Wirecloud-IoT Application Tutorial

This document provides a description on how to develop an application integrating the IoT GEs (Configuration Manager and Broker) with the Wirecloud platform. In general, the application requests data about a determined device to the IoT GEs which in turn reply with the data to be presented to the user in the form of a widget.

Architecture

The architecture of the sample application uses a modular design and is based on the Wirecloud and IoT architectures. Figure 1 shows a general overview of the modules used and the types of messages exchanged by them.

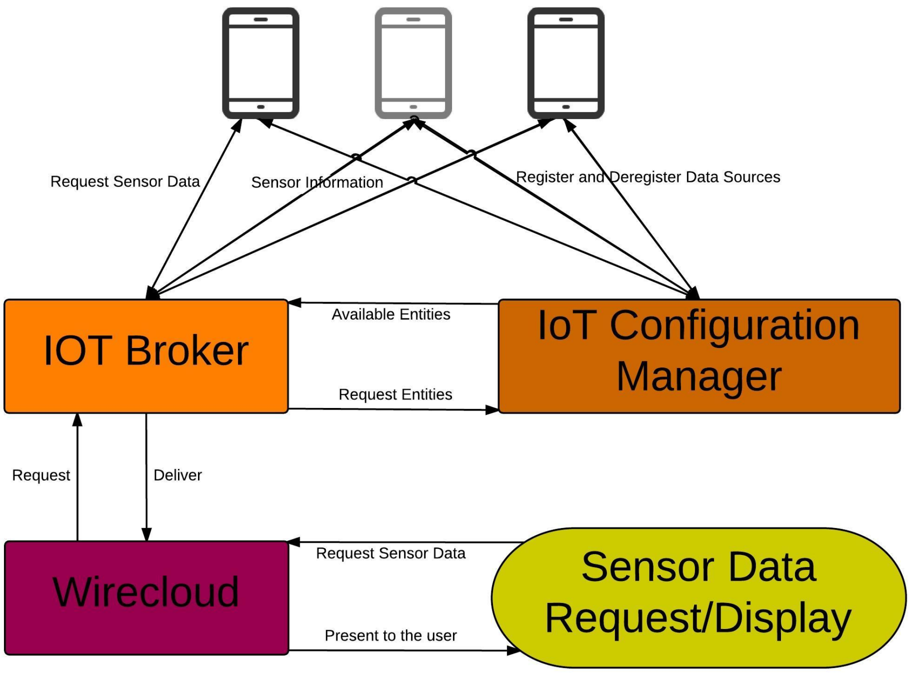


Figure 1: General Architecture of the Sample Application

The architecture is comprised of the following modules:

- **Wirecloud:** this module represents the user interface with the system. It provides several widgets for interaction, which are connected through the wireframe mechanism.
- **IoT Configuration Manager:** this module handles the registration, deregistration and availability of the devices that provide data. It communicates with the IoT Broker through the NGSI9 standard.
- **IoT Broker:** this module's purpose is to mediate the communication between the Wirecloud interface and the IoT devices. IT communicates with the Wirecloud interface through the NGSI10 standard and as mentioned above, with the IoT Configuration manager through the NGSI9 standard.

At the start of the application, the client browser connects to the Wirecloud interface (Figure 2). After logging in the user is presented with two widgets.

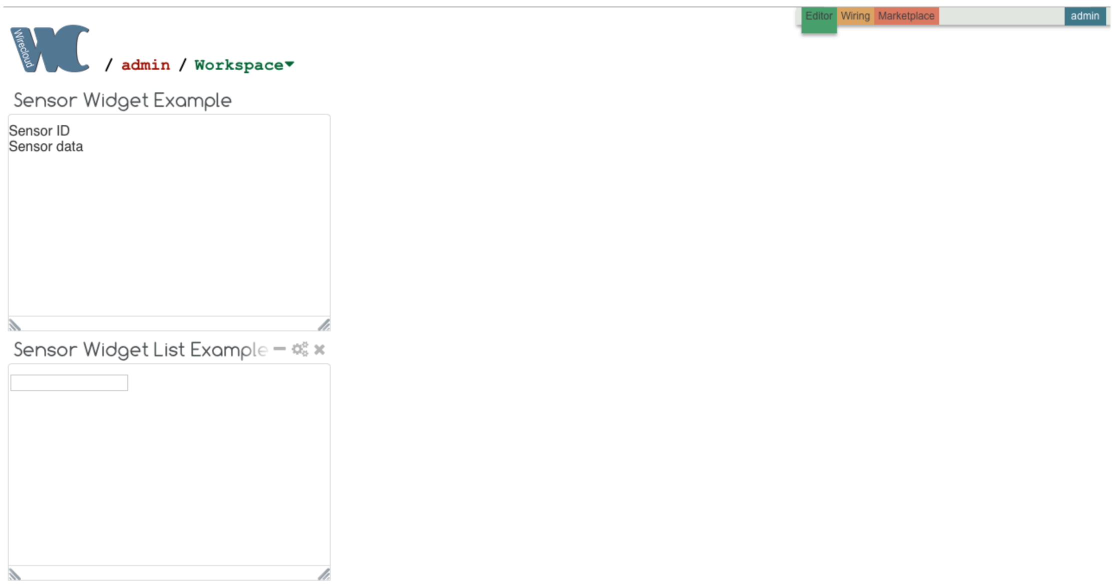


Figure 2: Wirecloud Interface

The first widget (Sensor Widget List Example) is an input widget where the user can enter the name of the device he wishes to get data from. The second widget (Sensor Widget Example) is the widget where the data will be requested, received and presented. The two widgets are connected through the wireframe architecture as shown in Figure 3, with the first widget providing the second with the name of the device to get the respective data.

