Virtual Infrastructure Operating System (VIOS)

EF-PI Smart Meter driver and manager

Status:	Concept
Version:	1.1
Date:	10-6-2015

Contents

1 Smart Meter: P1 port

This document describes the integration of the Smart Meter within EF-pi.

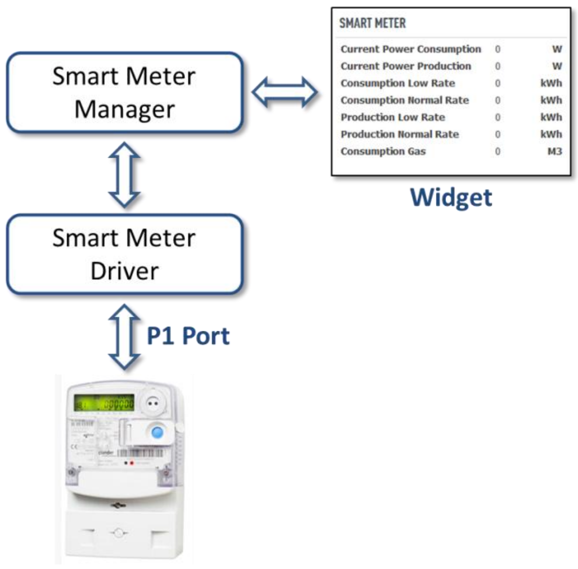

Smart Meter Manager
SMART METER
Current Power Consumption 0 W
Current Power Production 0 W
Consumption Low Rate 0 kWh
Consumption Normal Rate 0 kWh
Production Low Rate 0 kWh
Production Normal Rate 0 kWh
Consumption Gas 0 M3
Widget
Smart Meter Driver
P1 Port
Figure 1: Overview of the Smart Meter components

Figure 1 provides an overview of all the smart meter components. The actual smart meter itself has a serial port that provides measurement updates every 10 seconds: the P1 port.

The Smart Meter driver connects to the serial port of the P1 port and parses the datagrams that are being sent over this connection and makes them available in the form of a Java interface.

The Smart Meter manager translates the java methods of the driver in control spaces. It only takes electricity in consideration, gas values are being ignored. The manager also has a widget that display current information about the smart meter measurements.

1.1 Smart Meter resource driver

The resource driver communicates with the P1 port of the smart meter via a serial connection. Because there is no standard Java API to communicate with serial port, an external library is being used: the RXTX library. This library can be found at http://rxtx.qbang.org/wiki/index.php/Main_Page.

1.1.1 Configuration

The code snippet below shows the configuration of the smart meter resource driver.

```
    interface Config {
        @Meta.AD(deflt = "smartmeter")
        String resourceId();

        @Meta.AD(deflt = "COM11")
        String getcomport();
    }
```

The most important parameter is the COM port that the P1 port of the smart meter is connected to.

1.1.2 SmartMeterState

The smart meter resource driver communicates its state to the resource manager via the SmartMeterState interface, which is listed below.

```
public interface SmartMeterState extends ResourceState {

    public SmartMeterMeasurement getMeasurement();

    public Date getTimeStamp();

    public BigDecimal getCurrentPowerConsumptionW();

    public BigDecimal getCurrentPowerProductionW();

    public BigDecimal getElectricityConsumptionLowRateKwh();

    public BigDecimal getElectricityConsumptionNormalRateKwh();

    public BigDecimal getElectricityProductionLowRateKwh();

    public BigDecimal getElectricityProductionNormalRateKwh();

    public BigDecimal getGasConsumptionM3();

}
```

There are methods that provide the current electricity power consumption or production, the total amount of electricity consumption or production, both during low and normal rates and the current gas consumption in cubic meters.

1.2 Smart Meter resource manager

1.2.1 Configuration

```
  interface Config {
    @Meta.AD(deflt = "smartmeter")
    String resourceId();
  }
```

This code snippet shows the configuration of the smart meter manager. It only has a resourceId.

1.2.2 Registration

```
  protected List<? extends ResourceMessage> startRegistration(
      SmartMeterState state) {
    lastState = state;
    UncontrolledRegistration registration = new
UncontrolledRegistration(config.resourceId(), timeService.getTime(),
Measure.valueOf(0, SI.SECOND), CommoditySet.onlyElectricity, null);

    return Arrays.asList(registration);
  }
```

The registration is very straightforward. The manager registers itself as an uncontrolled device which only has information about electricity.

1.2.3 Control Space update

```
  protected List<? extends ResourceMessage> updatedState(SmartMeterState state) {
    Measurable<Power> currentUsage;

    if (state.getCurrentPowerConsumptionW().intValueExact() > 0) {
      currentUsage =
DecimalMeasure.valueOf(state.getCurrentPowerConsumptionW().intValueExact(),
SI.WATT);
    } else {
      currentUsage = DecimalMeasure.valueOf(-1 *
state.getCurrentPowerProductionW().intValueExact(), SI.WATT);
    }

        CommodityMeasurables measurables =
CommodityMeasurables.electricity(currentUsage);
        UncontrolledUpdate update = new UncontrolledMeasurement(config.resourceId(),

timeService.getTime(),

timeService.getTime(),
                                                                measurables);
        return Arrays.asList(update);
    }
```

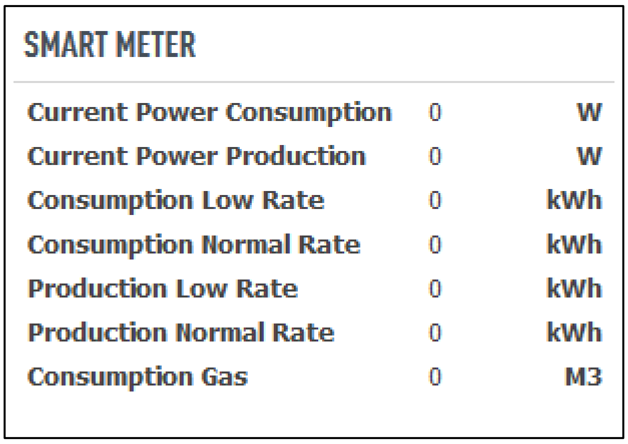
Within the EFI consumption and production are being communicated in a single parameter. The convention is that positive values denote consumption and negative values production. The Smart Meter however has a separate parameter for consumption and one for production.

Whenever a new SmartMeterState object is received by the manager it checks whether the consumption is larger than zero. If so this value will be copied to the control space update. If not, there must be production and the production value will be multiplied by -1 to obtain the correct EFI value.

1.2.4 Allocations

The Smart Manager does not support allocations.

1.2.5 Widget



SMART METER

Current Power Consumption	0	W
Current Power Production	0	W
Consumption Low Rate	0	kWh
Consumption Normal Rate	0	kWh
Production Low Rate	0	kWh
Production Normal Rate	0	kWh
Consumption Gas	0	M3

Figure 2: The Smart Meter widget

Figure 2 shows the widget for the smart meter. It display the information that is contained in the SmartMeterState (see section 1.1.2).