

OSGi: Simplifying the IoT Gateway

 **eclipsecon Europe**
Ludwigsburg, Germany, 3 - 5 November 2015

 **OSGi**TM
Community
Event 2015

Walt Bowers
Technical Solutions Architect -
Eurotech

Outline – Where we are headed

- **Eurotech Overview**
- **IoT Gateway Complexity**
- **Making IoT Gateway's Simple**
- **Use Case – Cold Chain**
- **How OSGi is leveraged**
- **Demo**

Eurotech Overview

Eurotech's Essence

- One of the world top players in the Embedded Computers market
- 20+ Years of experience in “M2M” and distributed systems
- Behind the products of more than 20 Global 500 companies
- Strong vertical market competences:
 - Industrial & Logistics
 - Transportation
 - Defense & Security
 - Healthcare & Medical



Eurotech Overview

Our Activity

- **Active in Eclipse IoT**
 - One of founding member companies
- **Contributors to Kura**
 - Contributed the original code from ESF
 - <https://www.eclipse.org/kura/>
- **OSGi Alliance**
 - Long time adopters/New Members
 - IoT Expert Group
 - <http://osgi.org>



The Eurotech IoT Approach

Overview

Application Layer



Analytics Mining ERP CRM
Enterprise Applications Databases CEP
aPaaS SaaS Enterprise IT Big Data

Application Integration Layer

M2M / IoT Integration Platform



System Infrastructure

Communication Infrastructure

Field Infrastructure

MQTT



Device HW



M2M Integration / IoT Application Enablement / Device and Data Management Platform

Public Cloud Private Cloud Aggregators & On-Premise Platforms

SIM Management & Communication Infrastructure

Optimum M2M / IoT Protocols

Gateway, OS, Security
Gateway Application Framework
Certifications, Device Connections

Sensors, HMIs, Actuators, etc.

IoT Gateway Software Development

Why is it so difficult?



fragmentation



complexity



lock-in

Fragmentation



Sensor Protocols

- ZigBee
- Z-Wave
- CANBus
- MODBus
- Bluetooth
- BLE
- DECT

Hardware

- ARM
- Intel

Standards

- oneM2M
- Thread
- AllSeen
- Industrial Internet Consortium
- IEEE

Protocols to Cloud

- MQTT
- LWM2M
- CoAP
- AllSeen

Complexity



Security

- Sensors
- Data
- Network

Reliability

- Store and forward
- Best Effort
- Guaranteed

Network Management

- LAN
- WAN
- Cell
- Always on
- On Demand
- Usage

Lifecycle Management

- Deployment/Install
- Upgrade
- Provisioning/Configuration

Lock-in



Hardware

Sensor(s)

Protocol(s)

Vertical Market

How to make it simple?