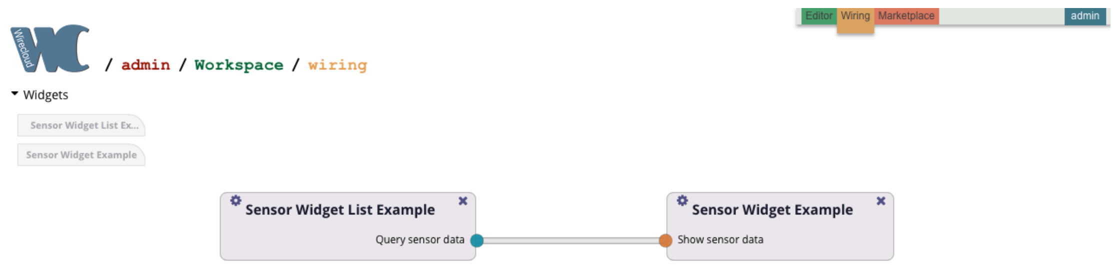


Figure 3: Widgets wiring

Generated Documentation

This application provides an example of how to integrate the IoT GEs with the Wirecloud architecture. The main goal is to use the communication architecture of the IoT GEs to retrieve data from devices and present in to the user in the form of a widget. Given that the IoT GEs were designed to receive and reply to enquiries regarding the registered devices, it is possible to develop an application that allows the visualization of information which originated on another part of the system, specifically in the IoT devices.

This is a basic application that for the sake of simplicity shows that it is possible to link two widgets which communicate with the IoT GEs to retrieve data and present it.

The installation and configuration instructions detailed in this section are for a deployment in local machines in your own premises and could easily be adapted to deployment in the FIWARE Cloud.

Install IoT Configuration Manager

To install and start the IoT Configuration Manager follow the installation procedures described in the Configuration Manager - IoT Discovery - Installation and Administration Guide. After the successful installation and configuration, the IoT configuration manager should be running on Apache. The REST interface will be receiving information in the URL: http://localhost:8080/ngsi9/.

Install IoT Broker

To install and start the IoT Broker follow the installation procedures described in the IoT Broker - Installation and Administration Guide. After the installation of the IoT Broker you can start the service by running the scripts in the main folder of the GE. The module should be accessible in the URL: http://localhost:8090/ngsi10/.

Install Wirecloud

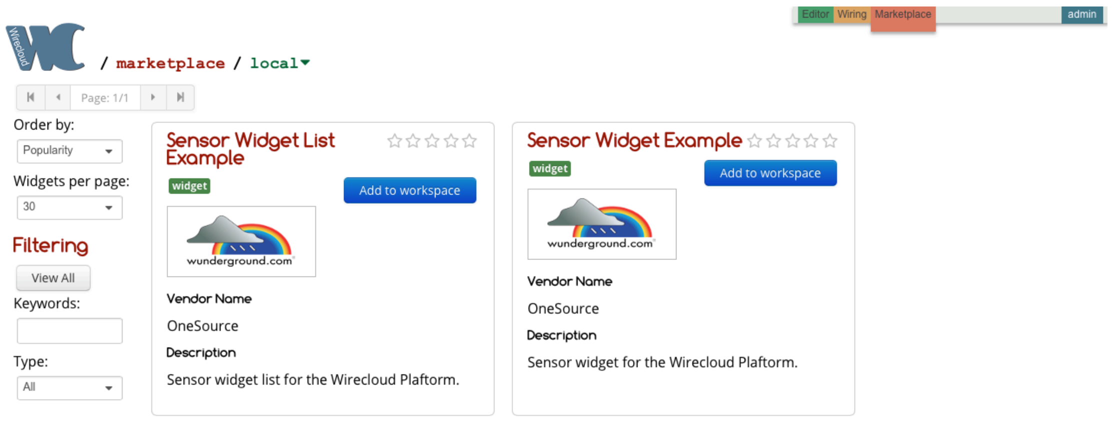
To install and start the IoT Broker follow the installation procedures described in the Application Mashup - Wirecloud - Installation and Administration Guide. After the installation of the IoT Broker you can start the service by running the scripts in the main folder of the GE. The Wirecloud module should be accessible in the URL: http://localhost.

