Introduction

This walkthrough shows one way to provide voice for Short Message Service (SMS) messages received on an Android device. The Android platform provides a Text-to-Speech (TTS) API which can be used to “speak” text of different languages.

Text-to-Speech (TTS) API

First, check for the support of the TTS engine at the launch of application Activity. Use the action ACTION_CHECK_TTS_DATA in the intent for activity result as shown below:

```
Intent findIntent = new Intent();
findIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(findIntent, MY_DATA_CHECK_CODE);
@Override
public void onCreate(Bundle savedInstanceState) {
                  super.onCreate(savedInstanceState);
                  setContentView(R.layout.main);
                  Intent checkIntent = new Intent();
                  checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
                  startActivityForResult(checkIntent, MY_DATA_CHECK_CODE);
}
```

Code 1: Checking Support for TTS engine

The result comes back through the onActivityResult() method.

The TextToSpeech.Engine.CHECK_VOICE_DATA_PASS result code indicates the availability of the TTS engine. If the TTS engine is not supported, then the app should notify user to install the relevant feature using the TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA action. The intent with TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA takes the user to Google Play and allows the user to download the TTS engine.

```
private TextToSpeech textSpeech;
protected void onActivityResult(
                    int requestCode, int resultCode, Intent data) {

          if (requestCode == MY_DATA_CHECK_CODE) {
         if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
                  //success initialization
                  textSpeech = new TextToSpeech(this, this);
         } else {
            //failed please initialize
            Intent installIntent = new Intent();
         installIntent.setAction(
                    TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
         startActivity(installIntent);
          }
```

```
        }
}
```

Code 2: TTS engine initialization/installation code

The following code snippet shows how to load and set the language:

```
textSpeech.setLanguage(Locale.US);
```

Code 3: Set TTS Language for speaking

When the engine is fully loaded, the TTS engine notifies the app through the onInit(int status) method. For the application to get the notification, it should implement the OnInitListener interface.

```
public void onInit(int status) {
  if (status == TextToSpeech.SUCCESS) {
        Toast.makeText(
                  TTSProjectActivity.this,       "Sucessfull intialization of Text-To-
                                             Speech engine", Toast.LENGTH_LONG).show();
   } else if (status == TextToSpeech.ERROR) {
        Toast.makeText(
                  TTSProjectActivity.this,"Unable to initialize Text-To-Speech
                  engine",Toast.LENGTH_LONG).show();
   }
}
```

Code 4: TTS engine initialization

To make the application speak the text, use the following code:

```
speak(String data, int mode, HashMap params) method as below:
private String text=”hi how are you”;
textSpeech.speak(text, TextToSpeech.QUEUE_ADD, null);
```

Code 5: TTS engine speak

The second parameter is the queue mode which can be QUEUE_ADD or QUEUE_FLUSH. As the name suggests, QUEUE_ADD adds text at the end of the playback queue and QUEUE_FLUSH flushes all entries present in the queue and adds the new text data in queue.

Using Text-to-Speech (TTS) API for Making SMS Speak

Create a new class extending BroadcastReceiver. This enables your application to listen to Incoming SMS messages. Implement onReceive() for listening to incoming SMS by checking for SMS action as follows.

```
public class SMSReceiver extends BroadcastReceiver {
          public static final String ACTION =
                                      "android.provider.Telephony.SMS_RECEIVED";
        @Override
        public void onReceive(Context context, Intent intent) {
```

```
                    if(intent.getAction().equals(ACTION)) {
                    }
            }
}
```

To get notified whenever there is an incoming SMS, dynamically register an instance of this class with context. registerReceiver() or statically publish an implementation through the <receiver>tag in AndroidManifest.xml file. Registering for incoming SMS dynamically can be done as below:

```
SMSReceiver receiver = new SMSReceiver();
public void registerSMS() {
   IntentFilter filter = new IntentFilter();
   filter.addAction(ACTION);
   registerReceiver(receiver, filter);
}
```

Code 6: SMS Receiver

To unregister this instance dynamically use the unregisterReceiver(receiver) method.