OSGi



Open
Source

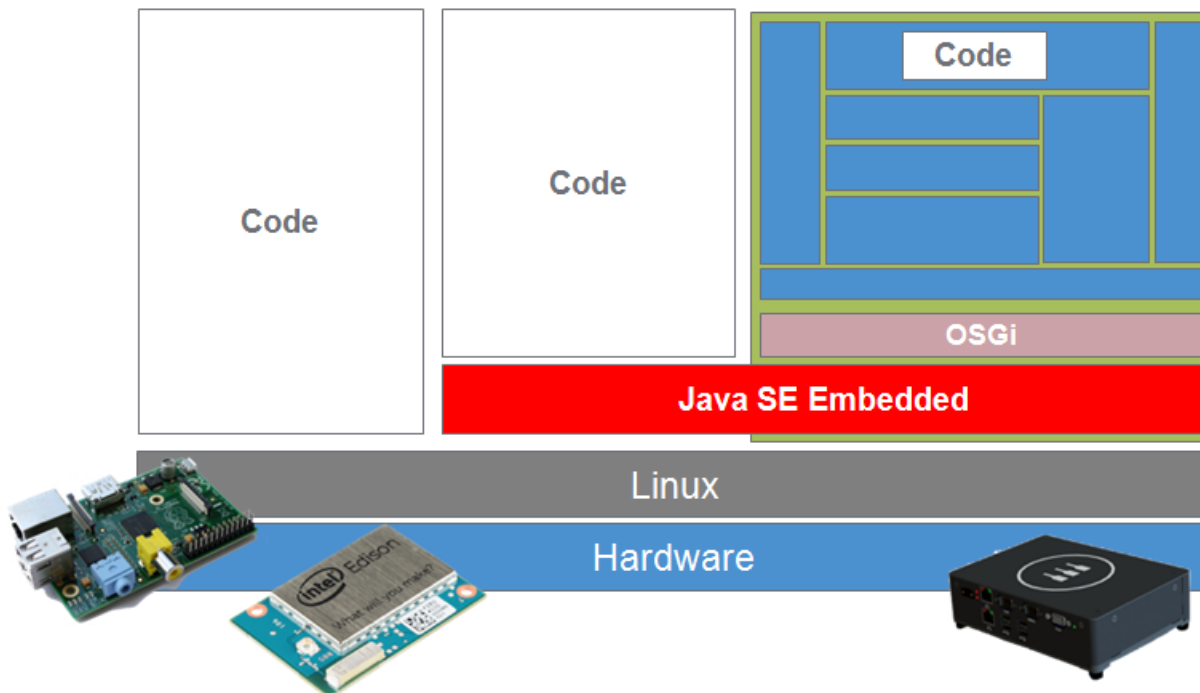


Why an OSGi IoT Gateway Stack?

Increase productivity

Developer's Productivity

- Modular
- Services – Reusable and discoverable
- Easier integration into complex systems
- Isolation from Fragments



Why Open Source IoT Gateway Stack?

IoT Gateway Challenges:

- Pressure to add value in shrinking timeframes
- Velocity of technology changes outstrips staffing
- Interoperability trumps exclusive differentiation
- Quest for quality w/o lock-in

Open Source is the Answer!



Founded in 2012 by

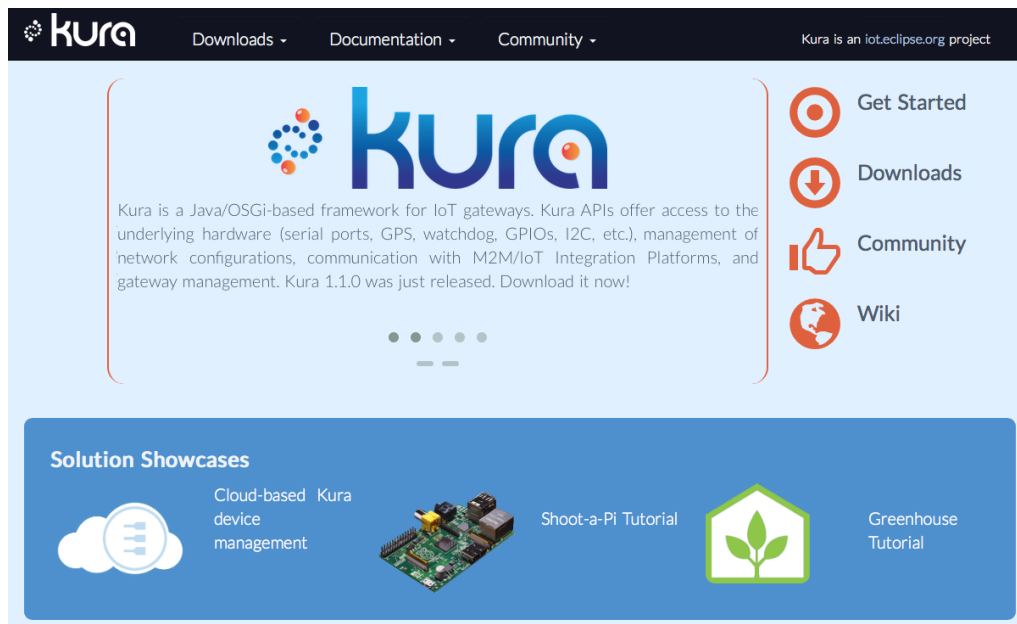


- **Now ...**
 - 23 Members
 - 15+ new projects
 - 1M+ lines of source code
 - The fastest growing Eclipse workgroup

