

FIWARE Points of Interest (POI) Data Provider R5.4

Why and How



Center for Internet Excellence

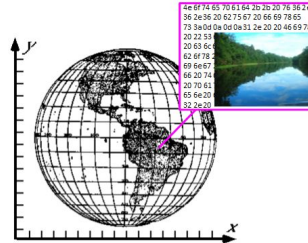
Nonprofit educational material. Fair use of copyrighted content, if any, is assumed.

This is an introduction to FI-WARE points of interest data provider.

Points of Interest - POI

A POI system

- Relates **information** to **places**
- Supports searches by location and other criteria, e.g. time



A points of interest, or POI, system relates information to places. The information related to a place can be whatever you need for your application:

- restaurants and scenic places for tourists,
- maintenance instructions and logs related to a location for maintenance business,
- photos from your trips,
- actually any information you want to find by the location.

Contents of this Presentation

This presentation has two parts

1. Where a POI system might be useful
2. Developing a POI system using FI-WARE POI GE

This presentation consists of two parts. The first part gives some ideas about utilisation of POI systems, The second, more technical, part introduces you to developing an application on top of the FI-WARE POI generic enabler.

1. Where a POI System Might be Useful

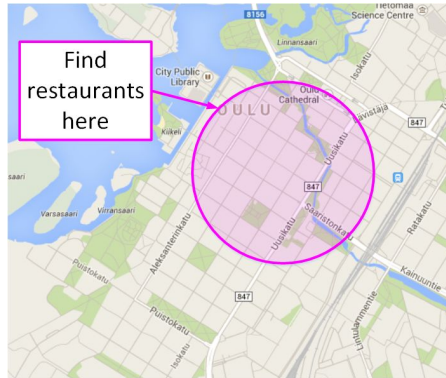
The following slides illustrate possible uses of POI data

- What does a POI data provider do
- Why to use FI-WARE POI Data Provider
- Using POI information – Map
- Using POI information – Augmented Reality
- Using POI information – Your own needs

This first part illustrates the area of application of POI technology in general. It also presents special benefits of FI-WARE POI technology. Examples of possible uses shown here are augmented reality and a map application.

What does a POI Data Provider do

- A POI data provider stores and provides **information based on location**
- Searches based on area e.g. circle or bounding box.
 - Filtering of results using other data values e.g. category or tags



A POI system is used to store and retrieve data by location. Data retrieval arguments may include different kinds of area descriptions and other conditions. A useful example may be finding restaurants near to you, when you are visiting a new town.

Why to use **FI-WARE** POI Data Provider

FI-WARE POI Data Provider uses **modular** and **distributed** data. This allow you to

- **Combine** your own data with public POI data
- **Speed up** mobile operation by fetching only the data your application needs
- Use your own data structures, if need for **extra flexibility**

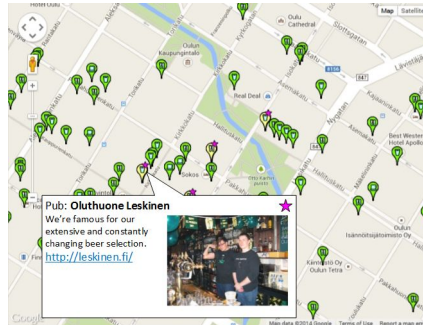
There are several other POI data and API definitions proposed and in use. However, it seems that there is room for one more with special features. The FI-WARE POI definitions are developed to avoid complexity and bloated data of other POI systems.

Due to modular structure of the FI-WARE POI system the amount of data transferred from the server is kept minimal, which is significant benefit in mobile use.

Using FI-WARE POI system the architecture allows you to create your - public or proprietary - system on top of existing - public or proprietary - POI service. Your system may enhance existing POIs with your data and add your special POIs to existing selection of POIs. Of course, your specialities are visible only to the users of your system.

Using POI Information - Map

- Show place **information** of selected categories, like restaurant and other criterias, like open times **on the map**.



POIs on the map is a very common way to present location dependent information. There are several systems using this approach. Google Maps and OpenStreetMap are familiar examples of this approach.

Using POI Information – Augmented Reality

- **Augment** the view **with** place **information** of your interest



A POI system can be used to supply the location dependent information to an augmented-reality application. The application obtains the relevant information from the server by e.g. location and category and shows it on top of the real view. Nokia/Here City Lens is an example of an application that helps to find interesting places in a strange town.

Using POI Information – Your own needs

- Attach the **information you need** to the locations
- Use or show the information **as you need**
- Search information based on the location, and the extra data you provided



In general, if the information needs to be found by location, a POI database is a good candidate for the back-end system. Specifically, FI-WARE POI back end is designed to be easily adaptable to your information needs. For a special purpose you may need a special purpose web client. One thing you have to implement in your special purpose client is the POI API. A key benefit of the FI-WARE POI API is that you can implement it with a straightforward and compact piece of code.

