Developing your first application using FI-WARE

Fermín Galán Márquez (fermin@tid.es), Miguel Jimenez (mjmenez@fi.upm.es), Carlos Ralli (ralli@tid.es), Juanjo Hierro (jhierro@tid.es)
Telefónica I+D, Universidad Politécnica de Madrid

Open APIs for Open Minds
Let’s go into detail...
Taking a closer look

Context Broker

subscriptions

Context Producers

update

update

update

1026

Context Consumers

query

notify

notify

Ticket Management

DB

History

Ticket Management

Wirecloud
The NGSI information model

Context Element
- EntityId
- EntityType

“has”
1
n
Context Element attributes
- Name
- Type
- Value

Meta-data
- Name
- Type
- Value

(We are not fully supporting metadata right now)

Electrical Regulator
Node Lamps

Vans
Technicians

…and many more (look to app code)
Typical Orion broker utilization steps

subscribeContext
Availability

registerContext

createContext

updateContext

"Write" context elements information

discoverContext
Availability

Look for existing context elements

queryContext

subscribeContext

To different ways of “reading” context elements information

NGSI9 operation (context availability)

NGSI10 operation (context information)
Context platform available at Campus Party

Nodes AMMS Regulator Vans

Context Broker

![Diagram](image)

Dedicated instances at http://yourvm:1026

“FI-WARE Cloud: bringing OpenStack to the next level” workshop (Wed 10-12am)

(Only for demonstration purposes during the workshop)

Global instance at orion.lab.fi-ware.eu (at FI-LAB cloud)

Nodes AMMS Regulator Vans

Context Broker

LiveDemo testbed

Context Broker

NGSI enabled Sensors

Backend Device Management

updateContext

1026

Context Broker

notify

1028

Echo server

notify

Shared!!!
What is Wirecloud?

[Diagram showing components such as Backend service, Context Broker, NGSI API, Object Storage API, and widget wiring to a Mashup.]
Catalogue of widgets and mashups

- Widget wgt

- Store API

- Application Mashup API

- Local catalogue

- widget .wgt
What can I do with Wirecloud?
To create my own application mashup...

- **Widgets and operators:**
  - Widgets can be developed with any Web technology (HTML, Flash, SVG…) that supports JavaScript.
  - Operators are coded in JavaScript

- **Both widgets and operators can be easily adapted from existing ones, or created from scratch**

- **Open Widget API**
  - JavaScript API
  - **Core Widgets Functionality:**
    - gain access to external resources (cross-domain problem)
    - wiring
    - preferences
    - ...

How to develop a widget?

- Setting up the development environment
  - Eclipse, VIM, etc...
  - Creating the directory structure (WGT)

- Creating the widget:
  - Create the template of the widget (config.xml)
  - Widget view: an HTML file + CSS
  - Widget app logic: JavaScript (using Widget API)

- Testing your widget
  - Uploading the widget to Local Catalogue
    - The WGT file is a ZIP file renamed to .wgt
  - Add widget to a mashup
    - Wiring, preferences, layout…
Widget structure

- **config.xml**
  - definition of the widget (based on WDL)

- **index.html**
  - main view file
  - references JS and CSS resources

- **/js**
  - directory for JavaScript files

- **/css**
  - directory for CSS files

- **/images**
  - directory for images

- **/doc**
  - documentation
The mandatory config.xml file contains the WDL XML template needed to tell Wirecloud what the widget looks like. This includes:

- Widget metadata (non-functional properties)
- User preferences
- Widget inputs and outputs (wiring)
- Link to the actual widget source code
- Default rendering information
<?xml version="1.0" encoding="UTF-8"?>

<Template xmlns="http://wirecloud.conwet.fi.upm.es/ns/template#">
    <Catalog.ResourceDescription>
        <Vendor>Company distributing the widget</Vendor>
        <Name>Widget name</Name>
        <Version>0.0.1</Version>
        <DisplayName>Widget Example</DisplayName>
        <Author>mjimenez</Author>
        <Mail>mjimenez@fi.upm.es</Mail>
        <Description>Full widget description to be shown in catalogue</Description>
        <ImageURI>Absolute or relative path to display image</ImageURI>
        <iPhoneImageURI>Path to image to display on mobile devices</iPhoneImageURI>
        <WikiURI>Path to widget doc</WikiURI>
    </Catalog.ResourceDescription>
</Template>
Config.xml example (II)

```xml
<Platform.Preferences>
  <Preference name="pref_name" type="text" description="..." />
</Platform.Preferences>

<Platform.Wiring>
  <OutputEndpoint name="identifier"
      type="text" label="Label to show on wiring"
      description="description"
      friendcode="same as compatible input endpoint friendcode"/>
  <InputEndpoint name="identifier"
      type="text" label="Label to show on wiring"
      description="description"
      friendcode="same as compatible output endpoint friendcode"/>
</Platform.Wiring>

<Platform.Link>
  <XHTML href="index.html"/>
</Platform.Link>

<Platform.Rendering width="6" height="24"/>
</Template>
```
index.html example

