# How to get information about sensors in Java ME

## Overview

The Mobile Sensor API (JSR-256) is now supported in S60 5th Edition devices. The Nokia N97 mobile computer is the first device to have the API as a built-in feature. In addition, the API can be added to the Nokia 5800 XpressMusic using a add-on, which can be downloaded from Nokia Developer here.

## Working with the Mobile Sensor API

### Determining the available sensors

Various mobile devices are equipped with different types of sensors. The first step for any application is therefore to find out which sensors are available for use with the Mobile Sensor API. The `SensorManager.findSensors()` method enables applications to do this. This method returns an array of `SensorInfo` objects, which contain information about the sensors.

The SensorInfo array can be created like this:

```
// if quantity and contextType are null, information about every sensor in the device is
returned
SensorInfo[] infos = SensorManager.findSensors(String quantity, String contextType);
```

Now, for example, a readable description of the sensors can be listed using the following code:

```
    for (int i = 0; i < length; i ++) {
        System.out.println("Sensor #" + (i+1) + ": Description: " +
infos[i].getDescription());
    }
```

### Working example

Here is a simple SensorInfoMIDlet for listing all the sensors and detailed information about them.

### Source code: SensorInfoMIDlet.java

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.MIDlet;
import javax.microedition.sensor.SensorInfo;
import javax.microedition.sensor.SensorManager;
import javax.microedition.sensor.MeasurementRange;
import javax.microedition.sensor.ChannelInfo;

public class SensorInfoMIDlet extends MIDlet implements CommandListener {
    private SensorInfo[] infos;
    private MeasurementRange[] ranges;
    private Form form;
    private Command exitCommand;

 public SensorInfoMIDlet() {
            form = new Form("Sensor Info");
            exitCommand = new Command("Exit", Command.EXIT, 1);
```


(C) Copyright Nokia 2014. All rights reserved.

```
            form.addCommand(exitCommand);
            form.setCommandListener(this);
            Display.getDisplay(this).setCurrent(form);
 }

 public void startApp() {
            listSensors();
 }

 public void destroyApp(boolean unconditional) {}

 public void pauseApp() {}

    /**
     * Lists all the sensors in the device and the following details:<BR>
     * - sensor quantity<BR>
     * - readable description of the sensor<BR>
     * - the context type of the sensor<BR>
     * - the model of the sensor<BR>
     * - the URL needed to open a SensorConnection<BR>
     * - the datatype of the sensor<BR>
     * - the measurement range: smallest and largest values and the resolution<BR>
     * - channel info: the datatypes of the channels, their names, scales and units<BR>
     */
    private void listSensors() {
 infos = SensorManager.findSensors(null, null);
 if (infos.length==0) return;
        int length = infos.length;
        int datatypes[] = new int[length];
        for (int i = 0; i < length; i ++) {
            datatypes[i] = infos[i].getChannelInfos()[0].getDataType();
            ranges = infos[i].getChannelInfos()[0].getMeasurementRanges();
            addText("Sensor #" + (i+1) + ": Quantity: " + infos[i].getQuantity());
            addText("Sensor #" + (i+1) + ": Description: " + infos[i].getDescription());
            addText("Sensor #" + (i+1) + ": ContextType: " + infos[i].getContextType());
            addText("Sensor #" + (i+1) + ": Model: " + infos[i].getModel());
            addText("Sensor #" + (i+1) + ": Url: " + infos[i].getUrl());
            String datatype = "";
            if (datatypes[i] == 1) datatype = "TYPE_DOUBLE";
//ChannelInfo.TYPE_DOUBLE = 1
            else if (datatypes[i] == 2) datatype = "TYPE_INT";
//ChannelInfo.TYPE_INT = 2
            else if (datatypes[i] == 4) datatype = "TYPE_OBJECT";
//ChannelInfo.TYPE_OBJECT = 4
            addText("Sensor #" + (i+1) + ": DataType: " + datatype);
            addText("Sensor #" + (i+1) + ": MeasurementRange, smallest value: " +
ranges[0].getSmallestValue());
            addText("Sensor #" + (i+1) + ": MeasurementRange, largest value: " +
ranges[0].getLargestValue());
            addText("Sensor #" + (i+1) + ": MeasurementRange, resolution: " +
ranges[0].getResolution());
            SensorInfo sensorinfo = infos[i];
            ChannelInfo channelInfo[] = sensorinfo.getChannelInfos();
            for(int j = 0; j < channelInfo.length; j++) {
                ChannelInfo channelinfo = channelInfo[j];
                addText("Sensor #" + (i+1) + ": " + (j+1) + ". channel, accuracy:" +
channelinfo.getAccuracy());
```


(C) Copyright Nokia 2014. All rights reserved.

