How to use X-GSN

Table of Contents

[Configuring GSN for Integration with LSM 3](#__RefHeading__439_835426417)

[HTTP Listener Wrapper 3](#__RefHeading__6686_1129275050)

[X-GSN Port 3](#__RefHeading__441_835426417)

[How to Start /Stop X-GSN 3](#__RefHeading__443_835426417)

[Sensor Registration 4](#__RefHeading__445_835426417)

# **Configuring GSN for Integration with LSM**

In order to connect GSN with LSM, you need to edit the file **x-gsn/conf/lsm\_config.properties**. You need to specify user credentials and LSM schema to use as follows:

Figure 1: lsm\_config.properties

# HTTP Listener Wrapper

For IoE-enabled Meeting Management System, the HTTP Listener wrapper is created and used.

The HTTP Listener wrapper is stored in **modules/xgsn/src/main/java/org/openiot/gsn/wrappers/REST**

The HTTP Listener wrapper uses Restlet API and handles receiving raw sensor data from physical / virtual devices and publishing data to LSM. For the HTTP Listener wrapper, a single sensor observation value such as light, noise and proximity is dealt with data type *double,* whereas multiple sensor observation values are dealt with data type *string*. For example, a location sensor observes 2 values, latitude and longitude. They consider as one string value by concatenating, e.g., 53.2719\_-9.0489.

To create or use other wrappers, please see details in <https://github.com/OpenIotOrg/openiot/wiki/X-GSN-Use> or <https://github.com/OpenIotOrg/openiot/wiki/X-GSN-Develop>

# **X-GSN Port**

* HTTP Listener Port

A HTTP server is running on HTTP Listener wrapper and the default port is 8182 and it is set in **x-gsn/conf/httplistener\_config.properties**.

* X-GSN Web Server Port

We can register sensors to LSM through a REST service by using X-GSN web server. The default port is 22001 and the port is set in **x-gsn/conf/gsn.xml**.

See more details in <https://github.com/OpenIotOrg/openiot/wiki/X-GSN-Use>

# **How to Start /Stop X-GSN**

X-GSN can be started by running *gsn-start.sh (bat)* and stopped by running *gsn-stop.sh (bat)*.

# **Sensor Registration**

In order to register sensors to OpenIoT, it is necessary to create a sensor metadata file and a sensor configuration XML file. This section focuses on sensor registration to use HTTP Listener wrapper.

## ***Creating Sensor Metadata File***

The metadata file contains metadata about the sensors. Figure 2 and 3 show an example of a light (single sensor value) and location sensor (multiple sensor values) metadata files for HTTP Listener wrapper.

Figure 2: Light Sensor Metadata file

Figure 3: Location Sensor Metadata file

Table1 : Description of Parameters

|  |  |
| --- | --- |
| Field type | Description |
| sensorName | Define name of sensor, it should be a unique name. |
| metaGraph | Define LSM schema to use specified in lsm\_config.properties. |
| dataGraph | Define LSM schema to use specified in lsm\_config.properties. |
| fields | Define observation types. |
| field.FIELD\_NAME.  propertyName | Define URI of the FIELD\_NAME property. |
| field.FIELD\_NAME.unit | Define observation unit. Unit can be defined for **only single value**. If multiple values, unit must be empty as shown in Figure 3. |
| latitude/longitude | Define a sensor location coordinates. |
| sensorID | Initially, it should be empty. When registering a sensor, a unique sensorID will be provided by the LSM-Light, then replace it. |

## ***Creating sensor configuration XML file***

A wrapper class and a processing class have to be specified in XML configuration file. Figure 4 shows an example of a light sensor configuration XML file for HTTP Listener wrapper. Sample configuration XML files are stored in

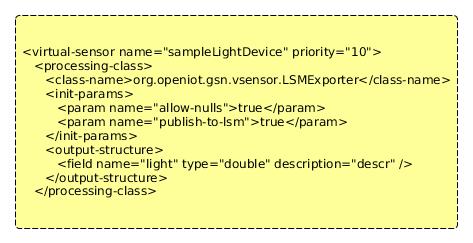
*x-gsn/virtual-sensors/LSM/sampleSensorsForHTTPListenerWrapper/*



* + 1. Figure 4: Light Sensor Configuration XML file

### Specify processing class

Figure 5 and 6 show specifying processing class.

