About This Article

Many of the developers in Android require the unique Device ID of mobile handsets in special some case.

for example, Device unique ID can be used , at tracking apps installation, at generating drm for copy protection.

A sample code for reference is provided at the end of this article.

Scope:

This article gives a introduction about how to read variety Android Device's ID for using identifier.

It assumes that the user has already installed Android and necessary tools to develop application. It also assumes that the user to be familiar with Basic knowledge of Android.

Introduction

In Android, There are already variety type of device identifier in android devices..

Previously all android devices are having telephony services hence it become easy to retrieve the IMEI, MEID, or ESN of the phone, which is unique to that piece of hardware.

But Wifi-only devices or music players that don’t have telephony hardware just don’t have this kind of unique identifier. Hence this article explains read identifier of device in different android devices.

Variety ways to retrieve device id in Android.

Given below are the different types of identifying devices ID in android devices.

- Unique number (IMEI, MEID, ESN, IMSI)
- MAC Address
- Serial Number
- ANDROID_ID

Unique number (IMEI, MEID, ESN, IMSI)

Description

In the past, when every Android device was a phone, things were simpler :

TelephonyManager.getDeviceId() is required to return (depending on the network technology) the IMEI, MEID, ESN and IMSI of the phone, which is unique to that piece of hardware.

The IMEI, MEID, ESN, IMSI can be defined as follows:

- IMEI(International Mobile Equipment Identity)
  The unique number to identify GSM, WCDMA mobile phones as well as some satellite phones
- MEID(Mobile Equipment IDentifier)
  The globally unique number identifying a physical piece of CDMA mobile station equipment, the MEID was created to replace ESNs(Electronic Serial Number)
- ESN(Electronic Serial Number)

The unique number to identify CDMA mobile phones

- IMSI(International Mobile Subscriber Identity)
  The unique identification associated with all GSM and UMTS network mobile phone users

To retrieve the Device ID, you would include the following code in your project:

```
import android.telephony.TelephonyManager;
import android.content.Context;

String   imeistring = null;
String   imsistring = null;



{
    TelephonyManager    telephonyManager;

    telephonyManager  =
          ( TelephonyManager )getSystemService( Context.TELEPHONY_SERVICE );

    /*
     * getDeviceId() function Returns the unique device ID.
     * for example,the IMEI for GSM and the MEID or ESN for CDMA phones.
     */
    imeistring = telephonyManager.getDeviceId();

   /*
    * getSubscriberId() function Returns the unique subscriber ID,
  * for example, the IMSI for a GSM phone.
  */
   imsistring = telephonyManager.getSubscriberId();
}
```

To allow read only access to phone state add permission **READ_PHONE_STATE** in the **AndroidManifest.xml**

```
<uses-permission
  android:name="android.permission.READ_PHONE_STATE" >
</uses-permission>
```

Disadvantages

- Android devices should have telephony services
- It doesn’t work reliably
- Serial Number
- When it does work, that value survives device wipes (“Factory resets”) and thus you could end up making a nasty mistake when one of your customers wipes their device and passes it on to another person.

Mac Address

Description

It may be possible to retrieve a Mac address from a device’s Wi-Fi or Bluetooth hardware. But, it is not recommended using this as a unique identifier.

Disadvantages

- Device should have Wi-Fi (where not all devices have Wi-Fi)
- If Wi-Fi present in Device should be turned on otherwise does not report the MAC address

Serial Number

Since Android 2.3 (“Gingerbread”) this is available via android.os.Build.SERIAL. Devices without telephony are required to report a unique device ID here; some phones may do so also.

Serial number can be identified for the devices such as MIDs (Mobile Internet Devices) or PMPs (Portable Media Players) which are not having telephony services.

Device-Id as serial number is available by reading the System Property Value “ro.serialno”

To retrieve the serial number for using Device ID, please refer to example code below.

```
import java.lang.reflect.Method;

String serialnum = null;

try {
         Class<?> c = Class.forName("android.os.SystemProperties");
         Method get = c.getMethod("get", String.class, String.class );
         serialnum = (String)(   get.invoke(c, "ro.serialno", "unknown" )  );
}
catch (Exception ignored)
{
}
```

Disadvantages

Serial Number is not available with all android devices

ANDROID_ID

Description

More specifically, Settings.Secure.ANDROID_ID. A 64-bit number (as a hex string) that is randomly generated on the device's first boot and should remain constant for the lifetime of the device (The value may change if a factory reset is performed on the device.) ANDROID_ID seems a good choice for a unique device identifier.