<http://www.slideshare.net/blackducksoftware/io-t-and-open-source>

Eclipse Kura

Open OSGi Framework for IoT Gateways



The screenshot shows the Eclipse Kura website. The header includes the Kura logo, navigation links for Downloads, Documentation, and Community, and a note that Kura is an IoT Eclipse project. The main content area features the Kura logo and a description: 'Kura is a Java/OSGi-based framework for IoT gateways. Kura APIs offer access to the underlying hardware (serial ports, GPS, watchdog, GPIOs, I2C, etc.), management of network configurations, communication with M2M/IoT Integration Platforms, and gateway management. Kura 1.1.0 was just released. Download it now!'. To the right are links for Get Started, Downloads, Community, and Wiki. Below this is a 'Solution Showcases' section with icons and links for Cloud-based Kura device management, Shoot-a-Pi Tutorial, and Greenhouse Tutorial.

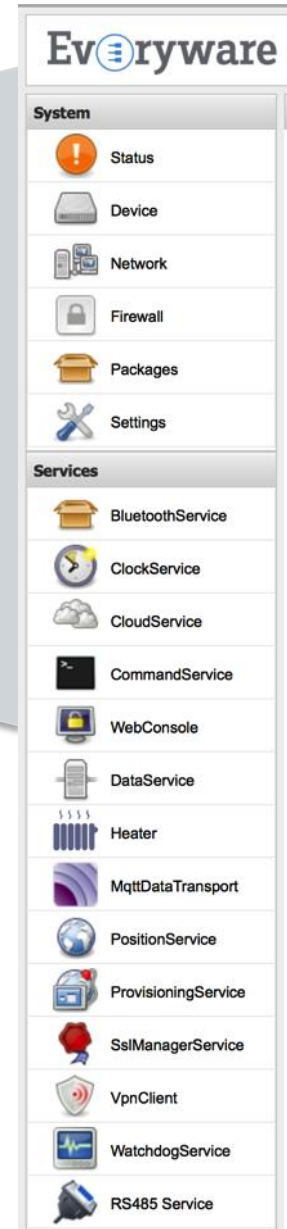
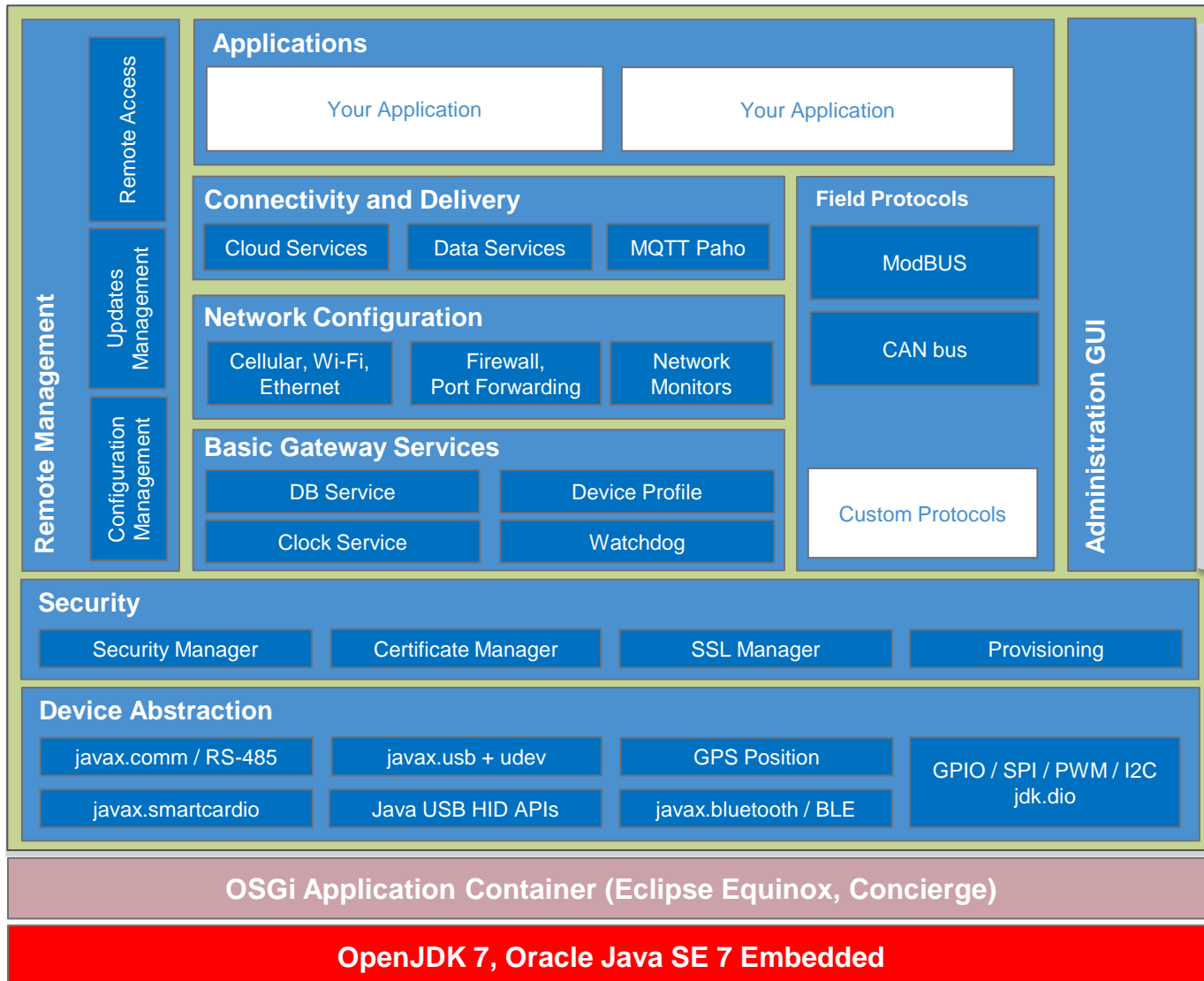
<https://www.eclipse.org/kura/>