Figure 5: Specifying Processing Class in Light Sensor Configuration XML file

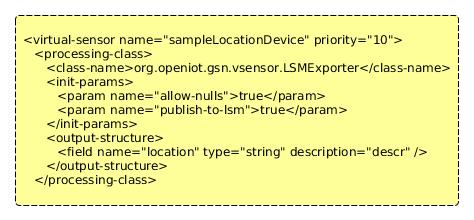


Figure 6: Specifying Processing Class in location Sensor Configuration XML file

Table 2: Description of Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| <virtual-sensor name> | Name of the sensor which is defined in the metadata file. |
| <class-name> | Define a processing class. A processing class in X-GSN handles processing data between the wrapper and the data publishing engine. **LSMExporter** processing class is used in HTTP Listener wrapper. LSMExporter processing class publishes data to LSM. |
| <field name, type> | Define field name as field which is defined in the metadata file.  Define data types. For HTTP Listener wrapper, two type of data types are used:   * **double** for single observation value such as light, noise and proximity * **string** for multiple observation values such as location (latitude and longitude) |

### Specify wrapper class

Figure 7 and 8 show specifying wrapper class.

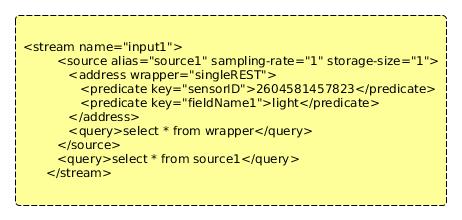
Figure 7: Specifying Wrapper Class in Light Sensor Configuration XML file

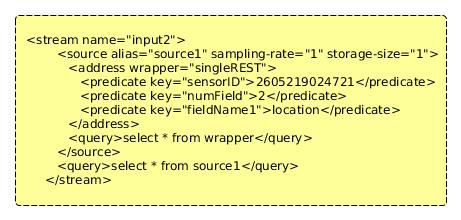
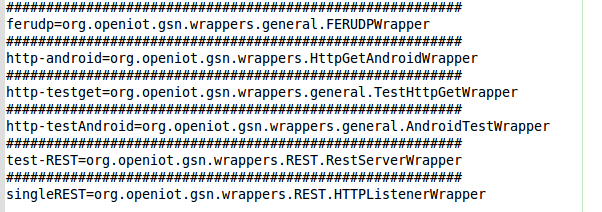
Figure 8: Specifying Wrapper Class in Location Sensor Configuration XML file

Table 3: Description of Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| <stream name> | Define **Input1** for single sensor value.  Define **Input2** for multiple sensor values. |
| <address wrapper> | Name of a wrapper defined in wrappers.properties.  Define **singleREST** for HTTP Listener wrapper. |
| <predicate key="sensorID"> | Define the sensor ID provided by LSM, only digit parts. |
| <predicate key="numField"> | Define the number of fields(this is a number of sensor observation values), ***required if more than one field.*** |
| <predicate key="fieldName*1*"> | Define the field name. |

**Note that:**

A wrapper class has to be defined in wrappers.properties (x-gsn/conf)



If you create a new wrapper, you need to specify the wrapper class in wrappers.properties and define this wrapper in a XML configuration file.

## ***How to register sensors***

The IoE mobile client application registers sensors and inserts additional triples for our extended information model into LSM automatically, therefore it is unnecessary to register sensors manually.

However, sensors can be registered manually to LSM as follows :

* Script (lsm-register.sh/bat)
* HTTP post request via a REST service

See more details in <https://github.com/OpenIotOrg/openiot/wiki/X-GSN-Use>

When you register sensors manually, it is necessary to add additional triples to use our extended information model.

The following triples have to be inserted once for the first time as shown in Figure 9.

Figure 9: Inserting additional triples

Then, the following triples are necessary for each sensor as shown in Figure 10.

These additional triples contain the relation between a sensor and a device (ssn:platform) and relations between a device and device owner (ddo:owns). The device/platform is associated with the OpenMeetings user (ownership) which is associated with contact information of the user.



Figure 10: Inserting additional triples for each sensor

LSM-light-server can be accessed via LSM-light.client and the additional triples can be pushed to theLSM triple store directly from a client. LSMTripleStore class in the lsm-light.client module supports handling sensor data to LSM.

See more details in <https://github.com/OpenIotOrg/openiot/wiki/Sensor-Data-Develop>

Table 4: Description of additional triples

|  |  |
| --- | --- |
| IDs | Description |
| SensorID | A sensor ID which is provided by LSM when registering a sensor. |
| DeviceID | A unique device ID. If four sensors are registered to the OpenIoT, the four sensors are associated with the same device ID. |
| UserID | OpenMeetings user ID. |
| ContactUserID | A unique ID for personal information model to associate the user with the contact information of the user.  A unique ID for personal information model to integrate/link sensor data with the contact information of the user. This ID is different from userID. |

How to register sensors with the script:

* Create a metadata file and store under x-gsn/virtual-sensors
* Run lsm-register.sh/bat script; ./lsm-register.sh <metadata\_filename>
* Replace a sensor ID which is provided by LSM into the metadata file
* Create XML file and then store under x-gsn/virtual-sensors
* X-GSN loads up XML files from x-gsn/virtual-sensors