Global Summit 2016 IoT Experience Setup

Version 0.8

InterSystems®

Inhaltsverzeichnis

1 About this document

This document describes the infrastructure and setup used for the Internet of Things experience lab session at Global Summit 2016.

2 Network and container infrastructure

The following figure outlines the network and container infrastructure for the IoT experience.

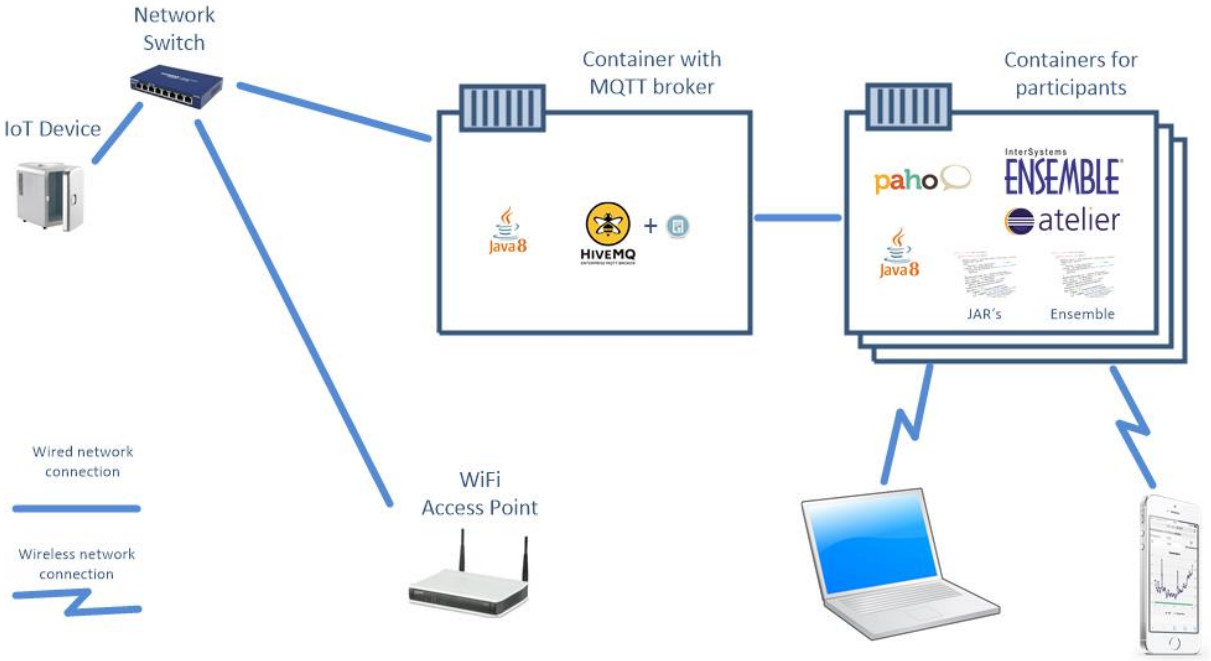

Figure 1: Infrastructure overview

2.1 Network setup

All components communicate using the TCP-based protocol MQTT. Hence, TCP communication between the device, the MQTT message broker and Ensemble must be possible.

2.1.1 Device

The Device is equipped with an Ethernet shield, so it can be connected to a network using a network cable (as of now, WiFi is not built-in). The device does not use DHCP, its IP address needs to be configured using the Arduino IDE. The same is true for the IP address and the MQTT port number of the MQTT broker the device communicates with. The updated Arduino sketch then needs to be transferred to the device using USB.

2.1.2 MQTT Message Broker

The MQTT Message Broker HiveMQ is installed in a dedicated container. It is accessed both from the device and the Ensemble instances running in the participants containers. HiveMQ uses the default port 1833 for MQTT communication, both inbound and outbound. The firewall of the container needs to be configured accordingly.

Mapping the default MQTT port (1833) to some other port number for external access is possible. The external port number will need to be configured in both the device and the Ensemble production.

2.1.3 Ensemble

The Ensemble instance(s) need to have network access to the MQTT message broker. The Firewall of the container(s) need to be configured accordingly. Furthermore, participants will heavily use the Ensemble management portal, the DeepSee user portal and potentially their own mobile devices to work with a Zen Mojo application. The CSP port of the Ensemble instance hence needs to be visible externally. If the participants want to use their mobile devices to access their Ensemble instance, there needs to be network access between their mobile device and their container. This can be achieved by either having the containers be running in the cloud or by having both the containers and the mobile devices be running in the same network (e.g. by providing an WiFi access point for the mobile devices).

