O-MI/O-DF Wrapper

Manik Madhikermi

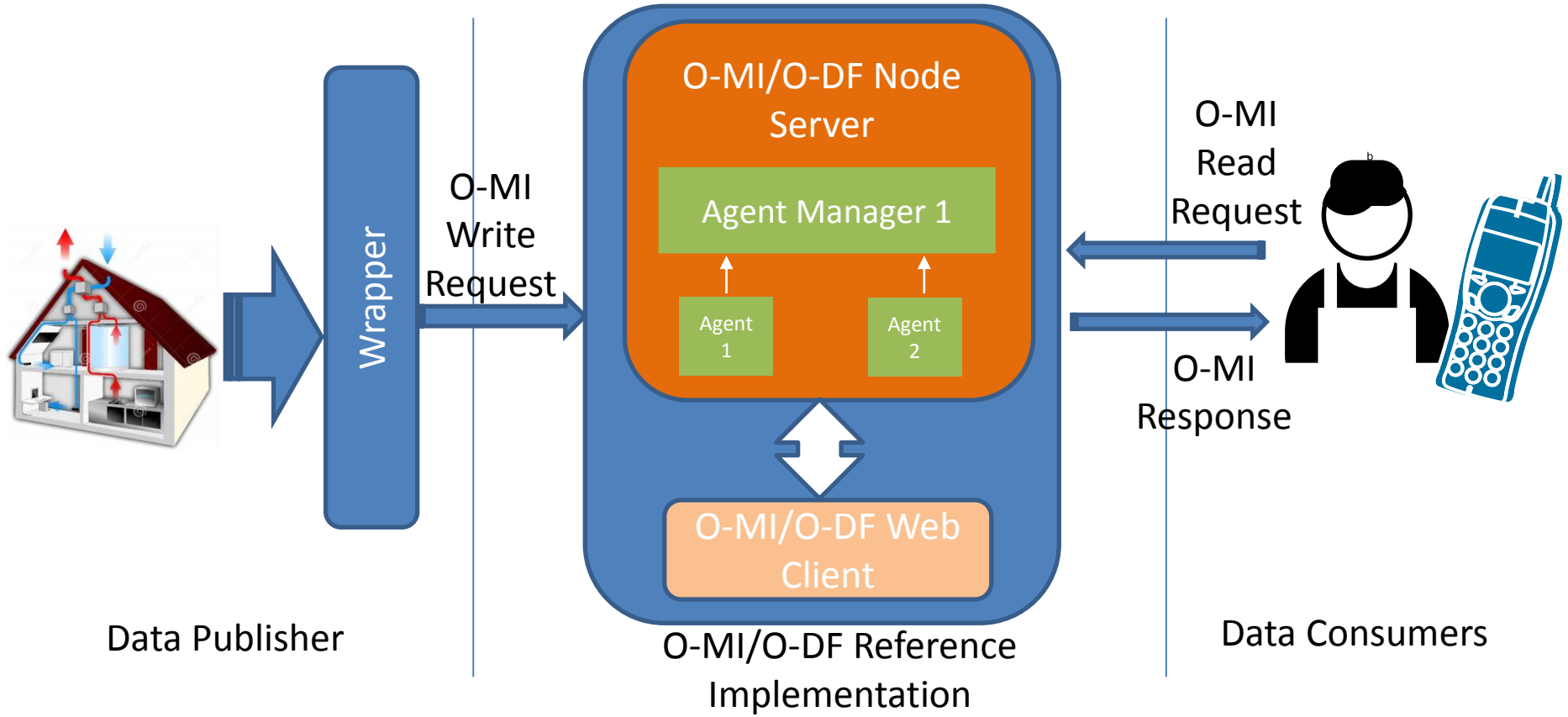
Aalto University

Introduction

- Piece of code that generate O-MI/O-DF message to publish data.
- Combine data from different source or system with different/same protocol into single O-DF structure for publishing.
- Encapsulate underlying lower protocol for data consumer
  - e.g. 1-Wire, third party system, KNX etc.
- Wrapper can be written in any programming or scripting language.
  - e.g.: Shell script, Java, Python, C etc.

.

Smart Home Scenario



Smart Home Scenario Description

- Publish smart home data to reference implementation
- Smart Home equipped with
  - Air Handling unit (AHU) with different sensor installed in its sub-units (proprietary protocol).
  - Bed room and Living room with sensors measuring Temperature and Humidity of corresponding room.
- O-MI/O-DF Wrapper reads and integrate these sensors value into single O-DF structure and send it to reference implementation using O-MI write messaging interface.
- Developer test the system using web-client.
- Data consumer (e.g. AHU maintenance company) be interested in AHU and sensors data for developing predictive maintenance methods. They can subscribe the object of interest.