2. Developing a POI system using FI-WARE POI GE

Content

- Development prerequisites
- Modular and distributed data
- Components in JSON format
- Multilingual data items
- Default language in multilingual data
- Spatial searches
- Additional data retrieval
- Creating a new POI
- Updating POI data
- Deleting a POI

This second part of the presentation introduces you to using FI-WARE POI GE in your web client. The presentation starts from development prerequisites and continues through technical principles that are behind the special capabilities of the FI-WARE POI architecture. These include modular and distributed data, components, and multilingual data needed for internationally available systems. Finally the details of client side code of POI search, update, and delete operations are covered. This should give you the necessary information and pointers to utilize a FI-WARE POI Generic Enabler.

Development prerequisites

- Basic **JavaScript** knowledge
 - E.g. from here: <http://www.w3schools.com/js/>
- To show maps you may use e.g.
 - **Google Maps** API <https://developers.google.com/maps/>
 - **OpenStreetMap** http://wiki.openstreetmap.org/wiki/OpenLayers_Simple_Example
- Detailed instructions
 - http://fiware-poidatapvider.readthedocs.org/en/latest/POI_Data_Provider_User_and_Programmers_Guide/ - later referred **The Manual**

In order to write FI-WARE POI API to your web client using the method presented here you need to know some JavaScript. If you want to show the POI information on a map, you need to find some map provider and its API documentation. Google Maps and OpenStreetMap are good, public generic purpose choices. Detailed step-by-step instructions and code examples are found in the User_and_Programmers_Guide later referred as The Manual.

- Links:

JavaScript <http://www.w3schools.com/js/>

Google Maps <https://developers.google.com/maps/>

OpenStreetMap http://wiki.openstreetmap.org/wiki/OpenLayers_Simple_Example

FI-WARE POI GE - The Manual

http://fiware-poidatapvider.readthedocs.org/en/latest/POI_Data_Provider__User_and_Programmers_Guide/

Modular and Distributed Data

- Modular data to **avoid unnecessary data transmission**, by requesting only the data components you want
- Data of a POI consists of components
 - E.g. fw_core, fw_time, fw_contact, fw_media, ...
- HTTP Request defines wanted data components
- Different **data components** of the same POI **may reside in different servers**.
- Application specific **proprietary data components** can be used

The data of a POI consists of modules for different purposes. This is

- * to avoid coding, storing, and transmitting unnecessary data,
- * to make introducing new data for new needs easy and controlled,
- * to enable distributed storage for POIs and their data items
- * to avoid bloating of application code

The data for a POI consists of components. Examples of components are:

- * fw_core - core information needed in most POIs: name, location, category,...
- * fw_time - times of availability. Can be used e.g. to tell, when a shop, tourist attraction etc. is open for customers.
- * fw_contact - address, phone number, email, etc. to contact a service related to the POI
- * fw_media - pictures, videos, voice recordings, etc. related to the POI

Modularity of POI data is used so that the components, that the client needs, are listed in the HTTP request used to query the data.

Different POIs, specially their fw_core components, of the system may reside in different servers. Moreover, the different components of the same POI may reside in different servers. In these cases the client send queries to all relevant servers. This is easy, because the components are identified by the UUID of the POI.

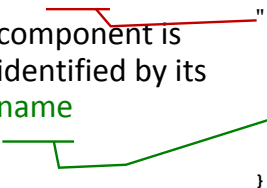
If the application needs data that is not suitably contained in already defined components, it is fine to introduce new components to meet the need. Naming rules of the components try to prevent naming conflicts.

Components in JSON Format

- POI is identified by its **UUID**

- component is identified by its **name**

```
"8e57d2e6-f98f-4404-b075-112049e72346": {  
  "fw_core": {  
    <location, identity & other core information>  
  },  
  "fw_time": {  
    <availability times>  
  },  
  <other requested components>  
}
```



This is the upper structure of a POI in JSON format. The UUID of the POI is used as the key of the data object. The POI data object consists of properties having component names as their keys.

Multilingual Data Items

- Most texts and web links (URLs) are for several languages

```
{  
  "": "<default or language independent data>",  
  "<lang_ID1>": "<data in language lang_ID1>",  
  "<lang_ID2>": "<data in language: lang_ID2>",  
  ...  
}
```

- *ISO 639-1 language code* identify the language
- Example:

```
"label": {  
  "": "Uniresta Lobby Restaurant at University of Oulu",  
  "fi": "Unirestan Aularavintola Oulun yliopistolla.",  
  "en": "Uniresta Lobby Restaurant at University of Oulu",  
  "es": "Restaurante Aularavintola"  
}
```

Multilingual data items are objects that contain properties for different languages. ISO 639-1 language codes are used as keys to the properties. Default or language independent property is identified as two underscores as its key. Examples of multilingual items are texts and URLs.

