

Motion

Programming Guide

Version 2.0.0

Revision History

Version	Date	Description
2.0.0		

Table of Contents

1. OVERVIEW	4
1.1. ARCHITECTURE.....	4
1.2. CLASS DIAGRAM	5
1.3. SUPPORTED PLATFORMS.....	6
1.4. SUPPORTED FEATURES	6
1.5. COMPONENTS	6
1.6. INSTALLING THE PACKAGE FOR ECLIPSE	6
2. HELLO MOTION	8
3. USING THE SMOTION CLASS	9
3.1. USING THE INITIALIZE()METHOD	9
3.2. HANDLING SSDKUNSUPPORTEDEXCEPTION	9
3.3. CHECKING THE AVAILABILITY OF MOTION PACKAGE FEATURES	10
4. USING THE MOTION PACKAGE	11
4.1. RECEIVING DATA FROM THE MOTION PACKAGE	11
4.2. USING THE MOTION TYPES.....	12
4.2.1. <i>Using Call Motion</i>	12
4.2.2. <i>Using Pedometer</i>	14
4.2.3. <i>Tracking UserActivities</i>	17
4.2.4. <i>Using ActivityNotifications</i>	18
COPYRIGHT	19

1. Overview

Motion allows you to retrieve call and pedometer information in your application. The Motion package processes raw data from the device motion sensors to collect call and pedometer information.

You can use Motion to:

- Access information about calls initiated by device motion.
- Access pedometer information.

1.1. Architecture

The following figure shows the Motion architecture.

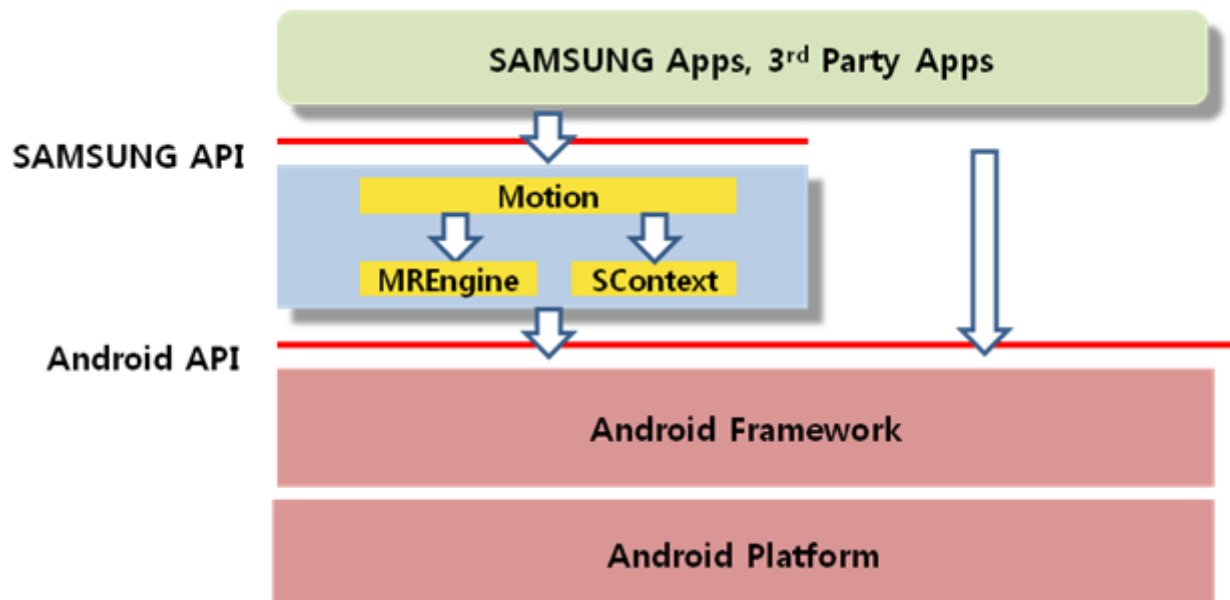


Figure 1: Motion architecture

The architecture consists of:

- **Applications:** One or more applications that use Motion.
- **Motion:** Motion components for managing specific call and pedometer events.
- **SContext:** Motion components for providing Motion with pedometer events.
- **MREngine:** Motion components for providing Motion with call motion events.

1.2. Class Diagram

The following figure shows the Motion classes and interfaces that you can use in your application.

