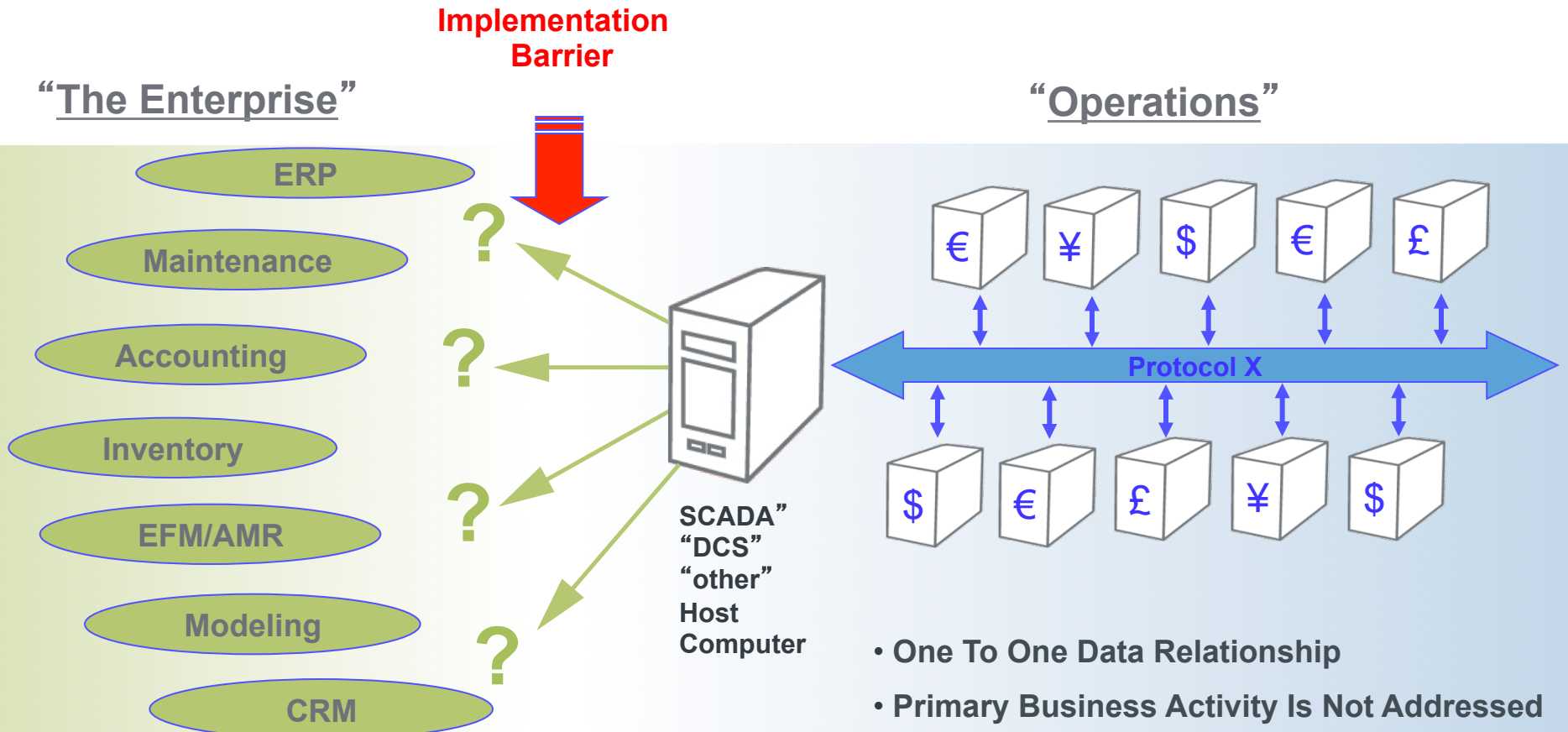


Using MQTT and Eclipse Tools to Write an End-to-End M2M Application

Wes Johnson
Principal Software Engineer
Eurotech

The Internet of Things

Decoupling Producer / Consumer



- One To One Data Relationship
- Primary Business Activity Is Not Addressed
- 60%-90% of Project Effort on Data Delivery
- Costly Bandwidth Consumption
- Protocol is the Transport – Can't break apart

Introduction

Additional considerations

- **Getting data to the Enterprise**
 - Applications need to be modular and extendable
 - Bandwidth is expensive
 - Networks are unreliable
 - Some components can be on private networks