To install the widgets you can download them from the repository by using the following command:

```
git clone https://github.com/PedroBorges9/Fiware-Wirecloud-IoT-DeviceData/
```

Next, go to each of the folders and compress its contents into a file with a .wgt extension. This can be easily done by zipping the files and by changing the files' extension to .wgt. After this is done for each of the folders you are ready to upload the widgets to the Wirecloud platform. To do so, enter the Marketplace tab (Figure

4) and click the local drop-down menu and select upload. Upload both widgets and add them to the Workplace.



Sensor Widget List Example

Vendor Name

OneSource

Description

Sensor widget list for the Wirecloud Plaftorm.

Sensor Widget Example

Vendor Name

OneSource

Description

Sensor widget for the Wirecloud Plaftorm.

Figure 4: Wirecloud Marketplace

Lastly, connect the wiring between both widgets as shown in Figure 3.

Running the Wirecloud-IoT Application

After successfully starting all the application components and configuring them as explained in the previous sections, the application is ready to run. Go to the Wirecloud interface page located in: http://localhost.

Assuming that the user has registered some IoT devices with the IoT Configuration Manager, query the name of one of those devices (in our case Dorm) using the widget with the form. The second widget receives this information and performs a REST request to the IoT Broker:

```
http://10.0.1.29:8090/ngsi10/contextEntities/Dorm
```

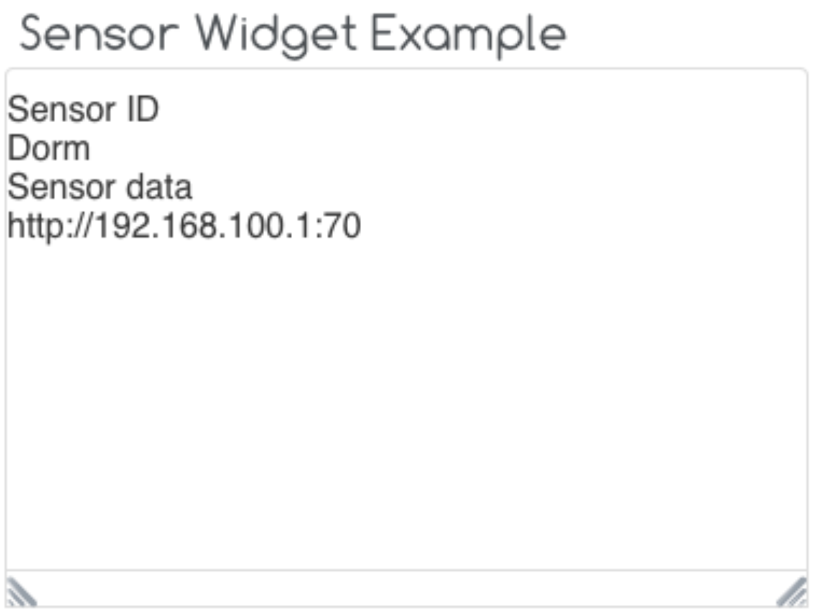
The IoT Broker performs a request to the IoT Configuration Manager using the following XML:

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><discoverContextAvailabilityRequest><entityIdList><entityId
isPattern="false"><id>Dorm</id></entityId></entityIdList><attributeList/><restri
ction><scope><operationScope><scopeType>IncludeAssociations</scopeType><s
copeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">SOURCES</scopeValue></operationScope></scope></restric
tion></discoverContextAvailabilityRequest>
```

To which the IoT Configuration Manager replies with the following XML:

```
<?xml version="1.0" encoding="UTF-8"?><discoverContextAvailabilityResponse>
  <contextRegistrationResponseList>
    <contextRegistrationResponse>
      <contextRegistration>
        <entityIdList>
          <entityId type="Room" isPattern="false">
            <id>Dorm</id>
          </entityId>
        </entityIdList>
        <contextRegistrationAttributeList>
          <contextRegistrationAttribute>
            <name>temperature</name>
            <type>degree</type>
            <isDomain>false</isDomain>
            <metadata>
              <contextMetadata>
                <name>ID</name>
                <type>string</type>
                <value/>
              </contextMetadata>
              <contextMetadata>
                <name/>
                <type/>
                <value/>
              </contextMetadata>
            </metadata>
          </contextRegistrationAttribute>
        </contextRegistrationAttributeList>
        <providingApplication>http://192.168.100.1:70
                                </providingApplication>
      </contextRegistration>
    </contextRegistrationResponse>
  </contextRegistrationResponseList>
  <errorCode>
    <code>200</code>
    <reasonPhrase>OK</reasonPhrase>
    <details xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Result</details>
  </errorCode>
</discoverContextAvailabilityResponse>
```

This XML is then forwarded to the widget that uses it to retrieve some data and present it onscreen. For the sake of simplicity we presented the “id” and the “providingApplicaiton” fields to the user as shown in Figure 5.



Sensor Widget Example

```
Sensor ID
Dorm
Sensor data
http://192.168.100.1:70
```

Figure 5: Output of the sample application