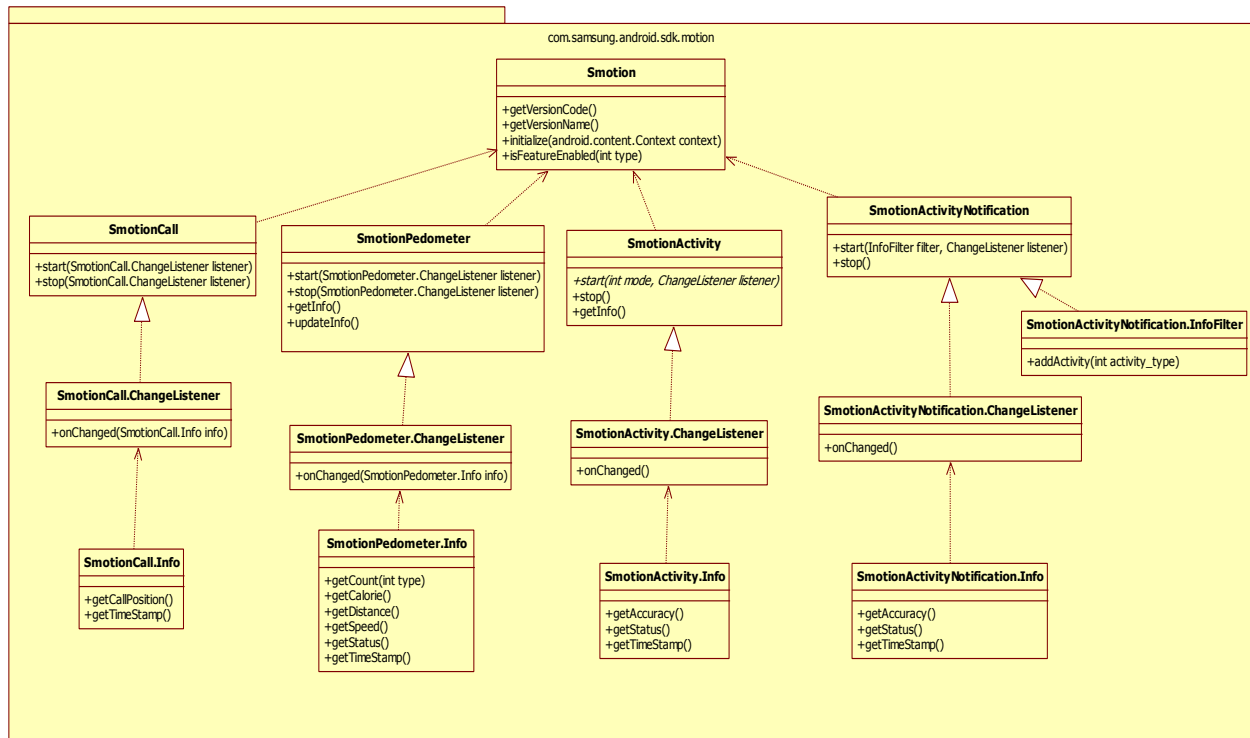


Figure 2: Motion classes and interfaces

The Motion classes and interfaces include:

- **Smotion:** Initializes the Motion package.
- **SmotionCall:** Provides access to call information for calls placed based on device motion.
- **SmotionPedometer:** Provides access to pedometer information.
- **SmotionActivity:** Provides access to activity information.
- **SmotionActivityNotification:** Provides access to activity notification information for specified activity events.
- **SmotionCall.Info:** Contains call motion information.
- **SmotionPedometer.Info:** Contains pedometer information.
- **SmotionActivity.Info:** Contains activity information.
- **SmotionActivityNotification.Info:** Contains activity notification information.
- **SmotionCall.ChangeListener:** Listens for call motion events.
- **SmotionPedometer.ChangeListener:** Listens for pedometer events.
- **SmotionActivity.ChangeListener:** Listens for activity events.
- **SmotionActivityNotification.Changelistener:** Listens for activity notification events.
- **SmotionActivityNotification.InfoFilter:** Creates specified activity notification actions.

1.3. Supported Platforms

- Android 4.3 (Android API level 18) or above support Motion.
- Android 4.4 (Android API level 19) or above support SmotionActivity and SmotionActivityNotification.