To retrieve the ANDROID_ID for using Device ID, please refer to example code below

```
String androidId =
Settings.Secure.getString(getContentResolver(),Settings.Secure.ANDROID_ID);
```

Disadvantages

- Not 100% reliable of Android prior to 2.2 (“Froyo”) devices

- Also, there has been at least one widely-observed bug in a popular handset from a major manufacturer, where every instance has the same ANDROID_ID.

Conclusion

For the vast majority of applications, the requirement is to identify a particular installation, not a physical device. Fortunately, doing so is straightforward.

Given below are the some of the best approaches for using Device ID:

- Another approach for supporting various device types is the combination use of getDeviceID() API and ro.serialno
- There are many good reasons for avoiding the attempt to identify a particular device. For those who want to try, the best approach is probably the use of ANDROID_ID on anything reasonably modern, with some fallback heuristics for legacy devices

Sample Example

Given below is the sample example on tracking installation in Android

Class: ReadDeviceID.java

```
package com.deviceid;

import java.lang.reflect.Method;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.provider.Settings;
import android.telephony.TelephonyManager;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class ReadDeviceID extends Activity {

        Button bt;
        TextView idView;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        bt=(Button)findViewById(R.id.button1);
```

```
        idView=(TextView)findViewById(R.id.textView1);
        bt.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
          String imeistring=null;
              String imsistring=null;

        TelephonyManager   telephonyManager  =  ( TelephonyManager
                                                    )
getSystemService( Context.TELEPHONY_SERVICE );

            /*
         * getDeviceId() function Returns the unique device ID.
         * for example,the IMEI for GSM and the MEID or ESN for CDMA phones.
         */
         imeistring = telephonyManager.getDeviceId();
        idView.append("IMEI No : "+imeistring+"\n");



            /*
             * getSubscriberId() function Returns the unique subscriber ID,
             * for example, the IMSI for a GSM phone.
             */
         imsistring = telephonyManager.getSubscriberId();
         idView.append("IMSI No : "+imsistring+"\n");

            /*
         * System Property ro.serialno returns the serial number as unique number
         * Works for Android 2.3 and above
         */

         String hwID = android.os.SystemProperties.get("ro.serialno", "unknown");
         idView.append( "hwID : " + hwID + "\n" );
             String serialnum = null;
         try {
           Class<?> c = Class.forName("android.os.SystemProperties");
           Method get = c.getMethod("get", String.class, String.class );
               serialnum = (String)(   get.invoke(c, "ro.serialno", "unknown" )  );
            idView.append( "serial : " + serialnum + "\n" );
                 } catch (Exception ignored) {
           }
        String serialnum2 = null;
           try {
```

```
        Class myclass = Class.forName( "android.os.SystemProperties" );
        Method[] methods = myclass.getMethods();
        Object[] params = new Object[] { new String( "ro.serialno" ) , new String(
              "Unknown" ) };
                serialnum2 = (String)(methods[2].invoke( myclass, params ));
            idView.append( "serial2 : " + serialnum2 + "\n" );
           }catch (Exception ignored)
        {
        }
          /*
           * Settings.Secure.ANDROID_ID returns the unique DeviceID
           * Works for Android 2.2 and above
           */
        String androidId = Settings.Secure.getString(getContentResolver(),
                                                     Settings.Secure.ANDROID_ID);
            idView.append( "AndroidID : " + androidId + "\n" );
         }
    });

    }
}
```

