Generic Configurable “Dynamic POI-DP” Specification

# Introduction

This document specifies the FIWARE POI\_DP (Point of Interest Data Provider) Generic Enabler additional functionality provided by a new feature called Dynamic POI’s. Dynamic POI’s are POI’s with additional data elements describing how to retrieve POI specific external information together with data structures to store that information in the POI DP server. The main use case for Dynamic POI’s is for example describing, storing and viewing the values of sensor data elements, like temperature, wind, traffic, moisture, air pressure etc. together with the other POI information.

# Specification

This chapter is divided into two sections, the first one describes the additional internal POI data elements required by the Dynamic POI system, and the second one the configuration file needed to specify where to retrieve the external information and how to map the response data to the internal data structures used to store the data.

## Additional internal data elements

The Dynamic POI system requires an additional data element called “fw\_dynamic” to be included in the POI data:

“fw\_dynamic”: {

“host\_type”: “<name of the host connection type as described in the configuration file>”

“host\_id”: [“<optional ID(s) used to identify a specific sensor by the above host>”]

“data\_type”: “<name of the data type as described in the configuration file>”

“data\_params”: “<optional additional data parameters>”

“last\_update”: { …} /\* for internal use only \*/

}

Depending on the contents of the configuration file additional data elements will be generated automatically to the internal data elements of a specific POI. These would typically include e.g. the “fw\_sensor” structure, but the depending on the configuration file any other fields can also be created/updated.

“fw\_sensor”: {

“data”: [ {

“name”: {

“de”: “<sensor name in german>”

“en”: “<in english>”

“es”: “<in spanish>”

“fi”: “<in finnish etc.>”

},

“label”: {

“de”: “<sensor printable label in german>”

“en”: “<in english>”

“es”: “<in spanish>”

“fi”: “<in finnish etc.>”

},

“type”: “<sensor type, e.g. temp, humidity, wind etc.>”

“value”: “<sensor current/last updated value>”

“unit”: “<unit of above value>”

} ],

valid\_duration:”<valid value cached time in seconds>”

last\_update: {

“timestamp”: “<Unix timestamp of last value update>”

“responsible”: /\* Reserved for future use \*/

}

}

## Configuration file

The configuration file is located in the poi\_dp server in the same directory as the poi\_dp itself and named poi\_dp\_dyn\_conf.json. The contents of this file defines where the external dynamic POI data elements should be fetched and how the response content should be mapped to POI additional data structures, like the “fw\_sensor” described above. The same file can contain several host and data mapping configurations. If you modify this file please make sure to use valid JSon format, or use the JSONLint to validate the syntax before saving the file. Content of the file is described below (note that comments are not allowed in the actual JSon file):

{

"**host\_type**": /\* This section describes the external host connection types \*/

{

"<name of the host connection>":

{

"method": "REST\_GET", /\*Only REST GET method supported so far \*/

"params":

{

"url": "<http url of the host connection>",

"params": "<additional url parameters>",

"headers": {"<header>":"<value>","<header>":"<value>",…}

}

}

},

"**data\_mapping**": /\* Data mapping portion begins here \*/

{

"<data\_type>": /\* Name of the data type as referred in “fw\_dynamic” \*/

{

"fw\_sensor": /\* Name of the additional data structure to be included, typically “fw\_sensor” \*/

{

"data": /\* Name of the data structure below “fw\_sensor”, typically “data” \*/

[

{

"name": {"<Sensor names in different languages as described above for “fw\_sensor” >"},

"type": "<type of the sensor, e.g. temp, humidity, wind, etc.>",

"value": ["<mapping function, see below>", “<mapping function parameters>”],

"unit": “<unit or mapping function reference as above for unit>”

}

]

}

}

}

}

Currently two mapping functions are supported, “\_fw\_json” and “\_fw\_match”. The first one can be used to extract values of Json responses, the second from text/html responses. Please see also examples in Sec 3.

["**\_fw\_json**", {"<json string including “**?**” mark in place of the searched/returned value if host response is Json>”}]

[“**\_fw\_match**”, ”<search string including **?**-mark in place of searched/returned value if response is html>”]

# Orion CB integration demo

A short simple demonstration of how to use Dynamic POI’s is provided using the FILAB Orion Context Broker with Santander city sensor data (sound meters and traffic monitors) as an example case:

* Configuration file: <http://dev.cie.fi/FI-WARE/poi_dp_dyn/poi_dp_dyn_conf.json>
* Demo: <http://dev.cie.fi/FI-WARE/demos/dyn_poi/edit_poi.html>

Open the above demo page with Chrome or Firefox browser and enter “Santander” as position and click the “Go to position” button. Select category as “Sensor” in the left window pane and click visible POIs to view the sensor values and/or right click to edit the POI data. A specific JSon POI data editor will open in case you choose to modify the POI data. See figure 1 below.