<https://iot.eclipse.org/java/>






The banner features the Eclipse IoT logo and the text 'OPEN IOT STACK FOR JAVA DEVELOPERS'. Below this, it states: 'Eclipse IoT has the open source technology that makes it easy for Java developers to create IoT applications.' The background is a dark, circuit-like pattern.

Kura Under the Covers



Developer's Experience

Emulate on PC	Deploy on Target	Cloud Managed
		
<p>Start developing your IoT /M2M application in the comfort of your PC.</p> <ul style="list-style-type: none">• Full Eclipse Integration• Target Platform Definition• Emulated Services• Run/Debug from Eclipse• Support Mac/Linux Hosts	<p>When you are ready, deploy your application on the gateway.</p> <ul style="list-style-type: none">• One-click Deployment• Eclipse Plugin• Remote Debugging	<p>Provision and manage your applications in field devices from the Cloud.</p> <ul style="list-style-type: none">• Remote OSGi Management via MQTT• Web-based Console

From Prototype to Production

Efficient Development & Investment Protection

Software portability
across HW Platforms

Industrial
M2M/IoT
Gateways



Open Hardware



Vertical Market Example Use Case

Use Case: Cool Chain Monitoring

Application:

The customer, a mid size company growing and selling vegetables required a monitoring solution for their green houses (temperature, light, humidity) and for their delivery trucks (temperature, cool chain monitoring).

The monitoring of both the green houses as well as the trucks is mainly to protect the investments in terms of “goods”. The truck solution was especially appealing because of the ability to react when a cooling system fails before the load is lost.



