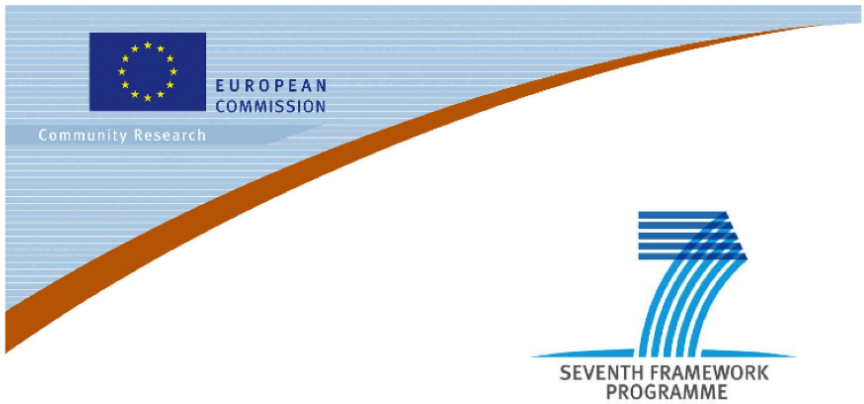

Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

POI Data Provider Installation and Administration Guide

Project full title: Future Internet Core
Project acronym: FI-Core
Contract No.: 632893

Contents

POI Data Provider - Installation and Administration Guide

by Ari Okkonen, Adminotech Oy

Introduction

The purpose of this document is to provide the required information for a system administrator in order to install and configure the POI (Points of Interest) Data Provider generic enabler reference implementation. The POI GE is implemented as a RESTful web service using PHP programming language. It is described in more detail in FIWARE.OpenSpecification.MiWi.POIDataProvider.

NOTES:

1) If you are installing a secure POI server, use directory "`/var/www/ssl`" everywhere in stead of "`/var/www/html`".
2) All editor invocation commands in this document are for the Nano editor. `$ sudo nano ...` Of course, you are free to use your favorite editor to edit the files.

Document Releases

This document is associated to the latest release of the POI Data Provider. Links to versions related to earlier software releases are in the table below.

Release	Date	Description
r3.3	2014-09-17	Original release - a POI belongs to exactly one category
r5.1	2016-04-07	Dynamic POIs & Quality Boost
r5.4	2016-09-09	This release - Access Control

System Requirements

Hardware Requirements

The POI Data Provider should work on any modern PC computer that is capable of running Ubuntu Server 14.04. Therefore, the bare minimum requirements are:

- 300 MHz x86 processor
- 192 MiB of system memory (RAM) (256 MiB for a virtual installation)
- 1.4 GB of disk space

For a small practical deployment, the recommended system is:

- 1 GHz Dual core CPU
- 4 GB of system memory (RAM)
- 40 GB of disk space

The hardware needs of the POI Data Provider are mainly dominated by the databases (PostgreSQL and MongoDB), and as such the two most important factors are:

- memory size (the bigger the better)
- disk size
- disk speed (e.g. using SSD drives)

You can have a rough estimate for the required disk space by using the following formula:

So, for example for one million (1000000) POIs, you get the following estimate for data size on disk:

However, this estimate may be inaccurate for many cases, as the mean POI size can be much smaller or larger than 10 kilobytes.

Operating System Support

The implementation and this installation guide have been tested with Ubuntu 14.04. Other Linux distribution may need modifications in installation procedure and configuration files.

Software Requirements

In order to have the POI Data Provider up and running, the following software is required:

- PostgreSQL 9.3
- PostGIS 2.1.x spatial database extension
- MongoDB 2.0.x
- Apache HTTP Server 2.2.x
- (Or basically any HTTP server with PHP support)
- PHP 5.x
- PostgreSQL module
- MongoDB module
- JSON Schema for PHP 1.4.3 [1]
- pecl_http-1.7.6 module

Software Installation and Configuration

Update package lists

Get up-to-date package lists from update servers:

```
$ sudo apt-get update
```

Installing required packages

The required software packages can be installed using the 'apt-get' command-line package installation tool:

```
$ sudo apt-get install -y postgis postgresql-9.3-postgis-2.1
$ sudo apt-get install mongodb
$ sudo apt-get install apache2
$ sudo apt-get install php5 php5-pgsql
$ sudo apt-get install git
```

The installation of the MongoDB module for PHP5. :

```
$ sudo apt-get install php-pear php5-dev gcc make
$ sudo pecl install mongo
```

The installation of the Pecl_HTTP module for PHP5. This enables use of HTTP requests to obtain dynamic data from other sites. Note the version, because the interface changes to the version 2, and the version 3 is totally incompatible with PHP5.

```
$ sudo apt-get install libcurl3-openssl-dev
$ sudo pecl install pecl_http-1.7.6
```

Add these lines to /etc/php5/apache2/php.ini:

```
extension=mongo.so
extension=raphf.so
extension=propro.so
extension=http.so
```

You may use e.g.

```
$ sudo nano /etc/php5/apache2/php.ini
```

or some other editor.