Now the application is ready to receive the incoming SMS messages. For reading incoming SMS, get the bundle object from the intent in the onReceive() method and read all SMS details as shown in the following code snippet:

```
@Override
public void onReceive(Context context, Intent intent) {

 this.mcontext = context;
 if (intent.getAction().equals(ACTION)) {
   Bundle bundle = intent.getExtras();
   if (bundle != null) {
        readSMS(bundle);
   }
 }
}
```

Code 7: Getting sms from intent

The following code snippet shows the function for reading the SMS details. The messages are stored in an Object array in the PDU format. The SmsMessage.createFromPdu() method is used to extract the SMS messages.

```
public void readSMS(Bundle bundle) {
  SmsMessage[] msgs = null;
  try {
   Object[] pdus = (Object[]) bundle.get("pdus");
  if (pdus != null) {
        msgs = new SmsMessage[pdus.length];
        String smsBodyStr = null;
        String phoneNoStr = null;
   for (int k = 0; k < msgs.length; k++) {
         msgs[k] = SmsMessage.createFromPdu((byte[]) pdus[k]);
         smsBodyStr = msgs[k].getMessageBody().trim();
         phoneNoStr = msgs[k].getOriginatingAddress().trim();

         //Using these details for reading SMS using TTS
   }
 }
} catch (Exception exe) {
  exe.printStackTrace();
}
}
```

Code 8: Getting SMS from PDU Bundle

To display the name of the contact sending the SMS message, the following code snippet gets the contact name using the mobile number:

```
public String getContactName(String mobileno) {
  Uri uri = Uri.withAppendedPath(PhoneLookup.CONTENT_FILTER_URI,
                          Uri.encode(mobileno));
  Cursor cr = mcontext.getContentResolver().query(uri,
                                  new String[] { PhoneLookup.DISPLAY_NAME }, null,
                                                           null, null);
  String displayName = null;

  if (cr.getCount() > 0) {
        cr.moveToFirst();
        displayName = cr.getString(cr
                          .getColumnIndex(PhoneLookup.DISPLAY_NAME));
  }
  return displayName;
}
```

Here is a function that allows creating voice for the SMS received by your application:

```
public void speakSMS(String smsBodyStr, String phoneNoStr) {
  if (phoneNoStr != null) {
        String displayName = getContactName(phoneNoStr);
        if (displayName == null) {
           displayName = phoneNoStr;
        }

        if (smsBodyStr != null && smsBodyStr.length() > 0) {
           //make you SMS speak
        VoiceofText.textSpeech.speak("SMS Received From " + displayName
                + "message is" + smsBodyStr, TextToSpeech.QUEUE_ADD,null);
        }
  }
}
```

Code 9: Making TTS speak

Example

The following example demonstrates how to make your android application speak to all Incoming SMS messages.

- main.xml

The activity allows the user to register to listen to an incoming SMS messages and reading them out using TTS. It also allows user to deregister from listening to incoming SMS messages. The corresponding layout file is shown below

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
        <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/heading" />
        <Button
                android:id="@+id/regButton"
                android:layout_marginTop="10dp"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Speak SMS" />
        <Button
                android:id="@+id/unregButton"
                android:layout_marginTop="10dp"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Stop Speaking SMS" />
</LinearLayout>
```

Code 10: Main.xml

- AndroidManifest.xml

To receive SMS services and read the name of the contact, the user needs to add the following permissions. The application consists of one Activity and BroadcastReceiver.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
         package="com.voice.text"
         android:versionCode="1"
         android:versionName="1.0" >
         <uses-sdk android:minSdkVersion="8" />
         <uses-permission android:name="android.permission.RECEIVE_SMS"/>
         <uses-permission android:name="android.permission.READ_CONTACTS"/>
         <application
                  android:icon="@drawable/ic_launcher"
                  android:label="@string/app_name" >
                  <activity
                           android:label="@string/app_name"
                           android:name=".VoiceofText" >
                           <intent-filter >
                                    <action android:name="android.intent.action.MAIN" />
                                    <category
android:name="android.intent.category.LAUNCHER" />
                           </intent-filter>
                  </activity>
         </application>
</manifest>
```