```
                int d_type = channelinfo.getDataType();
                if (d_type == 1) datatype = "TYPE_DOUBLE";
//ChannelInfo.TYPE_DOUBLE = 1
                else if (d_type == 2) datatype = "TYPE_INT";      //ChannelInfo.TYPE_INT
= 2
                else if (d_type == 4) datatype = "TYPE_OBJECT";
//ChannelInfo.TYPE_OBJECT = 4
                addText("Sensor #" + (i+1) + ": " + (j+1) + ". channel, data type: " +
datatype);
                addText("Sensor #" + (i+1) + ": " + (j+1) + ". channel, name: " +
channelinfo.getName());
                int scale = channelinfo.getScale();
                String scaleString = "";
                if (scale == 0) scaleString = "scaling not needed";
                else scaleString = "" + scale;
                addText("Sensor #" + (i+1) + ": " + (j+1) + ". channel, scale: " +
scaleString);
                addText("Sensor #" + (i+1) + ": " + (j+1) + ". channel, unit: " +
channelinfo.getUnit());
            }
        }
    }

    private void addText(String text) {
        form.append(text + "\n");
        System.out.println(text);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) notifyDestroyed();
    }
}
```

## Jad and jar files

The **SensorInfoMIDlet.jad** and **SensorInfoMIDlet.jar** files can be downloaded from SensorInfoMIDlet.zip.

## Reviewing the output

The SensorInfoMIDlet prints the information generated to standard output. The MIDPLogger application can be used to view the output and saving it to a file.

If run on a Nokia N97 device - which supports accelerometer (actually two of them), battery charge sensor, charger state sensor, and network field intensity sensor - the output for an accelerometer sensor looks like this:

```
Sensor #1: Quantity: acceleration
Sensor #1: Description: acceleration sensor has channels axis_x, axis_y, and axis_z that
measure x, y, and z axis accelerations; used, for example, to get the orientation of the
device. Always on
Sensor #1: ContextType: user
Sensor #1: Model: Nokia
Sensor #1: Url: sensor:acceleration;contextType=user;model=Nokia;location=NoLoc
Sensor #1: DataType: TYPE_DOUBLE
Sensor #1: MeasurementRange, smallest value: -19.62
Sensor #1: MeasurementRange, largest value: 19.62
Sensor #1: MeasurementRange, resolution: 0.15328125
Sensor #1:, 1. channel, accuracy:0.1
```


(C) Copyright Nokia 2014. All rights reserved.

```
Sensor #1:, 1. channel, data type: TYPE_DOUBLE
Sensor #1:, 1. channel, name: axis_x
Sensor #1:, 1. channel, scale: 0
Sensor #1:, 1. channel, unit: m/s^2
Sensor #1:, 2. channel, accuracy:0.1
Sensor #1:, 2. channel, data type: TYPE_DOUBLE
Sensor #1:, 2. channel, name: axis_y
Sensor #1:, 2. channel, scale: 0
Sensor #1:, 2. channel, unit: m/s^2
Sensor #1:, 3. channel, accuracy:0.1
Sensor #1:, 3. channel, data type: TYPE_DOUBLE
Sensor #1:, 3. channel, name: axis_z
Sensor #1:, 3. channel, scale: 0
Sensor #1:, 3. channel, unit: m/s^2
```

# Development tools

## S60 SDKs

The Nokia N97 SDK provides support for developing with the Mobile Sensor API. The latest version of the SDK can be downloaded from Nokia Developer here.

## Adding support to NetBeans

Here are the instructions on how to add the RI to NetBeans as a platform:

- download "RI Binary for JSR-256 Mobile Sensor API 1.0" from the Nokia Developer website here.
- unzip the package to a suitable folder.
- In NetBeans, from the menu bar select **Tools** > **Java Platforms** > **Add platforms** > **Java ME MIDP Platform Emulator** > **Next**. After the search click **Find More Java ME Platform Folders** and choose the folder where the unzipped RI package was saved.
- Select a platform, such as C:\Users\JSR_256_RI_1_0\Nokia_Prototype_SDK_2_0\devices\Prototype_2_0_S60_MIDP_Emulator, and click **Next**.
- In the **Detected Platforms** give more descriptive name to the platform, if required, then click **Finish**.

# See also

- Mobile Sensor API (JSR-256) beta add-on for Nokia 5800 XpressMusic
- Nokia N97 SDK 0.5
- Mobile Sensor API (JSR-256) javadoc documentation and RI Binary
- Video of using Mobile Sensor API for controlling RC car in Nokia Developer Summit 2009 in Monaco
- How to get accelerator sensor values in Java ME
- How to use sensors in Java ME


(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.