OpenSensorHub
Development Training
SensorML Sensor Description through API

Lab 5
Requirements

• Java Programming Language – Entry Level Experience

• Lab 4 Complete!
Adding SensorML Description Programmatically

Using API
API: AbstractSensorModule

- Class providing default implementation of common sensor API methods. This can be used as the base for most sensor driver implementations as it generates defaults for the following:
- A random Unique ID using a UUID (the same is used between restarts)
- A short XML ID
- A default SensorML description including IDs, temporal validity, I/Os and position (location + orientation) if the sensor configuration provides static location and/or orientation
- A feature of interest if the sensor configuration provides static location
- All of these items can be overridden by derived classes. It also provides helper methods to implement automatic reconnection.
Sensor extends AbstractSensorModule
updateSensorDescription

• This method should be called whenever the sensor description needs to be regenerated.

• Default implementation reads the base description from the SensorML file if provided and then appends the unique sensor identifier, time validity and the description of all registered outputs and control inputs.

• Override method to provide sensor description programmatically
Navigate to Sensor(.java) and Open
Override updateSensorDescription
Building the Description

• All sensor description operations shall be performed within

```
synchronized (sensorDescLock) {
    super.updateSensorDescription();
}
```

• Make sure to call method on parent via `super`
Update Sensor Description if not Already Set
Textual Description – Overview of Sensor

```java
@Override
protected void updateSensorDescription() {
    synchronized (sensorDescLock) {
        super.updateSensorDescription();
        if (!sensorDescription.isSetDescription()) {
            sensorDescription.setDescription("A simulated sensor for training purposes, " + "demonstrating how to build a driver.");
        }
    }
```
SMLHelper API Builders

• Provides methods to build systems and processes compliant with SensorML
  • public SimpleProcessBuilder createSimpleProcess()
    • A builder to create a new SimpleProcess
  
  • public AggregateProcessBuilder createAggregateProcess()
    • A builder to create a new AggregateProcess
  
  • public PhysicalComponentBuilder createPhysicalComponent()
    • A builder to create a new PhysicalComponent
  
  • public PhysicalSystemBuilder createPhysicalSystem()
    • A builder to create a new PhysicalSystem
SMLHelper Editors

• Also Provides methods to edit systems and processes descriptions
  • public SimpleProcessBuilder edit(SimpleProcess sml)
    • Helper method to edit a SimpleProcess description in-place using a builder

  • public AggregateProcessBuilder edit(AggregateProcess sml)
    • Helper method to edit a AggregateProcess description in-place using a builder

  • public PhysicalComponentBuilder edit(PhysicalComponent sml)
    • Helper method to edit a PhysicalComponent description in-place using a builder

  • public PhysicalSystemBuilder edit(PhysicalSystem sml)
    • Helper method to edit a PhysicalSystem description in-place using a builder
Use SMLHelper Edit Functionality

Create instance of SMLHelper and begin editing (note the sensorDescription is cast to a PhysicalSystem):

```java
SMLHelper smlHelper = new SMLHelper();
smlHelper.edit((PhysicalSystem)sensorDescription);
```
Adding an Identifier - SerialNumber

SMLHelper identifiers provides several methods to create identifiers.
Add Classifier(s)

SMLHelper identifiers provides methods to create classifiers

Classifiers aid in identifying or defining the sensor type and can include ontological definitions via URL strings
Adding Characteristics

• SMLHelper through the characteristics member allows various characteristics to be defined or added to the sensor description
• These are not required but should be specified for robustness when appropriate
• “uom” refers to “unit of measure”
Example Characteristics