1.4. Supported Features

Motion supports the following features:

- Accessing information on calls placed based on device motion
- Accessing pedometer information
- Accessing activity information

1.5. Components

- Components:
 - motion-v2.0.0.jar
- Imported packages:
 - com.samsung.android.sdk.motion

1.6. Installing the Package for Eclipse

To install Motion for Eclipse:

1. Add the motion-v2.0.0.jar file to the libs folder in Eclipse.

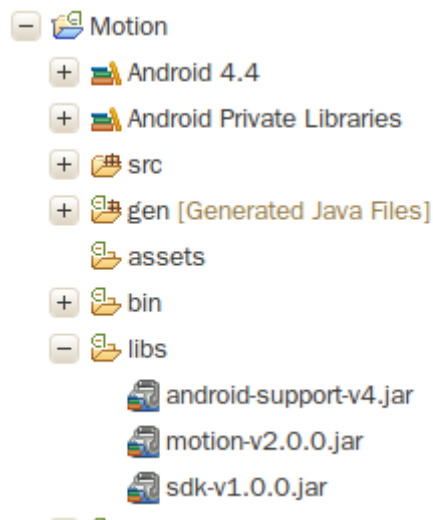


Figure 3: libs folder in Eclipse

2. Hello Motion

Hello Motion is a simple program that:

1. Creates `Smotion` and `SmotionPedometer` instances.
2. Implements, registers, and starts an `SmotionPedometer.ChangeListener` instance.
3. Handles motion events in the `ChangeListener.onChange()` method.
4. Stops the `ChangeListener` instance.

```
public class MainActivity extends Activity {
    private Smotion motion;
    private SmotionPedometer mPedometer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        motion = new Smotion();

        try {
            motion.initialize(this);
        } catch (IllegalArgumentException e) {
            //Error handling
        } catch (SdkUnsupportedException e) {
            //Error handling
        }

        // Create SmotionPedometer instance
        mPedometer = new SmotionPedometer(Looper.getMainLooper(), motion);
        // Start Pedometer
        mPedometer.start(changeListener);
    }

    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
        // Stop pedometer
        mPedometer.stop(changeListener);
    }

    private final SmotionPedometer.ChangeListener changeListener =
        new SmotionPedometer.ChangeListener() {

            @Override
            public void onChange(Info info) {
                // TODO Auto-generated method stub
                SmotionPedometer.InfoPedometerInfo = info;
                System.out.println("HelloMotion Pedometer");
            }

        };
}
```


3. Using the Smotion Class

The Smotion class provides the following methods:

- `initialize()` initializes Motion. You need to initialize the Motion package before you can use it. If the device does not support Motion, `SsdkUnsupportedException` is thrown.
- `getVersionCode()` gets the Motion version number as an integer.
- `getVersionName()` gets the Motion version name as a string.
- `isFeatureEnabled(int type)` checks if a Motion package feature is available on the device.

```
Smotion motion = new Smotion();
try {
    motion.initialize(this);
} catch (IllegalArgumentException) {
    //Error handling
} catch (SsdkUnsupportedException e) {
    //Error handling
}
```

3.1. Using the initialize() Method

The `Smotion.initialize()` method:

- Initializes the Motion package
- Checks if the device is a Samsung device
- Checks if the device supports the Motion package
- Checks if the Motion package libraries are installed on the device

```
void initialize(Context context) throws SsdkUnsupportedException
```

If the Motion package fails to initialize, the `initialize()` method throws an `SsdkUnsupportedException` exception. To find out the reason for the exception, check the exception message.

3.2. Handling SsdkUnsupportedException

If an `SsdkUnsupportedException` exception is thrown, check the exception message type using `SsdkUnsupportedException.getType()`.

The following types of exception messages are defined in the Smotion class:

- **VENDOR_NOT_SUPPORTED:** The device is not a Samsung device.
- **DEVICE_NOT_SUPPORTED:** The device does not support the Motion package.

3.3. Checking the Availability of Motion Package Features

You can check if a Motion package feature is supported on the device with the `isFeatureEnabled()` method. The feature types are defined in the `Smotion` class. Pass the feature type as a parameter when calling the `isFeatureEnabled()` method. The method returns a Boolean value that indicates the support for the feature on the device.

```
booleanisFeatureEnabled(int type);
```

The following types are defined in the `Smotion` class:

- `TYPE_CALL`
- `TYPE_PEDOMETER`
- `TYPE_PEDOMETER_WITH_UPDOWN_STEP`
- `TYPE_ACTIVITY`
- `TYPE_ACTIVITY_NOTIFICATION`

4. Using the Motion Package

This section describes how to use the Motion package in your application.