2.2 Container setup (MQTT container)

2.2.1 OS

- Java must be installed
- Java libraries must be installed
- HiveMQ must be installed
- SSH access must be available (to start/stop/monitor HiveMQ)

2.3 Container setup (participants containers)

2.3.1 OS

- Java must be installed
- Java libraries must be installed
- Java business hosts must be available
- Network access must be available as described above

2.3.2 Ensemble

In Ensemble, the following items need to be configured:

- Namespace ‘ENSEMBLE’
- Ensemble web app DeepSee-enabled
- Ensemble user gs16/2016 with role %All
- Workflow user gs16
- Workflow role “Supplier” with user gs16
- Credentials sys
- System Default Settings
- IoT Schema Category ‘IoT MQTT’ for XML VDoc

3 Software components

3.1 Arduino

For the experience, a small refrigerator was equipped with a couple of sensors and an Arduino with Ethernet shield used for hardware connectivity (sensors, display), local processing of sensor readings and network connectivity. Figure 2 outlines the communication between the device and Ensemble using the MQTT protocol.

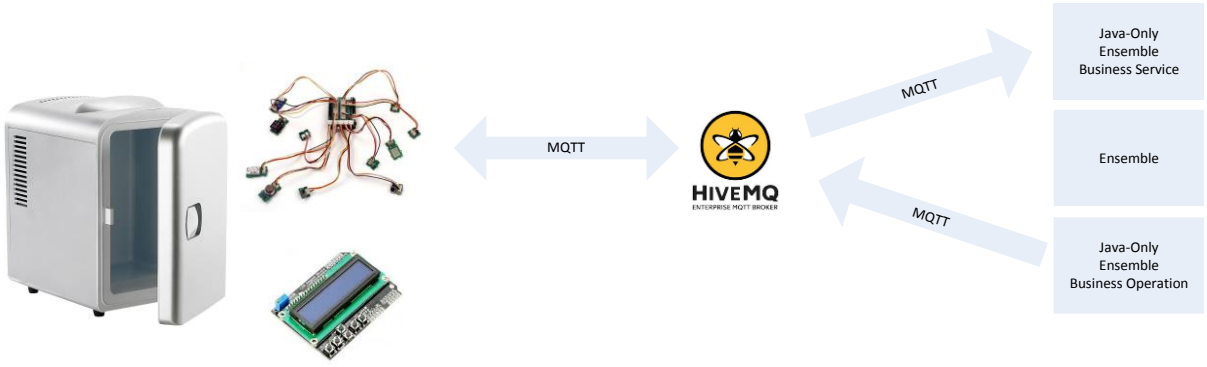

Figure 2: MQTT connectivity

The required Arduino code (called a 'sketch') was implemented using the Arduino IDE and the C-like language provided by that. The code uses a couple of external libraries (e.g. for sensors using the 1wire protocol, I^2C, MQTT etc.):

- #include <Wire.h>
- #include <LiquidCrystal_I2C.h>
- #include <NewPing.h>
- #include <OneWire.h>
- #include <DallasTemperature.h>
- #include <SPI.h>
- #include <Ethernet.h>
- #include <PubSubClient.h>

Using these libraries, the code written for the IoTexperience is very concise and contains around 500 lines of custom code (including comments).

3.2 Java

To communicate between the MQTT message broker and Ensemble, the new Java Business Hosts of Ensemble are used. These are written completely in Java, the resulting JAR files and the JAR files of any external Java library need to be available in the container.

Using a Java IDE (e.g. Eclipse or Atelier), two Java Business Hosts are created:

1. Java Business Service for receiving MQTT messages
2. Java Business Operation to send MQTT messages

Figure 3 shows the project structure on the Java side for the two Business Hosts – including the external libraries used.

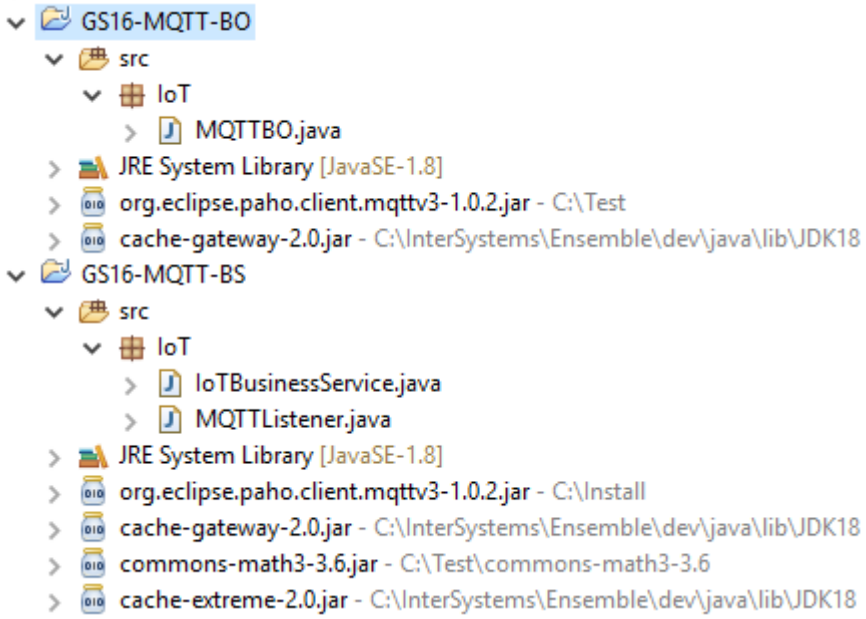

Figure 3: Java project structure

The three Java classes contain around 400 lines of code (including comments). Figure 4 outlines the process of writing Java Business hosts.