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script type="text/javascript" src="js/main.js"></script>
    <link rel="stylesheet" type="text/css" href="css/style.css" />
  </head>
  <body>
    <!-- Create here or create by JavaScript -->
  </body>
</html>
Using the Open Widget API

- Accessing widget preferences
  
  ```java
  MashupPlatform.prefs.get("google_key")
  MashupPlatform.prefs.set("color", "red")
  MashupPlatform.prefs.registerCallback(callbackFunction)
  ```

- Getting user login name
  
  ```java
  MashupPlatform.context.get("username");
  ```
Wiring endpoints

- Declaration on config.xml

```xml
<Platform.Wiring>
  <InputEndpoint name="my_input_name" type="text" label="my_label"
    description="my_description" friendcode="some_code" />
  <OutputEndpoint name="my_output_name" type="text" label="label"
    description="description" friendcode="url"/>
</Platform.Wiring>
```

- Input endpoints

```javascript
MashupPlatform.wiring.registerCallback("my_input_name", inputListener);
```

- Output endpoints

```javascript
MashupPlatform.wiring.pushEvent("my_output_name", data);
```
Making a cross-domain HTTP request

- Perform the invocation and register asynchronous call-backs:

  ```javascript
  MashupPlatform.http.makeRequest(url, {
    method: 'GET',
    onSuccess: function (transport) {
      var response;
      response = JSON.parse(transport.responseText);
      ...
    },
    onFailure: function (transport) {
      onError();
    }
  });
  ```
Connecting a widget to NGSI

- Register NGSI usage on `config.xml`

```xml
<Requirements> <Feature name="NGSI"/> </Requirements>
```

- Send a notification through NGSI

```javascript
var connection = new NGSI.Connection('http://orion.lab.fi-ware.eu');
connection.updateAttributes([{
  entity: {
    id: 'iss8',
    type: 'Issue'
  },
  attributes:[{
    name: 'technician',
    contextValue: 'tech1'
  }],
  onSuccess: function () { //... },
  onFailure: function () { //... }
}]);
```
Connecting a widget to NGSI

- Subscribe for notifications on certain entities

```javascript
var entityIdList = [
    {type: 'Van', id: '.*', isPattern: true},
    {type: 'Technician', id: '.*', isPattern: true}
];
var attributeList = null; var duration = 'PT3H'; var throttling = null;
var notifyConditions = [{
    type: 'ONCHANGE', condValues: ['name', 'current_position']
}];
var options = {
    flat: true,
    onNotify: process_entities,
    onSuccess: function (data) {
        subscriptionId = data.subscriptionId;
        refresh_interval = setInterval(refreshNGSISubscript, 1000*60*60*2);//2 hours
    }
};
connection.createSubscription(entityIdList, attributeList, duration, throttling, notifyConditions, options);
```
Using Object Storage from a widget

- Register Object Storage usage on config.xml
  
  ```xml
  <Requirements>
  <Feature name="ObjectStorage"/>
  </Requirements>
  ```

- Upload a file
  
  ```javascript
  var object_storage = new ObjectStorageAPI('<Object Storage instance url>');
  var fileParts = ['<a id="a"><b id="b">hey!</b></a>'];
  var myBlob = new Blob(fileParts, { "type" : "text/xml" });

  object_storage.uploadFile('folder_name', myBlob, token, {
    file_name: 'myFile.xml',
    onSuccess: function () {
      alert('File uploaded successfully');
    },
    onFailure: function () {
      alert('Something went wrong while uploading the file');
    }
  });
  ```
Uploading your widget

- Compress (zip) the widget contents
  - Rename it as a .wgt file

- Upload the WGT file to Local Catalogue through WireCloud web interface

Adding widgets from packages

Do you have a widget stored in a wgt file? Then you can upload it to the catalogue by means of this form.
Wiring your widget
Resources