4.1. Receiving Data from the Motion Package

To initialize the Motion package and receive motion data:

1. Create anSmotion instance.
2. Pass the Smotion instance as a parameter to create anSmotionCall or SmotionPedometer instance.
3. Call start() to register aChangeListener instance for SmotionCall or SmotionPedometer. When Motion starts, the SmotionCall or SmotionPedometer instance receives a callback to the ChangeListener.
4. In the onChanged(Info info) method, handle the Motion events.
5. Call stop() to remove the ChangeListener instance.

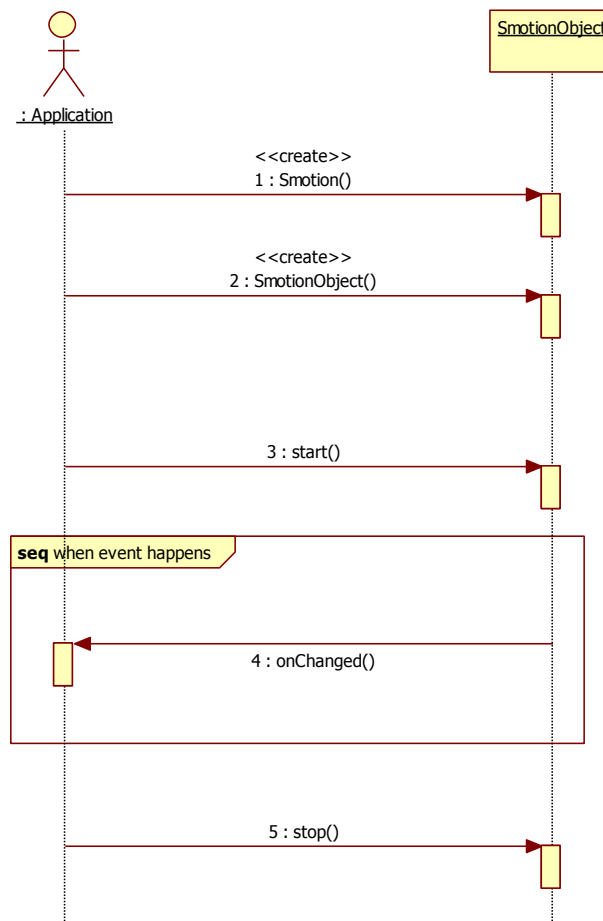


Figure 4: Receiving data from Motion

```

Smotionmotion;
SmotionPedometermPedometer;

// Initialize
Smotion motion = newSmotion();
try {
    motion.initialize(this);
} catch (IllegalArgumentException e) {
    //Error handling
} catch (SsdkUnsupportedException e) {
    //Error handling
}

// Create SmotionPedometer instance (SmotionCall follows the same pattern)
mPedometer = newSmotionPedometer(Looper.getMainLooper(),motion);

// Implement ChangeListener
privatefinalSmotionPedometer.ChangeListenerchangeListener =
    newSmotionPedometer.ChangeListener() {
        @Override
        publicvoidonChanged(Info info) {
            // TODO Auto-generated method stub
            SmotionPedometer.InfopedometerInfo = info;
        }
    };

// Add Smotion Listener
mPedometer.start(changeListener);

// Remove Smotion Listener
mPedometer.stop(changeListener);

```

4.2. Using the Motion Types

This section describes how to use the various motion types in your application.

4.2.1. Using Call Motion

The call motion recognizes the motion of first watching the device and then bringing the device up to your ear. It includes distinguishing between holding the device next to the left ear or the right ear.



Figure 5: Call motion in action

SmotionCall recognizes when you place the device next to your ear. When the call motion is recognized, the device dials the currently displayed on-screen Contact entry as soon as you place the device to your ear.

You can use the `getTimestamp()` method to get the timestamp to measure a duration by comparing it against another timestamp from the same process on the same device. The timestamp does not have a defined correspondence to wall clock times. The zero value is typically whenever the device was last booted. You can use `System.currentTimeMillis()` to get the current time.