Wrapper Code in Bash Script(test_omi_write.sh)

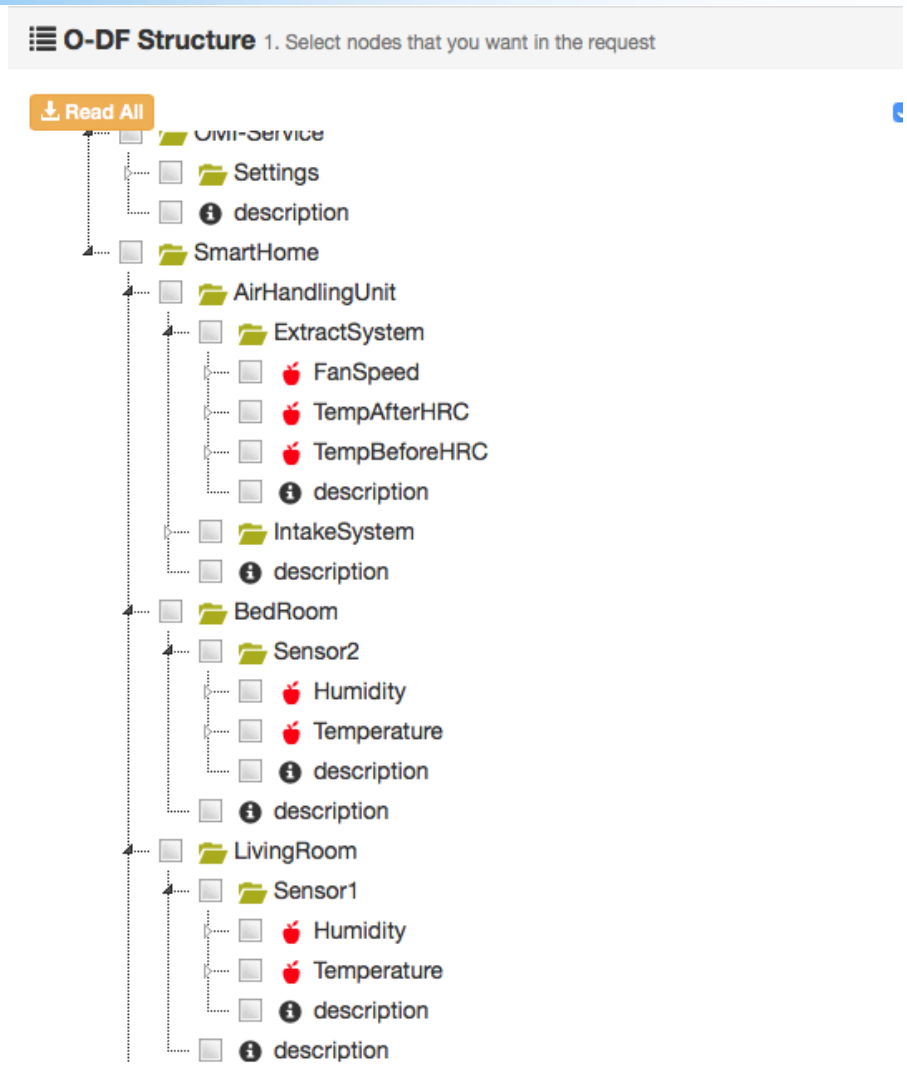
```
#!/bin/bash
#
# Testing genaration of O-MI/O-DF write message
#
# Kary Framling, 21 may 2015
# Updated Manik Madhikermi 20 Jan 2019,

#=============================================================
# Functions must be declared before being used.

# Create InfoItem  with given name and value
function odfInfoItem () {
    echo "<InfoItem name=\"$1\"><value unixTime=\"`date +%s`\">$2</value></InfoItem>"
}

# Create Object with given name and its InfoItems and values
function odfObject () {
    echo "<Object>"
    echo "<id>$1</id>"
    ii_list=$1"_InfoItems"
    set -f
    ii_array=(${!ii_list})
    value_list=$1"_values"
    set -f
    value_array=(${!value_list})
    obj_list=$1"_Objects"
    set -f
    obj_array=(${!obj_list})
    for ((ix=0;ix<${#ii_array[*]};ix+=1)); do
        echo "$(odfInfoItem ${ii_array[$ix]} ${value_array[$ix]})"
    done
    for ((ix=0;ix<${#obj_array[*]};ix+=1)); do
        echo "$(odfObject ${obj_array[$ix]})"
    done
    echo "</Object>"
}

# Create Objects from list of Object names (given as argument $1) or from top-level "Objects" variable if no argument
function odfObjects () {
    if [[ -z "$1" ]]
    then o=${Objects}
    else o=${1}
    fi
        echo "<Objects>"
    for x in ${o}; do
        echo "$(odfObject ${x})"
    done
    echo "</Objects>"
}

# Create a "write" O-MI message
function omiWrite () {
    echo "<omi:omiEnvelope xmlns:omi=\"omi.xsd\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"omi.xsd omi.xsd\" version=\"1.0\" ttl=\"-1\"><omi:write msgformat=\"odf\"><omi:msg xmlns=
    echo "$(odfObjects)"
    echo "</omi:msg></omi:write></omi:omiEnvelope>"
}

#=============================================================
# Get O-DF tree as variables
cd /Users/madhikm1/Desktop/bioTopeDeliverables/ShellScript/
source Create_odf.sh
   curlCmd="curl  --data '$(omiWrite)' -k 'http://localhost:8080'"
   eval $curlCmd
```