You can always find the most updated documentation of Wirecloud in:
- http://conwet.fi.upm.es/wirecloud

User and Programmer Guide:

Installation and Administration Guide:

Download from Github:
- Wirecloud source code: https://github.com/Wirecloud/wirecloud

Stuff for this workshop
- http://tinyurl.com/fiware-dropbox

https://mashup.lab.fi-ware.eu
Online videos

- FIWARE Live Demo: [http://www.youtube.com/watch?v=Wh_zPsLUg-8](http://www.youtube.com/watch?v=Wh_zPsLUg-8)
- ENVIROFI Demo: [https://www.youtube.com/watch?v=yEXILQYq7s4](https://www.youtube.com/watch?v=yEXILQYq7s4)

- Other videos:
  - [http://www.youtube.com/embed/d7_EG42AHJw](http://www.youtube.com/embed/d7_EG42AHJw) - Building a mashup from Geowidgets linked to geospatial services.
  - [http://www.youtube.com/embed/urDGWSnrbtE](http://www.youtube.com/embed/urDGWSnrbtE) - Using operators in a mashup to allow widgets to send tweets
  - [http://www.youtube.com/embed/kW0sXMxgMLI](http://www.youtube.com/embed/kW0sXMxgMLI) - An example of integration with the marketplace and the catalogue GEs
Sensors

Object Storage

Browser

Wirecloud (javascript runtime)

widgets

Wirecloud (server)

Context Broker

Cosmos

CEP

Ticket Management

History

Location

NGSI IoT Adapter

IoT Backend Device Management
FI-WARE IoT Extended Platform Vision

Plug IoT world to FI-WARE via NGSI notifications.
FI-WARE IoT Functional Architecture (Target)
FIWARE IoT Challenge Architecture

**FIWARE IoT Challenge Architecture**

- **FIWARE**
  - **fizway**
    - **fizway_register**
    - **sendobservation**
    - **addobservation**
  - **registerdevice**
  - **fizway_switchd**

- **CONTEXT BROKER**
  - NGSI9 (Register)
  - NGSI10 (Observations)

- **IDAS**
  - SensorML
  - SensorML

- **z-way server (Razberry HW)**

- **Devices**
Connecting a basic z-wave pack

NGSI (B) | IDAS | CONTEXT BROKER

(A) SensorML

Download the PI image with FIGWAY software at: http://tinyurl.com/figway-img
HOW-TO Get Started - Z-wave basic pack

0) You need: a RaspberryPI + Razbian card plugged in its GPIO

1) Include your z-wave devices into your z-wave network
   Complete your sensor details in our “IoT Challenge HUB doc”
   https://docs.google.com/spreadsheet/ccc?key=0ArcymbqnpOfkdGNYUkpaTF9qRVhkOTFlYW14SkxaZ1E#gid=0

2) Edit “fizway_register” & “fizway” scripts
   Update the Device_ID number of your sensors.

3) Edit the file SensorML/Register_SWITCH
   Update the line containing the callback URL:
   
   "<swe:value>http://1.0.0.1:9999</swe:value>"
   -> Instead of 1.0.0.1 put the RaspberryPI public IP address
   -> Instead of 9999 set the port the switch daemon will be listening (normally, 7777)

4) Check & Edit -if necessary- the figway "Config" file.
   Normally, modifications aren't needed so you can skip (4).
   A Config.example file is provided and comments should make this task really easy.

5) Register all your working sensors by executing
   `>./fizway_register`

6) Launch the script to interconnect your sensors to FI-WARE IoT Backend:
   `>./fizway &`
0) Mount RaspberryPI + Razbian
0.2) Flash FIGWAY.img & Get IP address

mac:~ ralli$
mac:~ ralli$ sudo diskutil unmountDisk /dev/disk1
Unmount of all volumes on disk1 was successful
mac:~ ralli$ sudo dd if=Fiware-RPi-figway.arm-2.0.img of=/dev/rdisk1 bs=1m
3947888640 bytes transferred in 777.424620 secs (5078163 bytes/sec)
mac:~ ralli$
mac:~ ralli$ sudo diskutil unmountDisk /dev/disk1
Unmount of all volumes on disk1 was successful
mac:~ ralli$
mac:~ ralli$ sudo diskutil eject /dev/disk1
Disk /dev/disk1 ejected
mac:~ ralli$

