

Original: [How to get information about sensors in Java ME](#) (Inglês)

## Contents

- [1 Visão geral](#)
- [2 Trabalhando com a API de sensores](#)
  - ◆ [2.1 Determinando os sensores disponíveis](#)
  - ◆ [2.2 Exemplo de execução](#)
    - ◇ [2.2.1 Código fonte: SensorInfoMIDlet.java](#)
    - ◇ [2.2.2 Arquivos jad and jar](#)
    - ◇ [2.2.3 Visualizando a saída](#)
- [3 Ferramentas de desenvolvimento](#)
  - ◆ [3.1 S60 SDKs](#)
  - ◆ [3.2 Adicionando suporte ao NetBeans](#)
- [4 Veja também](#)

## Visão geral

A API de sensores (*Mobile Sensor API*) (JSR-256) é agora oficialmente suportada em dispositivos S60 5ª edição. O dispositivo N97 será o primeiro dispositivo a vir com esta API nativa no sistema. Além disso, esta API pode ser adicionado ao dispositivo Nokia 5800 XpressMusic usando um *add-on*, que pode ser baixado a partir do Forum Nokia [here](#).

## Trabalhando com a API de sensores

### Determinando os sensores disponíveis

Vários dispositivos móveis são equipados com diferentes tipos de sensores. A primeira etapa de qualquer aplicação é portanto definir quais sensores estão disponíveis para uso através da API de sensores. O método **SensorManager.findSensors()** possibilita que aplicações realizem tal tarefa. Este método retorna um *array* de objetos *SensorInfo*, que contém as informações sobre os sensores.

O *array SensorInfo* pode ser criado como segue:

```
// Se as variáveis quantity e contextType possuem referências 'null', informações sobre todos os  
SensorInfo[] infos = SensorManager.findSensors(String quantity, String contextType);
```

No código abaixo, por exemplo, é apresentado uma maneira de recuperar uma descrição a respeito de cada sensor disponível:

```
for (int i = 0; i < length; i++) {
```

## Como recuperar informações de sensores em Java ME

```
        System.out.println("Sensor #" + (i+1) + ": Description: " + infos[i].getDescription());
    }
}
```

### Exemplo de execução

Nesta seção apresenta-se um simples exemplo para listar todos os sensores e as descrições associadas a cada um deles.

### Código fonte: 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);
        form.addCommand(exitCommand);
        form.setCommandListener(this);
        Display.getDisplay(this).setCurrent(form);
    }

    public void startApp() {
        listSensors();
    }

    public void destroyApp(boolean unconditional) {}

    public void pauseApp() {}

    /**
     * Lista todos os sensores no dispositivo e os seguintes detalhes:<BR>
     * - quantidade de sensores<BR>
     * - descrição que pode ser lida do sensor<BR>
     * - o tipo de contexto do sensor<BR>
     * - modelo do sensor<BR>
     * - a URL necessária para abrir uma conexão SensorConnection<BR>
     * - o tipo de dado do sensor<BR>
     * - a escala de medição: valores mais baixo e mais alto e a resolução<BR>
     * - informação do canal (channel): os tipos de dados do canal, seus nomes, escalas e unidade
     */
    private void listSensors() {
        = SensorManager.findSensors(null, null);
        if (infos.length==0) return;
        int length = infos.length;
    }
}
```

## Como recuperar informações de sensores em Java ME

```
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"; //ChannelType.TYPE_DOUBLE = 1
    else if (datatypes[i] == 2) datatype = "TYPE_INT"; //ChannelType.TYPE_INT = 2
    else if (datatypes[i] == 4) datatype = "TYPE_OBJECT"; //ChannelType.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());
        int d_type = channelinfo.getDataType();
        if (d_type == 1) datatype = "TYPE_DOUBLE"; //ChannelType.TYPE_DOUBLE = 1
        else if (d_type == 2) datatype = "TYPE_INT"; //ChannelType.TYPE_INT = 2
        else if (d_type == 4) datatype = "TYPE_OBJECT"; //ChannelType.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();
}
}
```

### Arquivos jad and jar

O download dos arquivos *SensorInfoMIDlet.jad* e *SensorInfoMIDlet.jar* pode ser realizado de [SensorInfoMIDlet.zip](#).

### Visualizando a saída

A MIDlet *SensorInfoMIDlet* imprime as informações geradas para a saída padrão. A aplicação *MIDPLogger* pode ser usada para visualizara a saída e salva-a em um arquivo de log.

## Como recuperar informações de sensores em Java ME

Se a execução for realizada no N97 - que tem suporte a acelerometro, sensor de bateria, sensor de carregamento e intensidade da rede sem-fio - a saída para um sensor do acelerometro deve ser semelhante a seguir:

```
Sensor #1: Quantity: acceleration
Sensor #1: Description: acceleration sensor has channels axis_x, axis_y, and axis_z that measure
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
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
```

## Ferramentas de desenvolvimento

### S60 SDKs

A SDK para o dispositivo N97 possibilita o suporte ao desenvolvimento com a API de sensores. A última versão deste SDK pode ser baixado a partir do Forum Nokia [aqui](#).

### Adicionando suporte ao NetBeans

Aqui são apresentadas as instruções de como adicionar a RI ao Netbeand como uma plataforma:

- faça o download "RI Binary for JSR-256 Mobile Sensor API 1.0" do site do Forum Nokia [aqui](#).
- descompacte o pacote para o diretório desejado.
- no Netbeand, da barra de menus selecione **Tools > Java Platforms > Add platforms > Java ME MIDP Platform Emulator > Next**. Após a busca, clique **Find More Java ME Platform Folders** e escolha a pasta onde você descompactou o arquivo baixado.
- Selecione a plataforma, tais como  
C:\Users\JSR\_256\_RI\_1\_0\Nokia\_Prototype\_SDK\_2\_0\devices\Prototype\_2\_0\_S60\_MIDP\_Emulator,  
e clique **Next**.
- Na opção **Detected Platforms** dê uma descrição para a plataforma adicionada, se solicitado clique **Finish**.

## Veja também

- [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](#)