MQTT

The Solution

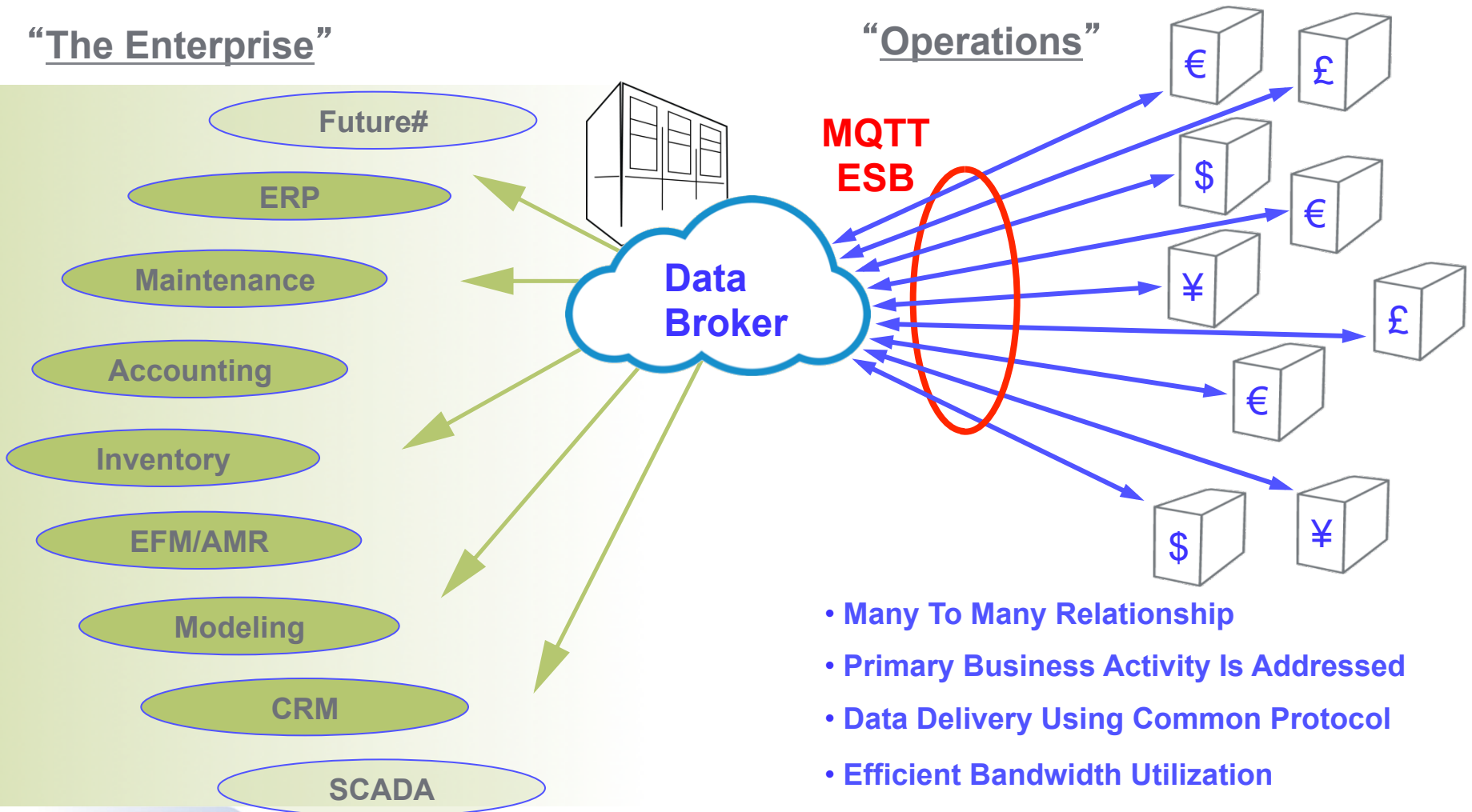
- **Publish/Subscribe Model – Broker and clients**
 - Decouple data producers from consumers to increase modularity and extensibility
- **Relies on TCP because it works**
- **Limits message overhead**
 - Uses less bandwidth
- **Doesn't impose complex/restrictive requirements on data format**
 - Let the application control its data
- **Specifies Quality of Service to ensure delivery is a specific way**
 - 3 levels of QoS

The Internet of Things

Decoupling Producer / Consumer

“The Enterprise”

“Operations”



- Many To Many Relationship
- Primary Business Activity Is Addressed
- Data Delivery Using Common Protocol
- Efficient Bandwidth Utilization
- Utilizes TCP/IP Network Topology