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk0</td>
<td>GUID_partition_scheme</td>
<td>250.1 GB</td>
<td>disk0s0</td>
</tr>
<tr>
<td>disk1</td>
<td>GUID_partition_scheme</td>
<td>3.9 GB</td>
<td>disk1s1</td>
</tr>
<tr>
<td>disk2</td>
<td>GUID_partition_scheme</td>
<td>131.1 MB</td>
<td>disk1s2</td>
</tr>
<tr>
<td>disk3</td>
<td>GUID_partition_scheme</td>
<td>605.0 MB</td>
<td>disk1s3</td>
</tr>
</tbody>
</table>
```
1) Include devices into a z-wave Network

EXCLUDE
- Razberry exclusion mode
- Press 4IN1 inclusion button

INCLUDE
- Razberry inclusion mode
- Press 4IN1 Inclusion button

CONFIGURATION
- Select profile Aeon multisensor 1.18
- Press 4IN1 inclusion button 3 times
- Configure to send Group1 all obs (720s)
1) Include devices into a z-wave Network

EXCLUDE
- Razberry exclusion mode
- Press KEYFOB 4 buttons 5sec (led slow blink)
- Press button “1”, (try twice if it doesn’t work)

INCLUDE
- Razberry inclusion mode
- Press KEYFOB 4 buttons 5sec (led slow blink)
- Press button “1”

CONFIGURATION
- Configure remote KeyFob
  To apply config: WAKE-UP Device
  Press KEYFOB 4 buttons 5sec
  - Press button “2” (Try twice)
  Buttons in pair mode(1-3, 2-4):
  -> SEPARATELY
  Groups to send:
  -> SWITCH ON/OFF ONLY (SEND BASIC SET)
1) Include devices into a z-wave Network

EXCLUDE
- Razberry exclude mode
- Switch ON
- Press ON/OFF 3 times between 1,5sec

INCLUDE
- Razberry inclusion mode
- Switch ON
- Press ON/OFF 3 times between 1,5sec

NO CONFIG is needed.

ALL DEVICES INCLUDED.
DEVICES IDs SHOWN:
2 (4IN1), 3 (KEYFOB), 4 (SWITCH)
1) Include devices into a z-wave Network

**EXCLUDE**
- Razberry exclude mode
- Switch ON (use a screw-driver)
- Press ON 3 times between 1.5 sec

**INCLUDE**
- Razberry exclude mode
- Switch ON (use a screw-driver)
- Press ON 3 times between 1.5 sec

**CONFIGURATION**
- Select Zwave Description Record: Wintop iLED
1) Include devices into a z-wave Network

Provide your sensors to all: “IoT Challenge HUB doc”

https://docs.google.com/spreadsheet/ccc?key=0ArcymbqnpOdkGNYUkpaTF9qRVhkOTFIYW14SkxaZ1E#gid=0

SHARE SENSORs IS A MUST
SHARE SWITCHES IS OPTIONAL
2) Edit "fizway_resgister" & "fizway" scripts

Update correct IDs of your z-wave network (only for Devices you have)
Actuators ports should be: 7777, 7778, etc.
   -> Update scripts “fizway_resgister” and “fizway”.
3) Configure the SWITCH to receive commands

Set the IPv4 Address of your Raspberry PI. Port is SWITCH one (7777).
4) Check & Edit the figway "Config" file

Pre-configured for IoT Challenge. No modifications are needed.
5) Register all working devices in the Backend

Launch de script "./fizway_register"
For every device you’ll see a similar output as shown above.

If it works, your devices are correctly register in the IDAS Backend.
If it fails, increasing DEBUG level in the “Config” file helps a lot.
6) Launch fizway script

Launch de script "./fizway &"
You may redirect the output to a log file if you’ close the window:
  "./fizway >> ./fizway_log &"

You’ll see:
- Devices you are listening to and their Device Number (z-wave network ID).
- Daemons listening for Actuators commands coming from the Backend.
Building your Fi-WARE IoT Apps

1) IDAS ADMIN API – Check Services, Subscriptions & Details
2) IDAS ADMIN API – Check Devices & Details
3) IDAS ADMIN API – Send a command to a z-wave Switch
4) IDAS ADMIN API - Subscribe your App to Devices Notifications
5) IDAS (or Global Context Broker) NGSI API – Get last observations of a device
6) WIRECLOUD API - Connect a Wirecloud widget
Building your Fi-WARE IoT Apps

1) IDAS ADMIN API – Check Services, Subscriptions to Service & Service Details
Building your Fi-WARE IoT Apps

2) IDAS ADMIN API – Check Devices & Device Details
Building your Fi-WARE IoT Apps

3) IDAS ADMIN API – Send a command to a z-wave Switch

Follow “sendcommand” example.

Send: “FIZCOMMAND N”  
N: 0-255. 0 = off, 255 = on
Building your Fi-WARE IoT Apps

4) IDAS ADMIN API – Send a command to a z-wave Switch

mac:idas-api ralli$
mac:idas-api ralli$ more subscribeservicereg
curl -v --request POST --data-binary @subscription_reg_CAMPUS.json --header "content-type:application/json" http://130.206.80.44:5371/m2m/v2/services/CampusofThings/subscriptions

mac:idas-api ralli$
mac:idas-api ralli$ more subscription_reg_CAMPUS.json
{
    "name" : "Subscription_CampusofThings_Register_[YOUR_TEAM_NAME]",
    "type" : "Register",
    "notifyURI" : "http://1.0.0.1:9999"
}
mac:idas-api ralli$
mac:idas-api ralli$ more subscribeserviceobs
curl -v --request POST --data-binary @subscription_obs_CAMPUS.json --header "content-type:application/json" http://130.206.80.44:5371/m2m/v2/services/CampusofThings/subscriptions

mac:idas-api ralli$
mac:idas-api ralli$ more subscription_obs_CAMPUS.json
{
    "name" : "Subscription_CampusofThings_Observation_[YOUR_TEAM_NAME]",
    "type" : "Observation",
    "notifyURI" : "http://1.0.0.1:9999"
}
mac:idas-api ralli$
Building your Fi-WARE IoT Apps

5) IDAS NGSI API – Get last observations of a device