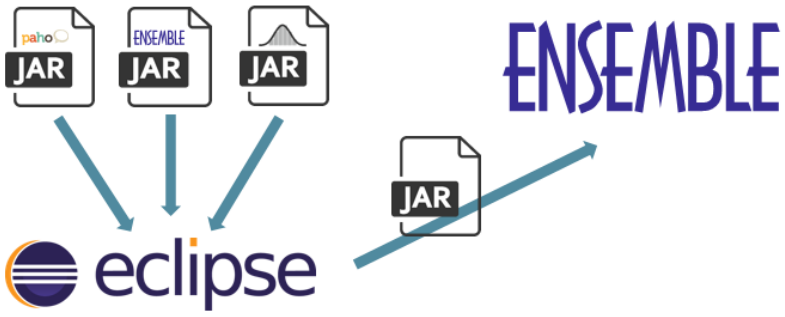

Figure 4: Writing Java Business Hosts

3.3 Ensemble

Figure 5 outlines the components used in the Ensemble production.

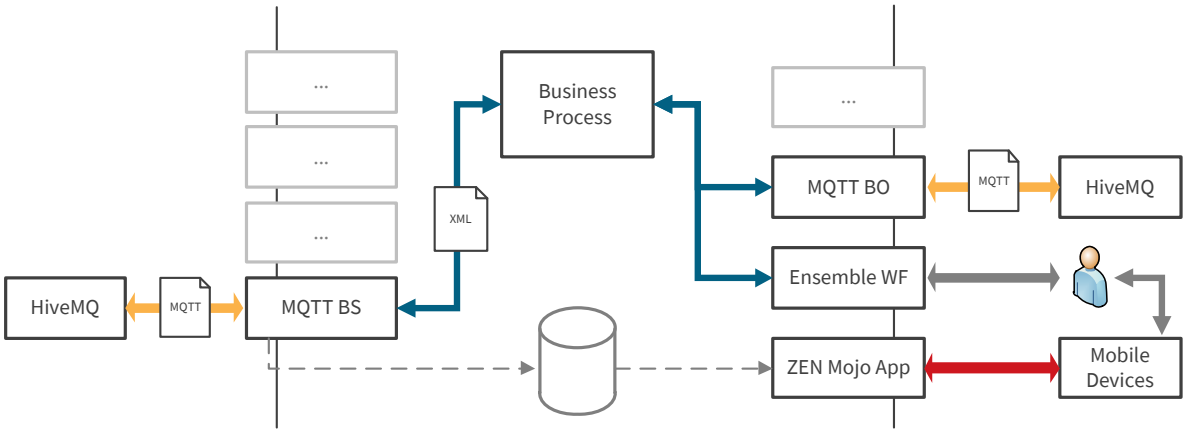

Figure 5: Ensemble production overview

In Ensemble, the only components that is implemented directly using COS are the ZEN Mojo App and the data structure for the sensor data. These components contain around 500 lines of COS code (including comments). The participants are not required to work directly with this code.

All other components are either generated during the assignments (i.e. the proxy class for the MQTT BS) or created using the graphical editors provided by Ensemble (Business Process, Routing Rules, Ensemble Workflow etc.) – this is what the participants are working with/using in a web browser.

Figure 6 shows a list of the classes. Additionally, the following components are required in Ensemble:

- XML VDoc-specification
- System default settings
- Ensemble credentials
- Workflow user definition
- Favorite links for Management portal

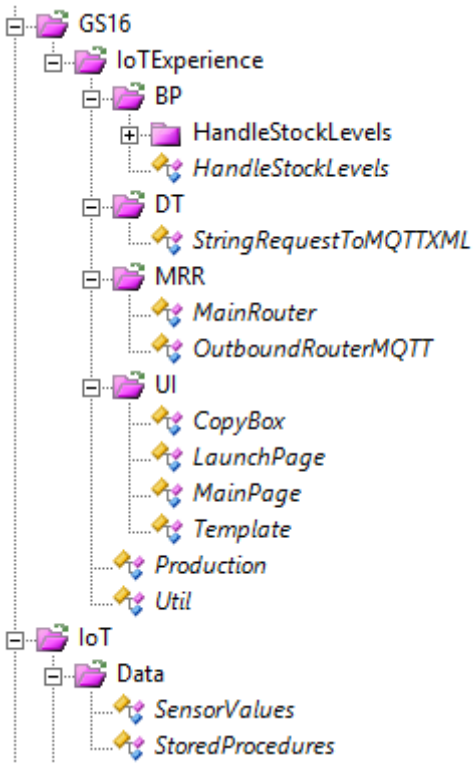

Figure 6: Ensemble classes

InterSystems Corporation
World Headquarters
One Memorial Drive
Cambridge, MA 02142-1356
Tel: +1 617 621 0600
InterSystems.com

InterSystems GmbH
Hilpertstraße 20a
64295 Darmstadt
Tel: +49 (0)6151 1747 0
InterSystems.de