A sampling of operating characteristics. Typically such characteristics are defined by the original equipment manufacturer (OEM). Characteristics can also include conditions, such as a temperature range.

```javascript
.addCharacteristicList("operating_specs", sml.characteristics.operatingCharacteristics()
    .add("voltage", sml.characteristics.operatingVoltageRange(3.3, 5., "V"))
    .add("temperature", sml.conditions.temperatureRange(-10., 75., "Cel")))
```
Adding Characteristics...

```java
if (!sensorDescription.isSetDescription()) {
    sensorDescription.setDescription("A simulated sensor for training purposes, " +
    "demonstrating how to build a driver.");

    SMLHelper smlHelper = new SMLHelper();
    smlHelper.edit(PhysicalSystem.sensorDescription)
        .addIdentifier(smlHelper.identifiers.serialNumber(value: "1234567890"))
        .addClassifier(smlHelper.classifiers.sensorType(value: "Simulated Sensor Platform"));

    .addCharacteristicList(name: "operating_specs", smlHelper.characteristics.operatingCharacteristics()
        .add(name: "voltage", smlHelper.characteristics.operatingVoltageRange(min: 3.3, max: 5.5, uom: "V"))
        .add(name: "temperature", smlHelper.conditions.temperatureRange(min: -10.0, max: 75.0, uom: "Cel")));
```
Adding Capabilities

• SMLHelper through the capabilities member allows various capabilities to be defined or added to the sensor description.

• These are not required but should be specified for robustness when appropriate.

• “uom” refers to “unit of measure”
Adding Capabilities...

```javascript
.addClassifier(smlHelper.classifiers.sensorType(value: "Simulated sensor Platform"))
.addCharacteristicList(name: "operating specs", smlHelper.characteristics.operatingCharacteristics()
  .add(name: "voltage", smlHelper.characteristics.operatingVoltageRange(min: 3.3, max: 5.0, unit: "V"))
  .add(name: "temperature", smlHelper.characteristics.temperatureRange(min: 10, max: 70, unit: "F")))

.addCapabilityList(name: "capabilities", smlHelper.capabilities.systemCapabilities()
  .add(name: "update_rate", smlHelper.capabilities.reportingFrequency(1.0))
  .add(name: "accuracy", smlHelper.capabilities.absoluteAccuracy(value: 2.5, unit: "m"))
  .add(name: "TFF_cold", smlHelper.createQuantity()
    .definition(SWHEntity.getDoubleProperty(propName: "Time_to_first_fix")
      .label("Cold Start TFF")
      .description("Time to first fix on cold start")
      . uomCode("s")
      . value(120))
  .add(name: "TFF_warm", smlHelper.createQuantity()
    .definition(SWHEntity.getDoubleProperty(propName: "Time_to_first_fix")
      .label("Warm Start TFF")
      .description("Time to first fix on warm start")
      . uomCode("s")
      . value(30))
  .add(name: "TFF_hot", smlHelper.createQuantity()
    .definition(SWHEntity.getDoubleProperty(propName: "Time_to_first_fix")
      .label("Hot Start TFF")
      .description("Time to first fix on hot start")
      . uomCode("s")
      . value(3))
  .add(name: "battery_life", smlHelper.characteristics.batteryLifetime(value: 72, unit: "h")));
```
Adding Other Details

• While editing the sensorDescription:
  • addComponentLocation
    • When the sensor is at a fixed location
    • Can be set through configuration
  • addComponent
    • Adds a component, an embedded PhysicalSystem that is part of the sensor, as in a sensor platform
  • addLocalReferenceFrame
    • Reference frame for orientation of component(s)
Example Reference Frame

```java
// Reference Frame
SpatialFrame localRefFrame = new SpatialFrameImpl();
localRefFrame.setId("LOCAL_FRAME");
localRefFrame
  .setOrigin("Center of the Kinect Device facet containing apertures for emitter and sensors");
localRefFrame.addAxis("x",
  "The X axis is in the plane of the facet containing the apertures for emitter and sensors and points to the right");
localRefFrame.addAxis("y",
  "The Y axis is in the plane of the facet containing the apertures for emitter and sensors and points up");
localRefFrame.addAxis("z",
  "The Z axis points towards the outside of the facet containing the apertures for emitter and sensors");
((PhysicalSystem) sensorDescription).addLocalReferenceFrame(localRefFrame);
```

One or more reference frames can be added to a description, for example in the case of a sensor platform containing N sensors whose spatial reference frame is needed to fully describe the sensor
Adding Identifiers

Unique Id & XML Id
The template sensor class provides placeholders [URN] and [XML-PREFIX] that need to be changed. The identifiers are composed of a prefix and a suffix, the Uniqueld takes the form of a URN while the XmlID is a text value. If no suffix is specified one is generated automatically, however, in our case we will retrieve suffix from config, which will be discussed in later lab.