Wrapper Code in Bash Script (create_odf.sh)

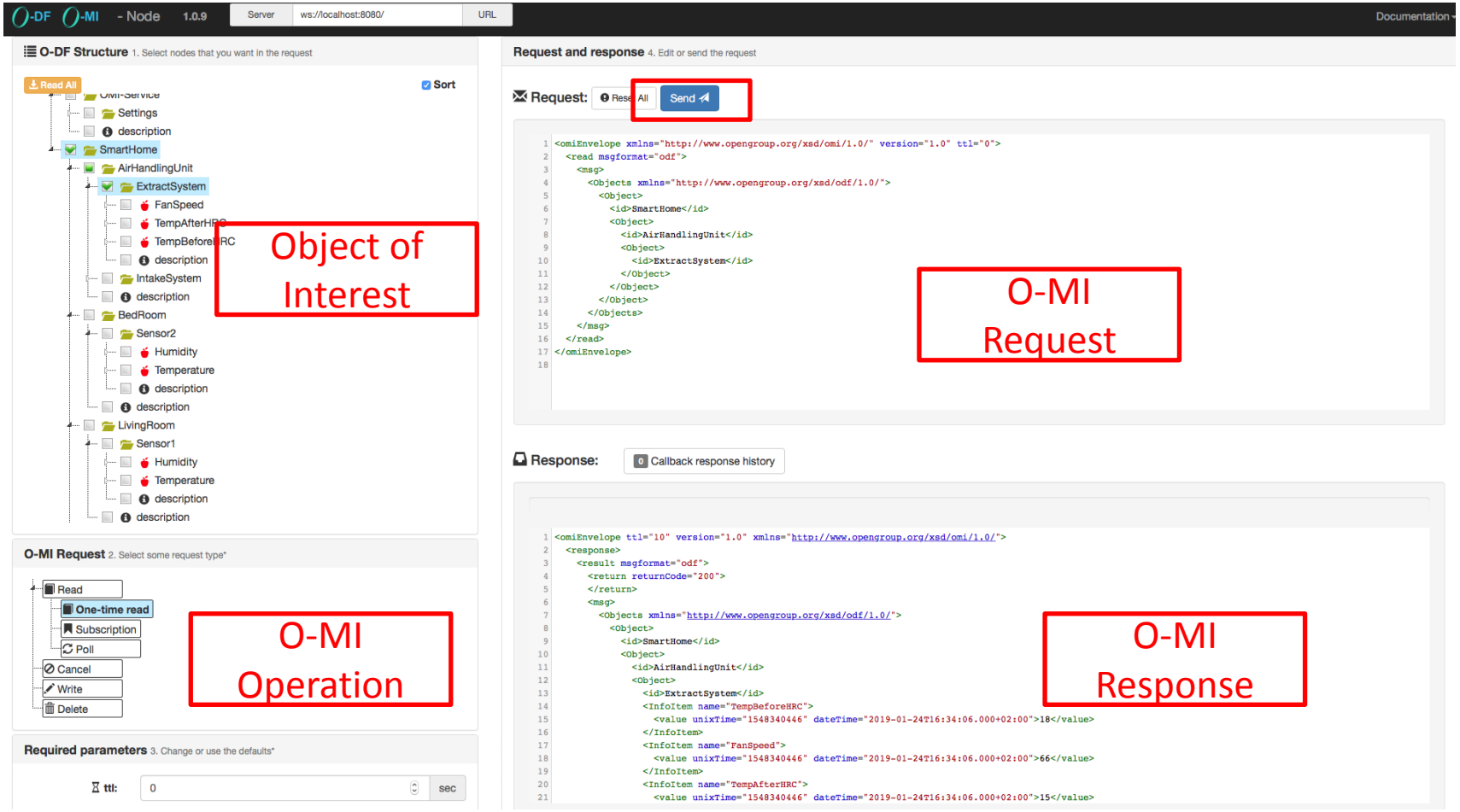
```
#!/bin/bash
#
# Create O-DF tree using "structurally named" variables. Use "source ./create_odf.sh" for calling this and
# making the variables available in the calling script.
#
# Kary Framling, 23 may 2015
#

function roll_die() {
FLOOR=$1;
CEILING=$2;
RANGE=$(($CEILING-$FLOOR+1));
RESULT=$RANDOM;
let "RESULT %= $RANGE";
RESULT=$(($RESULT+$FLOOR));



  # echo to screen
  echo $[ RESULT]

}

Objects="SmartHome"

SmartHome_Objects="LivingRoom BedRoom AirHandlingUnit"

LivingRoom_Objects="Sensor1"
BedRoom_Objects="Sensor2"
AirHandlingUnit_Objects="ExtractSystem IntakeSystem"

Sensor1_InfoItems="Temperature Humidity"
Sensor2_InfoItems="Temperature Humidity"
ExtractSystem_InfoItems="TempBeforeHRC TempAfterHRC FanSpeed"
IntakeSystem_InfoItems="TempBeforeHRC TempAfterHRC FanSpeed"

#Sensor1_values="`eval "cat /var/1wire/26.ACAC40010000/temperature"` `eval "cat /var/1wire/26.ACAC40010000/humidity"`"
#Sensor2_values="`eval "cat /var/1wire/26.0D4945010000/temperature"` `eval "cat /var/1wire/26.0D4945010000/humidity"`"
#Sensor3_values="`eval "cat /var/1wire/26.DB7E23010000/temperature"` `eval "cat /var/1wire/26.DB7E23010000/humidity"`"
#Sensor4_values="`eval "cat /var/1wire/26.1B4C45010000/temperature"` `eval "cat /var/1wire/26.1B4C45010000/humidity"`"

Sensor1_values="`roll_die 18 32` `roll_die 10 50`"
Sensor2_values="`roll_die 18 32` `roll_die 10 50`"
ExtractSystem_values="`roll_die 18 28` `roll_die 12 18` `roll_die 60 80`"
IntakeSystem_values="`roll_die 18 28` `roll_die 12 18` `roll_die 50 90`"
```

