties

Each geometry type may have the following properties:

IndoorJSON Property	GeoJSON geometry type	Description
level	any	0 for ground level, 1 for first…
geomType	Polygon	Room, Wall

	FeatureCollection	Section
id	Polygon, FeatureCollection	
name	any	Room name, room number..

8.2.6 Sample section with 2 rooms

In this example, each Section is a Feature collection, each Room is a Polygon feature type with the property “geomType”:”room”.

```
{ "type": "FeatureCollection",
    "features":
     { "type": "Feature",
         "geometry": {
           "type": "Polygon",
           "coordinates": [
             [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
               [100.0, 1.0], [100.0, 0.0] ]
             ]
         },
         "properties": {
           "name": "Room A",
           "id": 1,
           "geomType": "room",
           "level": 0
           }
         }
       ]
     },
     { "type": "Feature",
         "geometry": {
           "type": "Polygon",
           "coordinates": [
             [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
               [100.0, 1.0], [100.0, 0.0] ]
             ]
         },
         "properties": {
           "name": "Room B",
           "id": 2,
           "geomType": "room",
           "level": 0
           }
         }
       ]
     }
     "properties": {
           "name": "Section Name",
           "geomType": "section",
           "id": 1,
```

```
    }
}
```

8.3 Navigation Graph

The navigation graph for indoor navigation is defined as a set of points and edges connecting those points. To reduce the complexity of IndoorGML representation on the mobile client, a simple JSON representation of the navigation graph was defined to increase mobile devices performances and data exchange speed.

8.3.1 Simple IndoorGML

It is possible to retrieve and update navigation graphs on the i-locate portal by using the following services:

- To retrieve a graph:
  - **Rest method and URL**: `GET http://BASE_URL/sites/<site-id>/indoorGml/`
  - **Return Type:** JSON formatted connections between navigation points
- To update a graph:
  - **Rest method and URL**: `POST http://BASE_URL/sites/<site-id>/indoorGml/`
  - **Return Type**: JSON formatted connections between navigation points

8.3.2 Data Format

The data format for the navigation graph as explained above contains basically a list of points and edges. Each point has an Identifier, a global position (in latitude and longitude coordinates), the floor level information and a type, that can be room, door, stairs, elevator, etc.

Each transition is described as edge between two points and may have optional properties like a weight parameter (for routing purposes).

The JSON formatted navigation graph is compliant with the following format:

```
{
        “points” : [{ “id”:0, “long”:12, “lat”:45, “level”:0, “type”:”room” }, …],
        “edges” : [{ “point1”:0, “point2”:1, “properties”:{} }]
}
```