MQTT

Players

- **Broker**
 - Responsible for accepting client connections and message routing
- **Client**
 - Establishes a persistent connection to the broker to provide and/or consume data
- **Message Components**
 - Topic
 - Hierarchical – usa/virginia/reston with wildcards # and +
 - Defined by the application
 - Payload
 - Byte array
 - Quality of Service
 - Fire and forget, At least once, Once and only once

MQTT

Important options not shown in the code examples

- **Client Keep Alive**
 - Maintains client session awareness
 - Enforced via client initiated 'pings'
- **Last Will & Testament**
 - Published on behalf of a client
- **Message Retention**
 - Tells the broker to hang on to messages
- **Clean Start**
 - Tells the broker to forget about the previous client connection
- **MQTT Persistence**
 - Allows local persistence of data on the client side

MQTT Broker

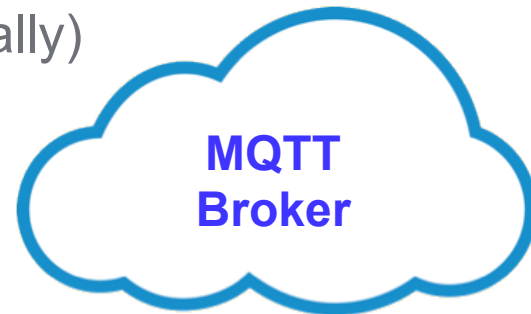
Mosquitto – an open source implementation

- **MQTT v3.1**
- **Free BSD License**
- **\$SYS support**
- **Access Control List support**
 - Allows only authenticated clients to connect
 - Supports limiting the topics clients can pub/sub on
 - Can be bound to an SQL database

MQTT Broker

Mosquitto

- **Starting the broker and configuration options**
 - Runs on Linux, Mac, Windows
 - Password setup (who has access)
 - ACL Setup (what clients can pub/sub on – plus support for pattern matching)
 - Basic scaling options (max clients, inflight messages, queued messages)
 - Timing parameters (retry for QOS resending)
 - Persistence options (how to store locally)
 - SQL options (how to authenticate)
- **Let's start the Broker!**



MQTT Client

Options

- **C client exists at Eclipse Paho now**
- **Java Client coming soon**
 - This example uses an older version of what will soon be released through Eclipse Paho
- **Eclipse MQTT client plugin coming soon via Eclipse Paho**
- **Lots of implementations at <http://mqtt.org>**
 - C/C++
 - Java
 - Perl
 - PHP
 - Python
 - .NET