Class: SystemProperties.java

```
package android.os;

/**
 * Gives access to the system properties store.  The system properties
 * store contains a list of string key-value pairs.
 *
 * {@hide}
 */
public class SystemProperties
{
    public static final int PROP_NAME_MAX = 31;
    public static final int PROP_VALUE_MAX = 91;

    private static native String native_get(String key);
    private static native String native_get(String key, String def);
    private static native int native_get_int(String key, int def);
    private static native long native_get_long(String key, long def);
    private static native boolean native_get_boolean(String key, boolean def);
    private static native void native_set(String key, String def);

    /**
```

```
     * Get the value for the given key.
     * @return an empty string if the key isn't found
     * @throws IllegalArgumentException if the key exceeds 32 characters
     */
    public static String get(String key) {
        if (key.length() > PROP_NAME_MAX) {
            throw new IllegalArgumentException("key.length > " + PROP_NAME_MAX);
        }
        return native_get(key);
    }

    /**
     * Get the value for the given key.
     * @return if the key isn't found, return def if it isn't null, or an empty string
otherwise
     * @throws IllegalArgumentException if the key exceeds 32 characters
     */
    public static String get(String key, String def) {
        if (key.length() > PROP_NAME_MAX) {
            throw new IllegalArgumentException("key.length > " + PROP_NAME_MAX);
        }
        return native_get(key, def);
    }

    /**
     * Get the value for the given key, and return as an integer.
     * @param key the key to lookup
     * @param def a default value to return
     * @return the key parsed as an integer, or def if the key isn't found or
     *         cannot be parsed
     * @throws IllegalArgumentException if the key exceeds 32 characters
     */
    public static int getInt(String key, int def) {
        if (key.length() > PROP_NAME_MAX) {
            throw new IllegalArgumentException("key.length > " + PROP_NAME_MAX);
        }
        return native_get_int(key, def);
    }

    /**
     * Get the value for the given key, and return as a long.
     * @param key the key to lookup
     * @param def a default value to return
```

```
     * @return the key parsed as a long, or def if the key isn't found or
     *         cannot be parsed
     * @throws IllegalArgumentException if the key exceeds 32 characters
     */
    public static long getLong(String key, long def) {
        if (key.length() > PROP_NAME_MAX) {
            throw new IllegalArgumentException("key.length > " + PROP_NAME_MAX);
        }
        return native_get_long(key, def);
    }

    /**
     * Get the value for the given key, returned as a boolean.
     * Values 'n', 'no', '0', 'false' or 'off' are considered false.
     * Values 'y', 'yes', '1', 'true' or 'on' are considered true.
     * (case insensitive).
     * If the key does not exist, or has any other value, then the default
     * result is returned.
     * @param key the key to lookup
     * @param def a default value to return
     * @return the key parsed as a boolean, or def if the key isn't found or is
     *         not able to be parsed as a boolean.
     * @throws IllegalArgumentException if the key exceeds 32 characters
     */
    public static boolean getBoolean(String key, boolean def) {
        if (key.length() > PROP_NAME_MAX) {
            throw new IllegalArgumentException("key.length > " + PROP_NAME_MAX);
        }
        return native_get_boolean(key, def);
    }

    /**
     * Set the value for the given key.
     * @throws IllegalArgumentException if the key exceeds 32 characters
     * @throws IllegalArgumentException if the value exceeds 92 characters
     */
    public static void set(String key, String val) {
        if (key.length() > PROP_NAME_MAX) {
            throw new IllegalArgumentException("key.length > " + PROP_NAME_MAX);
        }
        if (val != null && val.length() > PROP_VALUE_MAX) {
            throw new IllegalArgumentException("val.length > " +
                PROP_VALUE_MAX);
```

```
        }
        native_set(key, val);
    }
}
```

Create the project **"com.deviceid"** with the activity **"ReadDeviceID"**. Change the layout "main.xml" to the following.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button
    android:text="GetDeviceID"
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</Button>
<TextView
    android:id="@+id/textView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
</TextView>
</LinearLayout>
```

Add the permission **"READ_PHONE_STATE"** to **"AndroidManifest.xml"** to allow your application to access the internet.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.deviceid"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="7" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ReadDeviceID"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

    <uses-permission
            android:name="android.permission.READ_PHONE_STATE" >
      </uses-permission>

</manifest>
```

Output

Figure below shows the output of above sample example

ref:http://developer.samsung.com/android/technical-docs/How-to-retrieve-the-Device-Unique-ID-from-android-device