Enable access control in per-directory basis. The POI DP uses .htaccess file to protect external access keys for dynamic POIs.

Change the following line in the file /etc/apache2/apache2.conf within <Directory /var/www/> section:

```
AllowOverride None
```

—> Change to —>

```
AllowOverride All
```

Restart Apache web server:

```
$ sudo /etc/init.d/apache2 restart
```

Configuring PostGIS

1) Create GIS database user:

```
$ sudo -u postgres createuser gisuser
```

Answer "n" to all questions.

2) Create database owned by that user

```
$ sudo -u postgres createdb --encoding=UTF8 --owner=gisuser poidatabase
```

3) Activate PostGIS on the created database:

```
$ sudo -u postgres psql -d poidatabase -f /usr/share/postgresql/9.3/contrib/postgis-2.1/postgis.sql
$ sudo -u postgres psql -d poidatabase -f /usr/share/postgresql/9.3/contrib/postgis-2.1/spatial_ref_sys.sql
$ sudo -u postgres psql -d poidatabase -f /usr/share/postgresql/9.3/contrib/postgis-2.1/postgis_comments.sql
$ sudo -u postgres psql -d poidatabase -c "GRANT SELECT ON spatial_ref_sys TO PUBLIC;"
$ sudo -u postgres psql -d poidatabase -c "GRANT ALL ON geometry_columns TO gisuser;"
```

4) Enable UUID functions for that database:

```
$ sudo -u postgres psql -d poidatabase -c 'create extension "uuid-ossp";'
```

5) Grant local access to the database:

Before you can access the database, you must edit PostgreSQL configuration to allow local unix socket connections (from the same computer where the database is running) without password.

Edit the file /etc/postgresql/9.3/main/pg_hba.conf and change the following line:

```
# "local" is for Unix domain socket connections only
local   all             all                                     peer
```

—> Change to —>

```
# "local" is for Unix domain socket connections only
local   all             all                                     trust
```

6) Restart PostgreSQL

```
$ sudo /etc/init.d/postgresql restart
```

Installing POI Data Provider

1) Fetch the POI Data Provider from GitHub:

```
$ git clone https://github.com/Chiru/FIWARE-POIDataProvider.git
```

2) Create required database tables using the provided script:

```
$ cd FIWARE-POIDataProvider/install_scripts
$ ./create_tables.sh
$ cd ../..
```

3) Choice between secure (https) and unsecure (http) access to the data provider depends on required confidentiality and dependability of the service. Plaintext http access is easily eavesdropped and intercepted. Do not use it for confidential or dependable data. Encrypted https access requires more determination to intercept. However, https requires more work to set up and manage. See: Wikipedia HTTPS. Choose either:

   A. Unsecure (http)

   Copy the folder FIWARE-POIDataProvider/php from the cloned project under the current working directory, e.g. to /var/www/html/poi_dp

```
$ sudo cp -r FIWARE-POIDataProvider/php /var/www/html/poi_dp
```

   Installation of JSON Schema for PHP

```
$ wget http://getcomposer.org/composer.phar
$ php composer.phar require justinrainbow/json-schema:1.4.3
$ sudo cp -r vendor /var/www/html/poi_dp/
```

   More information about the JSON Schema for PHP implementation can be found at [3].

   B. Secure (https)

   Copy the folder FIWARE-POIDataProvider/php from the cloned project under the current working directory, e.g. to /var/www/ssl/poi_dp

```
$ sudo cp -r FIWARE-POIDataProvider/php /var/www/ssl/poi_dp
```

   Installation of JSON Schema for PHP

```
$ wget http://getcomposer.org/composer.phar
$ php composer.phar require justinrainbow/json-schema:1.4.3
$ sudo cp -r vendor /var/www/ssl/poi_dp/
```

   More information about the JSON Schema for PHP implementation can be found at [3].

Enable Handling of Cross-origin Resource Sharing and URL Rewrite in Apache

Cross-origin Resource Sharing (CORS) is required if the POI-DP client is a web application that is hosted on a different domain than the POI-DP backend. In practice this means that the POI-DP Apache server needs to add the following HTTP header for each response:

```
Access-Control-Allow-Origin "*"
```

Rewrite is used to default the .php extension from service requests. E.g. http//www.example.org/poi_dp/radial_search -> http//www.example.org/poi_dp/radial_search.php .