```
mac:idas-api ralli$ curl --request POST http://130.206.80.44:1029/ngsi10/queryContext --header 'Content-Type: application/xml' $CURL_VERBOSE --data-binary @ngsi_query_4IN1.xml

<queryContextRequest>
  <entityIdList>
    <entityId type="Sensor" isPattern="false">
      <id>4IN1:2E:36:21:0004</id>
    </entityId>
  </ entityIdList>
  <attributeList>
    <attribute>Move</attribute>
    <attribute>illuminance</attribute>
    <attribute>temperature</attribute>
    <attribute>relativeHumidity</attribute>
  </ attributeList>
</queryContextRequest>
```
Building your Fi-WARE IoT Apps

6) WIRECLOUD API - Connect a Wirecloud widget
ANNEX: where to find more docs

- IDAS APIs
  [http://www.fi-ware.eu](http://www.fi-ware.eu)
- z-way server:

- Aeon 4IN1 Manual

- Zwave.me KeyFob
  [https://www.uk-automation.co.uk/pdf/zwavemekeyfob.pdf](https://www.uk-automation.co.uk/pdf/zwavemekeyfob.pdf)

- Everspring Switch/Dimmer

- Fibaro RGB SWITCH
• More info at “FI-WARE Cloud: bringing OpenStack to the next level” workshop (Wed 10-12am)

• Contact Person: John Kennedy
  <john.m.kennedy at intel dot com>
Cosmos

- Have a look to [http://tinyurl.com/c0sm0s](http://tinyurl.com/c0sm0s)
- Contact Person: Francisco Romero
  <frb at tid dot es>
Extending Live Demo

Object Storage

Browser

Wirecloud (javascript)

widgets

https://github.com/wirecloud-fiware/live-demo-macs

Ticket Management

Context Broker

Cosmos

CEP

History

https://github.com/wirecloud-fiware/historymod

https://github.com/telefonicaid/fiware-livedemoapp

Adapter

IoT Backend Device Management

https://github.com/wirecloud-fiware/historymod

https://github.com/wirecloud-fiware/live-demo-macs

https://github.com/telefonicaid/fiware-livedemoapp
Useful additional references

- **Workshop Homepage** [http://tinyurl.com/fiware-cp-ws1](http://tinyurl.com/fiware-cp-ws1)

  - With information about FI-WARE GEIs, e.g. Orion Context Broker, Wirecloud, etc.

- **Dropbox for Workshop stuff**: [http://tinyurl.com/fiware-dropbox](http://tinyurl.com/fiware-dropbox)

- **Dropbox for Raspberry Pi image**: [http://tinyurl.com/figway-img](http://tinyurl.com/figway-img)
  - Use the following as backup in case of problems: [http://130.206.82.17](http://130.206.82.17)
Thanks!

- http://fi-ppp.eu
- http://fi-ware.eu
- Follow @Fiware on Twitter!