Write Request From Wrapper

```
<omi:omiEnvelope xmlns:omi="omi.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="omi.xsd omi.xsd" version="1.0" ttl="-1">
   <omi:write msgformat="odf">
      <omi:msg xmlns="odf.xsd" xsi:schemaLocation="odf.xsd odf.xsd">
         <Objects>
            <Object>
               <id>SmartHome</id>
               <Object>
                  <id>LivingRoom</id>
                  <Object>
                     <id>Sensor1</id>
                     <InfoItem name="Temperature">
                        <value unixTime="1548340446">32</value>
                     </InfoItem>
                     <InfoItem name="Humidity">
                        <value unixTime="1548340446">15</value>
                     </InfoItem>
                  </Object>
               </Object>
               <Object>
                  <id>BedRoom</id>
                  <Object>
                     <id>Sensor2</id>
                     <InfoItem name="Temperature">
                        <value unixTime="1548340446">25</value>
                     </InfoItem>
                     <InfoItem name="Humidity">
                        <value unixTime="1548340446">22</value>
                     </InfoItem>
                  </Object>
               </Object>
               <Object>
                  <id>AirHandlingUnit</id>
                  <Object>
                     <id>ExtractSystem</id>
                     <InfoItem name="TempBeforeHRC">
                        <value unixTime="1548340446">18</value>
                     </InfoItem>
                     <InfoItem name="TempAfterHRC">
                        <value unixTime="1548340446">15</value>
                     </InfoItem>
                     <InfoItem name="FanSpeed">
                        <value unixTime="1548340446">66</value>
                     </InfoItem>
                  </Object>
                  <Object>
                     <id>IntakeSystem</id>
                     <InfoItem name="TempBeforeHRC">
                        <value unixTime="1548340446">23</value>
                     </InfoItem>
                     <InfoItem name="TempAfterHRC">
                        <value unixTime="1548340446">17</value>
                     </InfoItem>
                     <InfoItem name="FanSpeed">
                        <value unixTime="1548340446">81</value>
                     </InfoItem>
                  </Object>
               </Object>
            </Object>
         </Objects>
      </omi:msg>
   </omi:write>
</omi:omiEnvelope>
```



Web-Client Sandbox



Steps to publish data with reference implementation

- Step 1: Deploy Reference Implementation
- Step 2: Develop Wrapper For publishing data
- Step 3: Send O-MI write request to create or update data periodically using wrapper
- Step 4: Use Web Client for to test other O-MI/O-DF functionality such as one time read, subscription, delete etc.

Note: Other system can also read/subscribe published data by sending read/subscription request.

Thank You