Enable mod_headers and mod_rewrite Apache modules:

```
$ sudo a2enmod headers
$ sudo a2enmod rewrite
$ sudo service apache2 restart
```

Set site information

1) Copy `poi_dp/site_info_t.json` to `poi_dp/site_info.json` .

```
$ cd /var/www/html/poi_dp
$ sudo cp site_info_t.json site_info.json
```

2) Edit `poi_dp/site_info.json` to show the correct data for your site. E.g.:

```
$ sudo nano site_info.json
```

Enabling secure server (SSL)

Optional feature - for confidential or dependable information

In general you have to enable the ssl mode in the server

```
$ sudo a2enmod ssl
$ sudo service apache2 restart
$ sudo mkdir /etc/apache2/ssl
```

Then you have to set up the secure certificate.

Professional secure sites need a certificate signed by a trusted authority. Obtaining a SSL certificate is explained at How To Order An SSL Certificate.

An experimental or hobby site can do with a self-signed certificate. Setting up a site with such can be done according to instructions at How To Create a SSL Certificate on Apache for Ubuntu 14.04.

Hint: the "Common Name (e.g. server FQDN or YOUR name)" seems to need to be the domain name of your server.

Edit the server configuration.

```
$ sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Detailed editing instructions at How To Create a SSL ….. **NOTE:** In default-ssl.conf set the DocumentRoot to point to the secure server root /var/www/ssl .

```
...
DocumentRoot /var/www/ssl
...
```

Setting up user authentication

User authentication is needed, if

- the server will contain confidential data not for anyone's eyes, or
- the REST interface will be used to add or update data.

Remove NGSI-10 support for confidential data

NGSI-10 support does not contain access control. If not all the POIs in the server are open data, remove the directory `/var/www/html/poi_dp/ngsi10/` .

Register the POI data provider to authentication services

Currently supported authentication services are:

- Google
- KeyRock

`poi_dp/authenticate_t.html` contains some hints for registering.

Notes:

- Google requires a name server entry for your server. A numeric IP address does not work.
- KeyRock does not support CORS and so does not reveal the identity of the user to the client program.

Register the POI data provider to the authentication services suitable for your purposes. The redirect callback is {your_poi_server}`/poi_dp/redirect_callback.html` , if needed. When you register, you get a client id to be used in authentication requests.

Configuring authentication client

1) Copy `poi_dp/authenticate_t.html` to `poi_dp/authenticate.html` .

```
$ sudo cp authenticate_t.html authenticate.html
```

2) Edit `poi_dp/authenticate.html` - update signin-client_id values for the authentication services. Search for the string "`*** REPLACE`" to find the right places, and read comments for some hints. E.g.:

```
$ sudo nano authenticate.html
```

Configuring the basic access rights

1) Copy `poi_dp/auth_conf_t.json` to `poi_dp/auth_conf.json` .

```
$ sudo cp auth_conf_t.json auth_conf.json
```

The template looks about the following:

```
{
  "description": [
    "These permissions override those in the database.",
    "..."
  ],
  "open_data": false,
  "hard_auths": {
    "google:john_doe@gmail.com": {
      "accounts": {
        "4d1a77c0-6cfb-4468-86fa-bff784012816": {"registration_time": 0}
      }
    },
    "fiware_lab:j_d": {
      "accounts": {
        "4d1a77c0-6cfb-4468-86fa-bff784012816": {"registration_time": 0}
      }
    }
  },
  "hard_users": {
    "4d1a77c0-6cfb-4468-86fa-bff784012816": {
      "name": "John Doe",
      "photo": "http://www.example.com/johndoe.jpg",
      "address": "Kotikatu 60 A 22, 90990 Oulu, Finland",
      "phone": "+356 8 999 999",
```

```
      "email": "john.doe@example.com",
      "additional_emails": [],
      "permissions": {
        "admin": false,
        "add": false,
        "update": false,
        "view": false
      },
      "identifications": {
        "google:john_doe@gmail.com": true,
        "fiware_lab:j_d": true
      }
    }
  }
}
```

The exemplary template represents one user account that can be logged in by both Google and Fiware Lab. UUIDs are used as the internal user Ids. Note that in this example the account has not been given any rights.

2) Edit `poi_dp/auth_conf.json`. E.g.:

```
$ sudo nano auth_conf.json
```

Replace the template data according to the following notes:

- `open_data` - Set `true`, if anybody can view the data
- `hard_auths` - These are authentications for the "root" users. These cannot be changed thru the API.
  - Keys of user authentications are of form <authentication provider>:<authentication_id> . E.g. `google:john_doe@gmail.com` . The authentication_id is the one used by the authentication provider.
  - **NOTE:** You can use only the authentication providers supported by the software.
  - `accounts` - These are user accounts that can be logged in using the authentication.
    - Keys of accounts are user Ids. These Ids are used to find user credentials and other user information under `hard_users`.
    - The content of an account is the Unix time stamp of the registration time. You may use `{"registration_time": 0}` .
- `hard_users` - These are accounts for the "root" users. These cannot be changed thru the API.
  - Keys of user accounts are user Ids. You may use e.g. https://www.uuidgenerator.net// for new Ids.
  - `name` must be unique.
  - `email` is used to send the invitation to register.
  - `permissions` specify, what the user can do. The boolean value `true` enables the permission.
    - `admin` - can manage users.
    - `add` - can add POIs.
    - `update` - can modify and delete POI data.
    - `view` - can view POI data.
  - `identifications` links back to `hard_auths`. This section must exactly have the authentication keys that have this account as a choice. The content of a key is `true`.
  - `photo`, `address`, `phone`, and `additional_emails` are for information only.

Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

End to End testing

You can do a quick test to see if everything is up and running by accessing the following URL:

```
http://hostname/poi_dp/radial_search?lat=1&lon=1&category=test_poi
```

For secure server use:

```
https://hostname/poi_dp/radial_search?lat=1&lon=1&category=test_poi
```

You should get a JSON structure representing a test POI as a response and possibly some general info about the site.

NOTE: *Authorization is not needeed in* `radial_search` *limited to category* `test_poi`*.*

List of Running Processes

You can use the following command to check if Apache HTTP server, PostgreSQL and MongoDB are running:

```
$ ps ax | grep 'postgres\|mongo\|apache2'
```

The output of the command should be something like the following:

Network interfaces Up & Open

The only required port open to the Internet is TCP port 80, used by HTTP protocol.

Databases

The POI Data Provider utilizes two database systems:

- PostgreSQL/PostGIS for storing and accessing data components with spatial data (fw_core data component)
- MongoDB for storing and accessing all other data components that do not require spatial searches

PostgreSQL

PostgreSQL has a database named 'poidatabase'. It contains a table called 'fw_core' and it contains the core information, such as name and location, about the POIs.

You can test if this table is succesfully created with the following commands:

```
$ psql -U gisuser poidatabase
poidatabase=> SELECT count(*) FROM fw_core;
```

If the table was created succesfully, this query should return '4', as there sould be four test POI entries created by the installation.

To exit PostgreSQL use:

```
poidatabase=> \q
```

MongoDB

MongoDB should also contain a database named 'poi_db'. It should contain a collection named 'testData' containing a single test POI entry, created by the installation.

You can test if MongoDB was succesfully configured with the following commands:

```
$ mongo
> use poi_db
> show collections
```

The show collections command should list five POI data component collections created by the installation: fw_contact, fw_marker, fw_media, and fw_time.

To exit MongoDB use:

```
> exit
```

Installing Demo Client

Demo Client is an optional feature.

The demo client allows you to immediately utilize your POI data provider. It shows POIs of selected categories on Google Maps background. It also allows you to add, modify, and delete individual POIs, if you have proper credentials.

NOTE: You need a Google Maps API key, because Google inc. requires you to obtain an API key for the application using Google Maps. You can obtain it from Get a Key/Authentication.

Copy the client:

```
$ sudo cp -r poi_mapper_client /var/www/html/pois
```

Edit the `pois/index.html` replacing the string "YOUR_GOOGLE_API_KEY" with your actual API key.

```
$sudo nano /var/www/html/pois/index.html
```

E.g.: from

```
<script type="text/javascript"
    src="https://maps.googleapis.com/maps/api/js?v=3.exp&key=YOUR_GOOGLE_API_KEY"\>
</script>
```

to

```
<script type="text/javascript"
    src="https://maps.googleapis.com/maps/api/js?v=3.exp&key=MIzaSyBk59fRpyN4-PGl4UwfIfQ3sjxQwRm
        3pjl"\>
</script>
```

Now, the POI browser can be accessed using a web browser at *{your_poi_server}*/pois . The POIs can be added, edited, and deleted at *{your_poi_server}*/pois/edit_poi.html .

Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

Resource availability

The amount of available resources depends on the size of the database and the usage rate of the service. The minimum recommended available resources are:

- Available memory: 4 GB
- Available disk space: 40 GB

Resource consumption

The load value reported e.g. by the 'top' utility should not exceed the number of CPU cores in the system. If this happens, the performance of the system can dramatically drop.

Remote Service Access

Check that the HTTP port (80) [HTTPS port (443) in a secure server] is open and accessible from all the networks from which POI-DP will be used.

I/O flows

All the incoming and outgoing data of the POI Data Provider will go through TCP port 80 [SSL port 443 in a secure server]. The size of the flow is entirely dependant on the usage of the service, e.g. number of users.

Updating Database to R5.1

The language key for the non-language-specific strings is changed from "" to "___" (two underscore characters. The reason is that the MongoDB database does not like zero-length keys. The keys in databases that are created with earlier versions of the POI-DP can be updated using the following command:

```
1 $ psql -U gisuser -d poidatabase -a -f FIWARE-POIDataProvider/install_scripts/update_fw_core_int
      l_to_5.1.sql
```