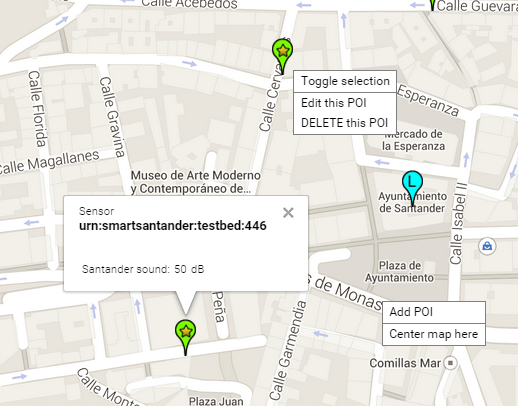


Figure 1. Santander demo case

Example of a dynamic POI data provider configuration file for Orion CB Santander content together with one html/text based Willab weather sensor (Oulu/Linnanmaa) configuration:

{

"host\_type":

{

"filab\_orion\_1026":

{

"method": "REST\_GET",

"params":

{

"url": "http://orion.lab.fi-ware.org:1026/ngsi10/contextEntities/",

"params": "?attributeFormat=object",

"headers": {"Content-Type":"application/json","Accept":"application/json","X-Auth-Token":"SDkJJ…..gfSDkJJBDQ"}

}

},

"willab":

{

"method": "REST\_GET",

"params":

{

"url": "http://weather.willab.fi/weather.html.fi",

"params": "",

"headers": {"Content-Type":"text/html","Accept":"text/html"}

}

}

},

"data\_mapping":

{

"orion\_sound\_sensor":

{

"fw\_sensor":

{

"data":

[

{

"name": {"en":"Santander sound"},

"type": "noise",

"value": ["\_fw\_json", {"contextElement":{"type":"santander:sound","attributes":{"sound":{"value":"?"}}}}],

"unit": ["\_fw\_json", {"contextElement":{"attributes":{"sound":{"metadatas":[{"value":"?"}]}}}}]

}

]

}

},

"orion\_traffic\_sensor":

{

"fw\_sensor":

{

"data":

[

{

"name": {"en":"Santander traffic"},

"type": "occupation",

"value":["\_fw\_json", {"contextElement":{"type":"santander:traffic","attributes":{"occupancy":{"value":"?"}}}}],

"unit": ["\_fw\_json", {"contextElement":{"attributes":{"occupancy":{"metadatas":[{"value":"?"}]}}}}]

}

]

}

},

"willab\_status":

{

"fw\_sensor":

{

"data":

[

{

"name": {"en":"Linnanmaa temperature"},

"type": "temp",

"value": ["\_fw\_match", "<p class=\"tempnow\">? &deg"],

"unit": "C"

},

{

"name": {"en":"Linnanmaa wind"},

"type": "wind",

"value": ["\_fw\_match", "<tr><th>Tuulen nopeus:</th><td>? m/s"],

"unit": "m/s"

},

{

"name": {"en":"Linnanmaa wind direction"},

"type": "winddir",

"value": ["\_fw\_match", "<tr><th>Tuulen suunta:</th><td>?&deg"],

"unit": "&deg;"

},

{

"name": {"en":"Linnanmaa humidity"},

"type": "humidity",

"value": ["\_fw\_match", "<tr><th>Ilmankosteus:</th><td>? %"],

"unit": "%"

},

{

"name": {"en":"Linnanmaa air pressure"},

"type": "pressure",

"value": ["\_fw\_match", "<tr><th>Ilmanpaine:</th><td>? hPa"],

"unit": "hPa"

}

]

}

}

}

}

# Adding new Dynamic POIs

To add a new Dynamic POI you need to:

1. Create the new POI either by using the add\_poi API or by using the above demo application “Add POI” menu option.
2. Fill in the required fields as shown on the right: Category, POI name and optional description plus the fw\_dynamic configuration fields: host\_type, host\_id and data\_type. In this case there is only one sensor connected to this host so no host\_id will be given.
3. The above fw\_dynamic fields refer to the configuration file respective fields. If this is a new host\_type or a new data\_type a new entry to the configuration file needs to be added, like in this case the “willab” host and “willab\_status” data types.
4. Analyse the response given by the host, JSon should be preferred if available and \_fw\_json function to be used to parse the return value from the response text, a question mark denotes the place of the expected return value for the given field. Likewise a text matching function \_fw\_match is available for text/html responses, like in this willab case. Similarly a question mark shows the location of the required return value to be fetched to the poi\_dp internal data structures. See the example configuration file above.
5. Before saving the configuration file back to the server it is wise to check the JSon syntax e.g. by using JSONLint.