Product:

ReliaGATE 10-20
ReliaCELL 10-20



Key Success Factors:

- ③ Short time to market due to EDC approach
- ③ ESF
- ③ Simple dashboards provided by Eurotech
- ③ Alarms sent when reaching temperature thresholds
- ③ Flexible hardware platform

Cool Chain Monitoring



Wireless
Open/Closed Sensor
Transmit when door is
opened and closed.
Show door status on
web / mobile app



Wireless Temperature Sensor
Show interior cooler
temperature
Correlate to dashboard at
defined demo interval



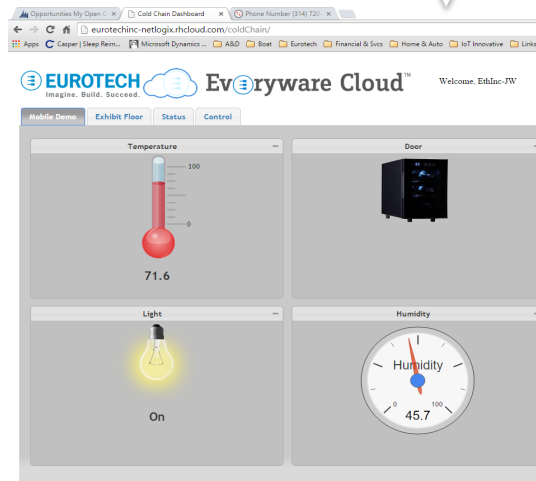
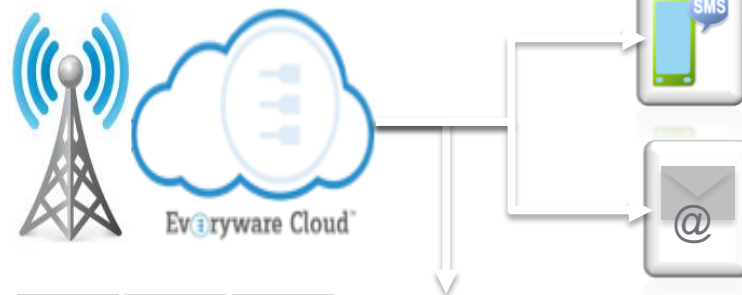
Wireless Humidity Sensor
Show interior cooler humidity level
Correlate to dashboard at defined
demo interval

Monnit 900 MHz USB Wireless
Gateway – Inserted into
ReliaGATE 10-20 USB port.
Application bundle(s) running
locally – sensor reporting
intervals configured via ESF



Cold Chain Demo
Products involved:

- Small AC powered cooler
- Monnit Wireless Sensors & USB Gateway
- ReliaGATE 10-20 (or other)
- ReliaCELL for cellular connection
- ESF & EC
- Dashboard – should display via PC browser and mobile browser



<http://www.monnit.com>

Process

- Install Kura on Raspberry Pi
- Hook up USB Dongle
- Use Kura USB services to access Monnit APN
- Use Kura Cloud service to publish
- Use Kura Configuration service for configuration
- In less than 2 days, publishing Temperature to the cloud.



What was the slowest part:

- Decoding bytes from the APN.

Frame=[c5,0e,04,55,24,f6,00,00,e2,e2,90,02,00,10,09,01,5c]

RSP=[55, 24, f6, 00, 00, e2, e2, 90, 02, 00, 10, 09, 01]

apRSSI:226

DeviceID:63012

sensorType:2

temperature:79.0

voltage:2.0

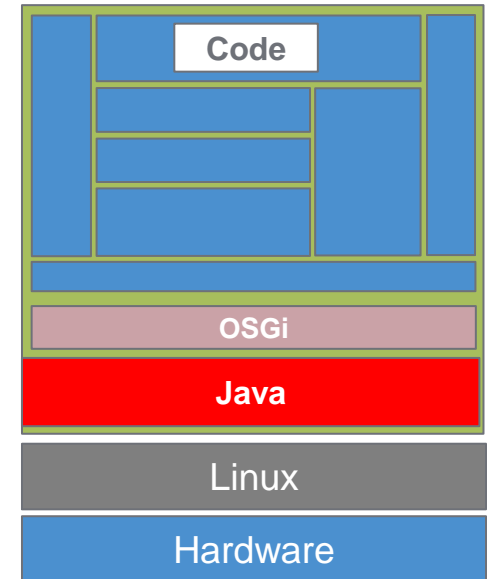


- Learning the instruction sequence
- Main gateway functions already provided.