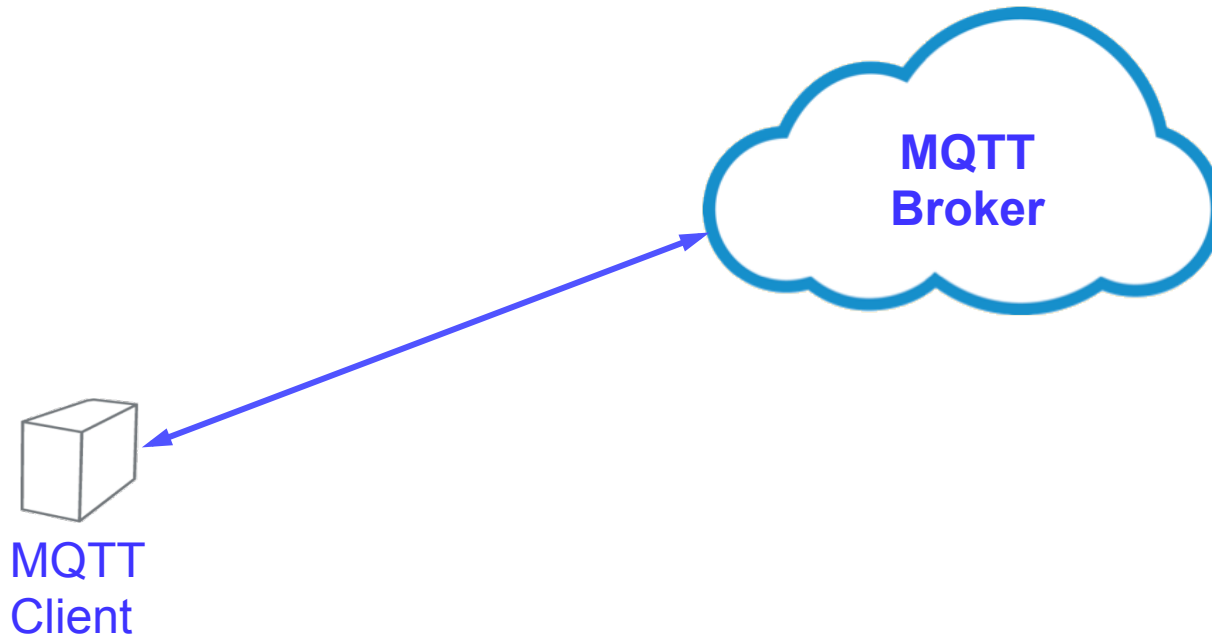


MQTT
Client

MQTT Client

A data producer example

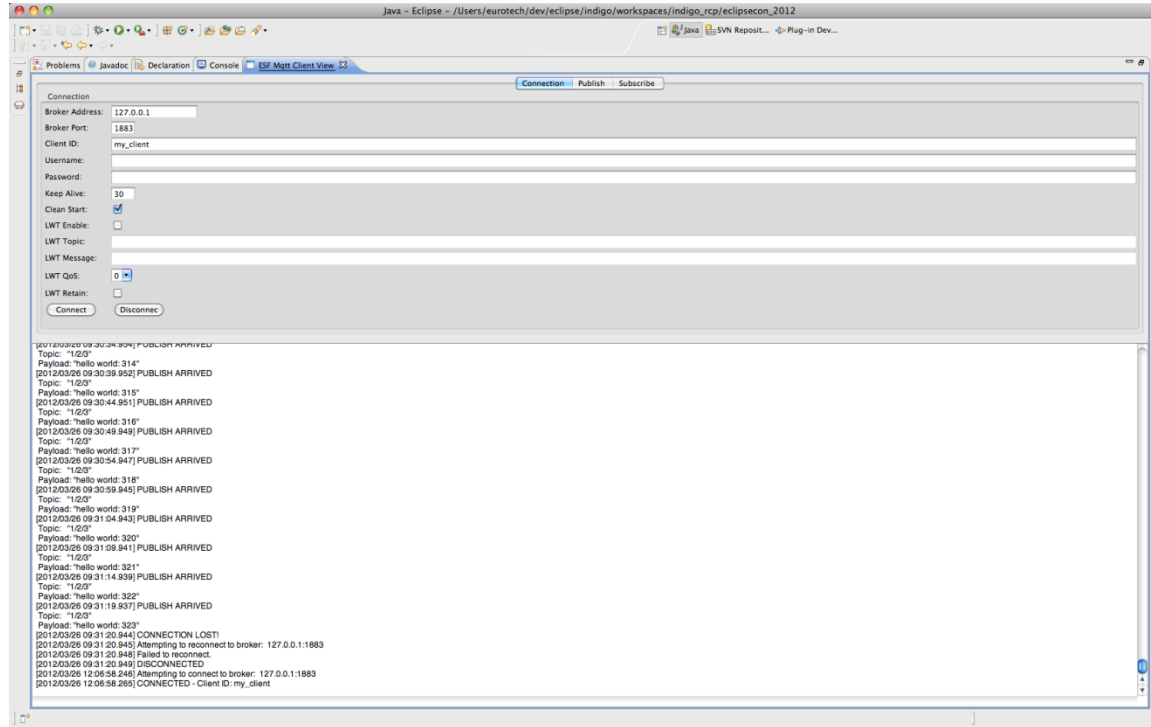
- **Basic Flow**
 - Create the MQTT client
 - Connect
 - Start publishing data
- **Let's see the code and start the client!**



MQTT Client

As an Eclipse Plugin

- Three basic controls
 - Connect/Disconnect
 - Publish
 - Subscribe
- Connection Parameters
 - Username/password
 - Keep alive
 - Clean start
 - LW&T
- Will soon be released through the Eclipse Paho project
- Let's take a look and connect to the broker!