You can use the `getCallPosition()` method to get which ear the device is next to.

```
public class MainActivity extends Activity {
    private Smotion motion;
    private SmotionCall mCall;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        mCall = new SmotionCall(Looper.getMainLooper(), motion);
        mCall.start(changeListener);
    }

    private final SmotionCall.ChangeListener changeListener =
        new SmotionCall.ChangeListener() {

        @Override
        public void onChanged(Info info) {
            // TODO Auto-generated method stub
            switch (info.getCallPosition()) {
                case SmotionCall.POSITION_LEFT:
                    break;
                case SmotionCall.POSITION_RIGHT:
                    break;
            }
        }
    };
}
```

4.2.2. Using Pedometer

SmotionPedometer offers you the following methods to get pedometer data:

- `getCount(int type)` gets the steps by type.
- `getSpeed()` gets the walking speed.
- `getDistance()` gets the distance moved.
- `getCalorie()` gets the calories burned.
- `getStatus()` gets the walking status.
- `updateInfo()` gets the accumulated data from the selected date.
- `getInfo()` gets the accumulated pedometer data by type from when the device was last booted.



Figure 6: Pedometer usage example

If you start walking or running after `SmotionPedometer.start()` is called, `SmotionPedometer` captures the number of steps, the speed, the distance covered, and the calories consumed.

You can use the `getTimeStamp()` method to get the timestamp to measure a duration by comparing it against another timestamp from the same process on the same device. The timestamp does not have a defined correspondence to wall clock times. The zero value is typically whenever the device was last booted. You can use `System.currentTimeMillis()` to get the current time.

Note

TYPE_PEDOMETER_WITH_UPDOWN_STEP

If a device does not support this feature,

- `getUpDownStepCount()` always returns 0.
- `getStepStatus()` does not return `PEDOMETER_STEP_STATUS_UP` or `PEDOMETER_STEP_STATUS_DOWN`.

Setting the user profile

The user height, weight, and sex can only be set in the SHealth application on the device. The profile settings are provided exclusively in SHealth because they can affect the measurement of speed, distance or calories. For more precise measurement of calories, distance or speed, encourage the users of your application to set their profile (height, weight and sex) in SHealth.

The following sample code shows how to use a listener to receive data.

```
public class MainActivity extends Activity {
    private Smotion motion;
    private SmotionPedometer mPedometer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        mPedometer = new SmotionPedometer(Looper.getMainLooper(), motion);

        mPedometer.start(changeListener);
    }

    @Override
    protected void onDestroy() {
        ...
    }

    // Update the pedometer data by listener callback.
    private final SmotionPedometer.ChangeListener changeListener =
        new SmotionPedometer.ChangeListener() {

            @Override
            public void onChanged(Info info) {
                // TODO Auto-generated method stub
                SmotionPedometer.InfoPedometerInfo = info;
                double calorie = pedometerInfo.getCalorie();
                double distance = pedometerInfo.getDistance();
                double speed = pedometerInfo.getSpeed();
                long count = pedometerInfo.getCount(SmotionPedometer.COUNT_TOTAL);
                int status = pedometerInfo.getStatus();
            }
        };
}
```

The following sample code shows how to receive data from the pedometer on an hourly basis by using the `SmotionPedometer.getInfo()` and `SmotionPedometer.updateInfo()` methods. You can use these methods to retrieve the latest `SmotionPedometer.Info` object for your application without waiting for a change event.

```
package com.android;

public class MainActivity extends Activity {
    private Smotion motion;
    private SmotionPedometer mPedometer;
    private SmotionPedometer.Info mInfo;
    private int isValidInfo = true;
    private final int WAITING_TIME = 3600000;
    private Timer mTimer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        mPedometer = new SmotionPedometer(Looper.getMainLooper(), motion);
        mPedometer.start(changeListener);

        mTimer = new Timer();
        mTimer.schedule(new CustomTimer(), WAITING_TIME);
    }

    private final SmotionPedometer.Changelistener changeListener =
        new SmotionPedometer.Changelistener() {

        @Override
        public void onChanged(INFO info) {
            // TODO Auto-generated method stub
            if (!isValidInfo) {
                mInfo = info;
                isValidInfo = true;
            }
        }
    };

    private class CustomTimer extends TimerTask {
        @Override
        public void run() {
            mInfo = mPedometer.getInfo();
            if (mInfo == null) { // This info is not valid.
                mPedometer.updateInfo();
            }
            isValidInfo = false;
        } else {
            isValidInfo = true;
        }
    }
}
```


4.2.3. Tracking UserActivities

SmotionActivity offers you the following methods to get activity data:

- getStatus() gets the user activity status.
- getAccuracy() gets the accuracy of the detected activity.
- getTimeStamp() gets the timestamp in milliseconds.