OSGi Enablement:

The secret sauce

- **OSGi is the key that makes this so easy**
 - Modularity
 - Services
- **Some key OSGi services leveraged by Kura**
 - Declarative Services
 - Metatype Service
 - Configuration Admin Service
 - Deployment Admin Service



Declarative Services - CloudService

```
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0"
  name="org.eclipse.kura.monnit.MonnitGateway"
  activate="activate"
  deactivate="deactivate"
  modified="updated"
  enabled="true"
  immediate="true"
  configuration-policy="require">
<implementation class="org.eclipse.kura.monnit.MonnitGateway"/>

<property name="service.pid" type="String" value="org.eclipse.kura.monnit.MonnitGateway"/>
  <service>
    <provide interface="org.eclipse.kura.monnit.MonnitGateway"/>
  </service>

  <reference name="CloudService"
    policy="static"
    bind="setCloudService"
    unbind="unsetCloudService"
    cardinality="1..1"
    interface="org.eclipse.kura.cloud.CloudService"/>
```

Declarative Services - CloudService

public class MonnitGateway implements ConfigurableComponent, CloudClientListener

```
{  
...  
    public void setCloudService(CloudService cloudService) {  
        m_cloudService = cloudService;  
        ...  
        m_cloudClient = m_cloudService.newCloudClient(APP_ID);  
    }  
  
    public void unsetCloudService(CloudService cloudService) {  
        m_cloudService = null;  
    }  
  
    public void doPublish()  
    {  
        // Publish the message  
        try {  
            //m_cloudClient.publish(topic, payload, qos, retain);  
            m_cloudClient.publish(pubTopic, payload, qos, retain);  
            s_logger.info("Published to {} message: {}", pubTopic, payload);  
        }  
        catch (Exception e) {  
            s_logger.error("Cannot publish topic: "+ pubTopic, e);  
        }  
    }  
...  
}
```

Metatype Service

```
<MetaData xmlns="http://www.osgi.org/xmlns/metatype/v1.2.0" localization="en_us">  
  <OCD id="org.eclipse.kura.monnit.MonnitGateway"  
    name="Monnit"  
    description="Monnit Gateway Application. Configuration params for gateway, apn and  
sensors. ">  
  
    <Icon resource="http://eurotechinc-netlogix.rhcloud.com/images/images/usb.png" size="32"/>  
  
    <AD id="reportingInterval"  
      name="reportingInterval"  
      type="Integer"  
      cardinality="0"  
      required="true"  
      default="60"  
      description="Reporting Interval. 0-43200 seconds Amount of time the sensors will wait  
before sending a message when not in aware state."/>
```

Metatype Services – Automatic Admin GUI

The screenshot displays the Kura Automatic Admin GUI. On the left, there is a sidebar with two main sections: 'System' and 'Services'. The 'System' section includes links for Status, Device, Network, Firewall, Packages, and Settings. The 'Services' section includes links for DataService, Monnit (which is currently selected), MqttDataTransport, PositionService, SslManagerService, and WatchdogService. The main content area is titled 'Monnit' and contains configuration parameters for the Monnit Gateway Application. At the top of this section are 'Apply' and 'Reset' buttons. The configuration parameters are as follows:

- * sensorID:** 64148
SensorID to register.
- * registerSensor:** ☐ true ☒ false
Register the SensorID. If set Sensor is registered with no other configuraton changes.
- * reportingInterval:** 60
Reporting Interval. 0-43200 seconds Ammount of time the sensors will wait before sending a message when not in aware state.
- * awareState:** False
Reporting Interval in Aware State 10-400 seconds Ammount of time the sensors will wait before sending a message when in aware state.
- * linkModeInterval:** 110
Amount of time a device will wait between consecutive link attempts. 1-24 interpreted as hours, 101-160 interpreted as minutes.
- * publish.rate:** 120
Regul publishing of values in seconds.
- * publish.qos:** Fire and forget
Default QoS to publish the messages with.
- * publish.retain:** ☐ true ☒ false
Default retaining flag for the published messages.