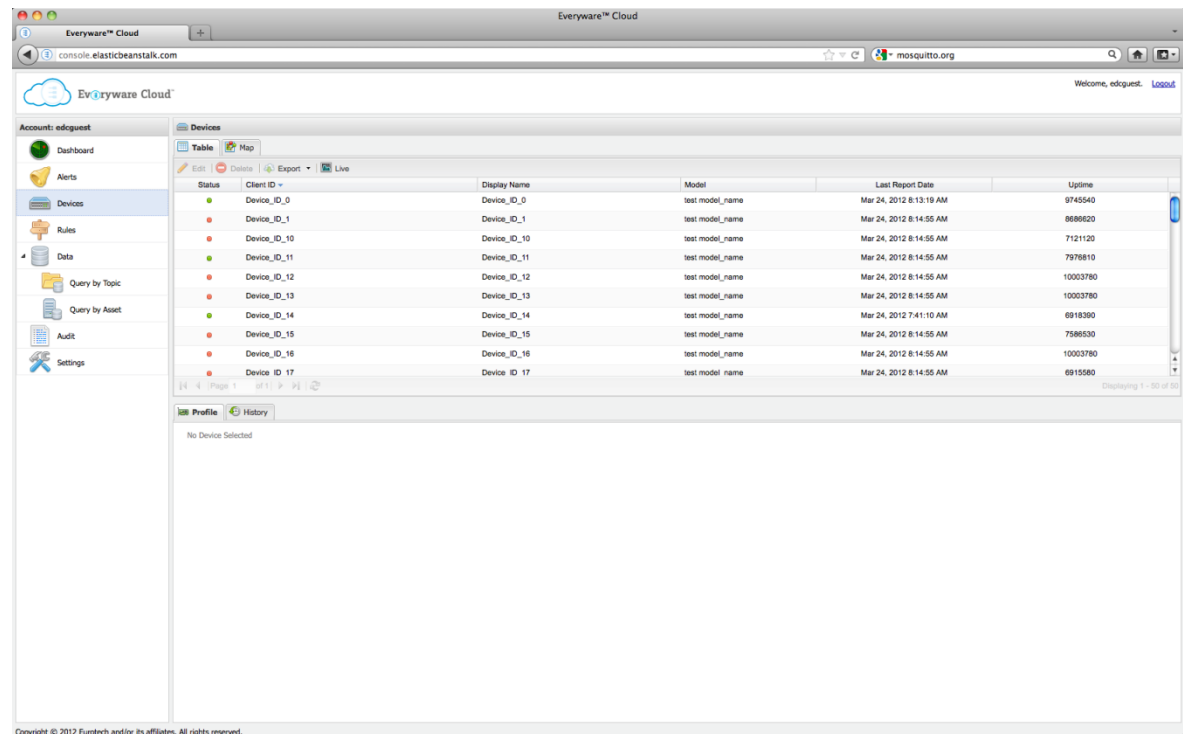
MQTT Client

A data consumer example

- **Basic Flow**
 - Create the MQTT client
 - Connect
 - Subscribe
 - Handle publish notifications as needed
- **Let's see the code and start the client!**

MQTT in a more complex example

- Broker in the cloud
- MQTT clients on devices publish to the broker
- Store data in the cloud
- Web console provides view of devices, query support, different views of data.



The screenshot displays the 'Everyware Cloud' web console interface. The main content area shows a table of devices with the following columns: Status, Client ID, Display Name, Model, Last Report Date, and Uptime. The table lists 17 devices, each with a unique Client ID and a corresponding Display Name and Model. The 'Last Report Date' for all devices is 'Mar 24, 2012 8:14:55 AM', and the 'Uptime' values range from 6915580 to 9745540. The interface includes a navigation sidebar on the left with options like Dashboard, Alerts, Devices, Rules, Data, Query by Topic, Query by Asset, Audit, and Settings. The top of the page shows the account name 'edgquest' and a 'Logout' link. The bottom of the page contains a copyright notice: 'Copyright © 2012 Eurotech and/or its affiliates. All rights reserved.'

Status	Client ID	Display Name	Model	Last Report Date	Uptime
●	Device_ID_0	Device_ID_0	test_model_name	Mar 24, 2012 8:13:19 AM	9745540
●	Device_ID_1	Device_ID_1	test_model_name	Mar 24, 2012 8:14:55 AM	8686620
●	Device_ID_10	Device_ID_10	test_model_name	Mar 24, 2012 8:14:55 AM	7121120
●	Device_ID_11	Device_ID_11	test_model_name	Mar 24, 2012 8:14:55 AM	7976810
●	Device_ID_12	Device_ID_12	test_model_name	Mar 24, 2012 8:14:55 AM	10003780
●	Device_ID_13	Device_ID_13	test_model_name	Mar 24, 2012 8:14:55 AM	10003780
●	Device_ID_14	Device_ID_14	test_model_name	Mar 24, 2012 7:41:10 AM	6918390
●	Device_ID_15	Device_ID_15	test_model_name	Mar 24, 2012 8:14:55 AM	7586530
●	Device_ID_16	Device_ID_16	test_model_name	Mar 24, 2012 8:14:55 AM	10003780
●	Device ID 17	Device ID 17	test_model name	Mar 24, 2012 8:14:55 AM	6915580

Resources

- <http://www.eclipse.org/paho/>
- <http://mqtt.org>
- <http://mosquitto.org>
- For further questions, you can reach me at
wes.johnson@eurotech.com
913-549-1000 x104



Thank You

www.eurotech.com