Code 11: Manifest xml

- VoiceofText.java

The activity shows two buttons: “Speak SMS”, for registering the broadcast receiver for SMS; and “Stop Speaking SMS”, for unregistering the SMS broadcast receiver. When the application launches, the TTS engine initializes if it is available. If it is not available, the user is prompted to install it from Google Play. Once the TTS engine is initialized, every incoming SMS message will be spoken by the application.

```
package com.voice.text;

import android.app.Activity;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
```

```
public class VoiceofText extends Activity implements OnClickListener,
                  OnInitListener {

  /*Button for resgistering and deregistering SMS broadcast receiver*/
  private Button regButton;
  private Button unRegButton;

  private int MY_DATA_CHECK_CODE = 0;
  public static TextToSpeech textSpeech;

  /*Action defined for receiving SMS */
  public static final String ACTION =
                                    "android.provider.Telephony.SMS_RECEIVED";

  private SMSReceiver receiver;

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

   regButton = (Button) findViewById(R.id.regButton);
   unRegButton = (Button) findViewById(R.id.unregButton);

   /*register button click listeners*/
   regButton.setOnClickListener(this);
   unRegButton.setOnClickListener(this);

   /*Start Activity for result*/
   Intent findIntent = new Intent();
   findIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
   startActivityForResult(findIntent, MY_DATA_CHECK_CODE);

  }

/*
At the launch of application Activity will check for the support of TTS by starting an
activity for Result.
The result comes back through onActivityResult() method.
*/
```

```
protected void onActivityResult(int requestCode,
int resultCode, Intent data) {
/*
MY_DATA_CHECK_CODE indicates availability of TTS Engine else if not Available then notify
user to install the
relevant feature using TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA action
*/
         if (requestCode == MY_DATA_CHECK_CODE) {
          if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
                          /*Initialize the TTS engine*/
          textSpeech = new TextToSpeech(this, this);
          } else {
                   Intent installIntent = new Intent();
          installIntent.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
                                   startActivity(installIntent);
         }
         }
 }

/*
Register and de register SMS broadcast receiver
*/
 @Override
 public void onClick(View v) {
   switch (v.getId()) {
         case R.id.regButton:
                    registerSMS();
         break;
         case R.id.unregButton:
           unregisterSMS();
         break;
   }
}

/*
If the engine is fully loaded, TTS engine notifies application through
onInit(int status) method.
*/
@Override
public void onInit(int status) {
  /*Success. Engine loaded.*/
  if (status == TextToSpeech.SUCCESS) {
         Toast.makeText(VoiceofText.this,
                  "Sucessfull intialization of Text-To-Speech engine",
                                             Toast.LENGTH_LONG).show();
  } else
  /*Failure. Unable to load Engine.*/
  if (status == TextToSpeech.ERROR) {
        Toast.makeText(VoiceofText.this,
                           "Unable to initialize Text-To-Speech engine",
                                             Toast.LENGTH_LONG).show();
  }
}

/*
On application exit, uninitialized TTS engine and unregister SMS broadcastreceiver.
*/
@Override
public void onDestroy() {
   if (textSpeech != null) {
        textSpeech.stop();
        textSpeech.shutdown();
   }
 unregisterSMS();
 super.onDestroy();
}



/*
Register SMS receiver broadcast for listening to incoming SMS.
*/
public void registerSMS() {
 receiver = new SMSReceiver();
 IntentFilter filter = new IntentFilter();
 filter.addAction(ACTION);
 registerReceiver(receiver, filter);
 Toast.makeText(getApplicationContext(), "registered for incomming sms",
                                    Toast.LENGTH_LONG).show();
}

/*
Unregister SMS receiver broadcast and stop listening to incoming SMS.
*/

public void unregisterSMS() {
 try {
  unregisterReceiver(receiver);
  Toast.makeText(getApplicationContext(),
                                            "unregistered for listening sms",
                                             Toast.LENGTH_LONG).show();
  } catch (Exception exe) {
        Toast.makeText(getApplicationContext(),
                                            "sms listener is already unregistered",
                                                             Toast.LENGTH_LONG).show();
  }
 }
}
```