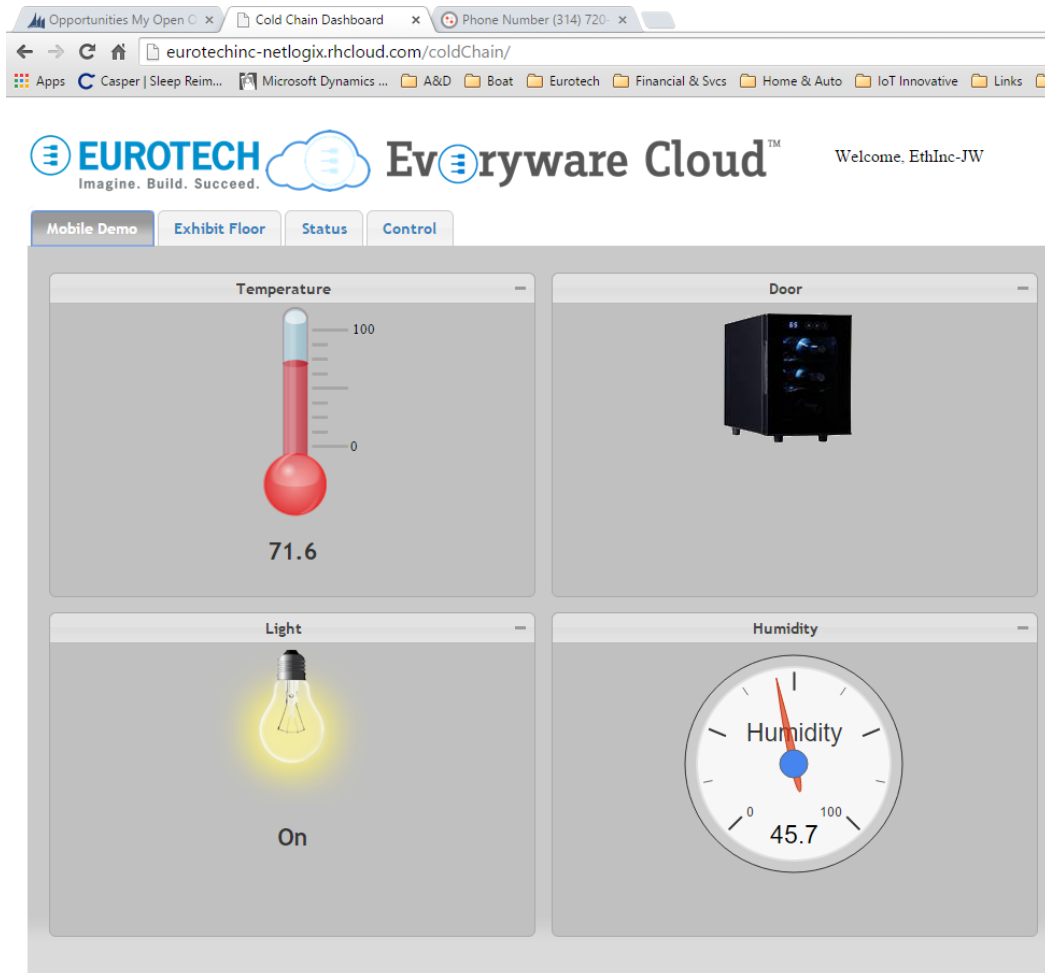
Configuration Admin Service

```
public void updated(Map<String,Object> properties)
{
    s_logger.info("Updated Monnit...");
    System.out.println("Updated Monnit...");
    // store the properties received
    m_properties = properties;
    for (String s : properties.keySet()) {
        s_logger.info("Update - "+s+": "+properties.get(s));
        System.out.println("Update - "+s+": "+properties.get(s));
    }

    ...
}
```


Demo

Here it is all together





Thank You!

www.eurotech.com



eclipsecon Europe

Ludwigsburg, Germany, 3 - 5 November 2015

Evaluate the sessions at www.eclipsecon.org

+1

0

-1