If you start walking or running or getting in a vehicle after calling SmotionActivity.start(), SmotionActivity captures the activity status and accuracy of the activity.

You can track activities in the following modes:

- Real time: When the status or accuracy changes, your application can receive activity information while the device's screen is on.
- Batch: The batch FIFO stores the timestamp, status and accuracy. When the FIFO is full, your application can receive activity information.

The following sample code shows how to use a listener for receiving data.

```
public class MainActivity extends Activity {
    private Smotion motion;
    private SmotionActivity mActivity;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        mActivity = new SmotionActivity(Looper.getMainLooper(), motion);
        mActivity.start(changeListener);
    }

    @Override
    protected void onDestroy() {
    }
    // Update the activity data by listener callback.
    private final SmotionActivity.ChangeListener changeListener =
        new SmotionPedometer.ChangeListener() {

        @Override
        public void onChanged(int mode, Info[] infoArray) {
            // TODO Auto-generated method stub
            SmotionPedometer.Info[] activityInfo = infoArray;
            if (mode == SmotionActivity.Info.MODE_REALTIME) {
                int status = activityInfo[0].getStatus();
                int accuracy = activityInfo[0].getAccuracy();
                int timestamp = activityInfo[0].getTimeStamp();
            } else if (mode == SmotionActivity.Info.MODE_BATCH) {
                for (int i = 0; i < infoArray.length; i++) {
                    int status = activityInfo[i].getStatus();
                    int accuracy = activityInfo[i].getAccuracy();
                    int timestamp = activityInfo[i].getTimeStamp();
                }
            }
        }
    };
}
```

4.2.4. Using ActivityNotifications

SmotionActivityNotification offers you the following methods to get specific activity data with notifications:

- `getStatus()` gets the user activity status.
- `getAccuracy()` gets the accuracy of the detected activity.
- `getTimeStamp()` gets the timestamp in milliseconds.

You can select a specific activity type for notifications using the `InfoFilter.addActivity()` method. If you start walking or running or getting in a vehicle after `SmotionActivityNotification.start()` is called, `SmotionActivityNotification` captures the specific activity status and accuracy of the activity.

The following sample code shows how to use a listener for receiving data.

```
public class MainActivity extends Activity {
    private Smotion motion;
    private SmotionActivityNotification mActivityNotification;
    private SmotionActivityNotification.InfoFilter mFilter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        mFilter = new InfoFilter();
        mFilter.addActivity(SmotionActivityNotification.Info.STATUS_VEHICLE);
        mActivityNotification =
            new SmotionActivityNotification(Looper.getMainLooper(), motion);
        mActivityNotification.start(changeListener, mFilter);
    }

    @Override
    protected void onDestroy() {
        ...
    }

    // Update the activity data by listener callback.
    private final SmotionActivityNotification.ChangeListener changeListener =
        new SmotionActivityNotification.ChangeListener() {

            @Override
            public void onChanged(Info info) {
                // TODO Auto-generated method stub
                int status = info.getStatus();
                int accuracy = info.getAccuracy();
                int timestamp = info.getTimeStamp();
            }
        };
}
```

Copyright

Copyright © 2013 Samsung Electronics Co. Ltd. All Rights Reserved.

Though every care has been taken to ensure the accuracy of this document, Samsung Electronics Co., Ltd. cannot accept responsibility for any errors or omissions or for any loss occurred to any person, whether legal or natural, from acting, or refraining from action, as a result of the information contained herein. Information in this document is subject to change at any time without obligation to notify any person of such changes.

Samsung Electronics Co. Ltd. may have patents or patent pending applications, trademarks copyrights or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give the recipient or reader any license to these patents, trademarks copyrights or other intellectual property rights.

No part of this document may be communicated, distributed, reproduced or transmitted in any form or by any means, electronic or mechanical or otherwise, for any purpose, without the prior written permission of Samsung Electronics Co. Ltd.

The document is subject to revision without further notice.

All brand names and product names mentioned in this document are trademarks or registered trademarks of their respective owners.

For more information, please visit <http://developer.samsung.com/>