Code 12: VoiceOfText.java

- SMSReceiver.java

The BroadcastReceiver class listens for incoming SMS messages. When one occurs, the onReceive() method notifies the application.

```
package com.voice.text;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract.PhoneLookup;
import android.speech.tts.TextToSpeech;
import android.telephony.SmsMessage;

public class SMSReceiver extends BroadcastReceiver {
  private Context mcontext;

 /*Check for the intent action and if it matches the one defined in the application, process
it*/

  @Override
  public void onReceive(Context context, Intent intent) {
        this.mcontext = context;
        if (intent.getAction().equals(VoiceofText.ACTION)) {
                Bundle bundle = intent.getExtras();
        if (bundle != null) {
         readSMS(bundle);
        }
```

```
          }
  }

/*
Method will get sms data from bundle
*/
  public void readSMS(Bundle bundle) {
         SmsMessage[] msgs = null;

         try {

/*It will get pdus object array*/
          Object[] pdus = (Object[]) bundle.get("pdus");

                  /*if the pdus data is not null*/
          if (pdus != null) {

  /*Create an SmsMessage array of pdus length*/
            msgs = new SmsMessage[pdus.length];

            String smsBodyStr = null;
            String phoneNoStr = null;

                    /* Loop through the length and create SmsMessage object*/
            for (int k = 0; k < msgs.length; k++) {
                  msgs[k] = SmsMessage.createFromPdu((byte[]) pdus[k]);
                              /* Get sms data*/
                  smsBodyStr = msgs[k].getMessageBody().trim();
                              /* Get contact phonenumber*/

                  phoneNoStr = msgs[k].getOriginatingAddress().trim();

                  /*provide the SMS data to TTS engine for speaking*/
                  speakSMS(smsBodyStr, phoneNoStr);
                    }
         }
         } catch (Exception exe) {
                  exe.printStackTrace();
         }
  }

/*
speakSMS method will speak the data provided in the speak() method.
```

```
It will get the contact name from the phonebook*/
  public void speakSMS(String smsBodyStr, String phoneNoStr) {

        /*if the phone number is not null*/
        if (phoneNoStr != null) {

/*Get name of the contact from phonebook*/
        String displayName = getContactName(phoneNoStr);

           /*if unable to find the name of the contact then consider phone number as
                 Name of the contact*/
        if (displayName == null) {
          displayName = phoneNoStr;
         }

        if (smsBodyStr != null && smsBodyStr.length() > 0) {
         VoiceofText.textSpeech.speak("SMS Received From " + displayName
                + "message is" + smsBodyStr, TextToSpeech.QUEUE_ADD, null);
        }
   }
 }

/*
Method will search ih the phonebook database and return the name of the contact
*/
  public String getContactName(String mobileno) {
           /*Create URI with phone number */
        Uri uri = Uri.withAppendedPath(PhoneLookup.CONTENT_FILTER_URI,
                                                              Uri.encode(mobileno));
/* Query database and get back the cursor*/
        Cursor cr = mcontext.getContentResolver().query(uri,
                new String[] { PhoneLookup.DISPLAY_NAME }, null, null, null);
                String displayName = null;

           /*If cursor count is greater than zero then contact name is found*/
        if (cr.getCount() > 0) {
         cr.moveToFirst();
         displayName = cr.getString(cr.getColumnIndex(PhoneLookup.DISPLAY_NAME));
        }
           return displayName;
  }
}
```

Code 13: SMSReceiver.java

ref:http://developer.samsung.com/android/technical-docs/Providing-Voice-for-Incoming-SMS