Default language in Multilingual Data

- **No need to duplicate** text for the default
- A specific language can also be the default

```
{  
  "_def": "<lang_IDn>",  
  ...  
  "<lang_IDn>": "<default data in language: lang_IDn>",  
  ...  
}
```

- Example (previous example without duplicate):

```
"label": {  
  "_def": "en",  
  "fi": "Unirestan Aularavintola Oulun yliopistolla.",  
  "en": "Uniresta Lobby Restaurant at University of Oulu",  
  "es": "Restaurante Aularavintola"  
},
```

If a specific language is also the default language, duplication of data is needed, if no special trick is available. The special trick is a possibility to define one of the languages as the default. The property with special key "_def" defines the default language. The content of the property is the language code of the default language. This helps to reduce amount of data and to avoid inconsistencies that often result from updates of duplicate data.

Spatial Searches

- Example of **radial search** – GET request
 - http://<your_poi_server>/radial_search?lat=65.01255&lon=25.47133&radius=250&category=cafe,restaurant&auth_t=<a_token>
 - gives restaurants and cafes in the center of Oulu Finland
- Code snippets in JavaScript to send the request and to receive the results can be found in *The Manual*
- Available categories are server dependent – values of OpenStreetMap Amenity-key are used in demo servers
- See *The Manual* for **bounding box** search

A typical interaction with a POI server begins with spatial search: "what do we have here". For spatial searches FI-WARE POI GE provides radial searches and bounding box searches. In radial search the client supplies coordinates of the center point in decimal degrees and the radius of the area in decimal meters. Limiting conditions e.g. accepted categories and languages can be used too. Available categories are system/server dependent. See possible server related documentation for categories. Syntax and other details of the spatial search REST query are defined in *The Manual*. Default spatial search results the core data components, only.

Additional Data Retrieval

- Example of extra information retrieval – GET request
 - http://<your_poi_server>/get_pois?poi_id=30ddf703-59f5-4448-8918-0f625a7e1122&component=fw_media&auth_t=<a_token>
 - gives media links associated to the POI
- Code snippets in JavaScript to send the request and to receive the results can be found in *The Manual*

When the client has sifted through the results of the spatial search, it may need some more information about some interesting POIs: "tell me more about these". Additional data components are queried using the UUIDs of the POIs and the names of interesting components. Again, see The Manual for exact syntax and examples.

Creating a New POI

- Example of adding a new POI – **POST** request
 - http://<your_poi_server>/add_poi?auth_t=<a_token>
 - data content is components of the POI in JSON format
 - response is the UUID created for the POI
- Code snippets in JavaScript to send the request and to receive the results can be found in *The Manual*

A new POI is introduced to the POI server using add_poi request. This is a POST request that delivers the POI description in JSON format. The description contains at least the fw_core component and possible other components depending on the purpose and implementation. Response from the server contains the UUID of the newly created POI. See The Manual for details.

Updating POI Data

1. Retrieve POI data **for update** – **GET** request
 - http://<your_poi_server>/get_pois?poi_id=30ddf703-59f5-4448-8918-0f625a7e1122&get_for_update=true&auth_t=<a_token>
 - This brings all language versions and **last_update** info
 2. Modify the data as needed – retain the **last_update** info unchanged
 3. Send updated data – **POST** request
 - http://<your_poi_server>/update_poi&auth_t=<a_token>
 - data content is the modified data in JSON format
 - response tells success or not
 - update fails if a conflict happens
- Code snippets in JavaScript to send the request and to receive the results can be found in *The Manual*

Updating the POI data is a little more complex exercise. First you have to retrieve the POI data for update. Then the updated data is sent back to the server. Finally, the server reports the success or failure of the update.

Because of the distributed nature of the system, there might occur update conflicts: two clients get the same POI for update and then send independently and differently updated data back. Depending on the purpose of the system this may lead to undesired or worse consequences, if not acted properly upon.

The FI-WARE POI GE uses update stamps to catch conflicts. If an update conflict occurs the conflicting - the latter - request is rejected, and the client is notified accordingly. In order to complete the rejected update the client must fetch the POI again to get the new data for a new update attempt. See *The Manual* for the details.

Deleting a POI

- Example of deleting a POI – DELETE request
 - http://<your_poi_server>/delete_poi?poi_id=30ddf703-59f5-4448-8918-0f625a7e1122&auth_t=<a_token>
- Code snippets in JavaScript to send the request and to receive the results can be found in *The Manual*

Deleting a POI is easy. Send a DELETE request with the UUID of the POI. See The Manual for the details.

The End

by Ari Okkonen, Center of Internet Excellence, University of Oulu

This completes the lesson. Good luck for your project!

A presentation by Ari Okkonen, Center of Internet Excellence, University of Oulu