```java
package io.renaud.ligo.text_demo;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.hardware.usb.UsbAccessory;
import android.hardware.usb.UsbManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.IBinder;
import android.os.ParcelFileDescriptor;
import android.util.Log;

/**
*
*/

import java.io.BufferedInputStream;  // test
import java.io.BufferedOutputStream;
import java.io.FileDescriptor;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.*;

/**
 * \class LigoService
 * Some multiline comments

      test
 * There should be a detailled LigoService description.
 */

public class LigoService extends Service {

    private UsbManager mUsbManager;
    private UsbAccessory mAccessory = null;
    private ParcelFileDescriptor mParcelFileDescriptor = null;
    private FileInputStream mInputStream = null;
    private BufferedOutputStream mOutputStream = null;
    private ExecutorService mPool;

    private int mTest_true = true;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        Log.d(Constants.LOGTAG, "LigoService.onCreate()");

        mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);
        // Use a custom ThreadPoolExecutor to reduce the KeepAliveTime of the threads (default to 60
        // seconds using Executors.newCachedThreadPool)???
        mPool = Executors.newCachedThreadPool();
    }

    @Override
    public void onDestroy() {
        Log.d(Constants.LOGTAG, "LigoService.onDestroy()");
        super.onDestroy();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(Constants.LOGTAG, "LigoService.onStartCommand()");
        String action = intent.getAction();

        if (action != null) {
            if (action.equals(Constants.ACTION_ATTACHED)) {

                Log.i(Constants.LOGTAG, "LigoService: processing ACTION_ATTACHED");
                UsbAccessory accessory =
                        (UsbAccessory)intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
                if (accessory != null) {
                    mAccessory = accessory;
                    new AccessoryInitializer().execute();
                }
            } else if (action.equals(Constants.ACTION_DETACHED)) {

                Log.i(Constants.LOGTAG, "LigoService: processing ACTION_DETACHED");
                closeAccessory();
                stopSelf();
            } else if (action.equals(Constants.ACTION_WRITE_DATA)) {

                Log.i(Constants.LOGTAG, "LigoService: processing ACTION_WRITE_DATA");
                if (mOutputStream != null) {

                    Bundle bundle = intent.getExtras();
                    mPool.execute(new UsbWriter(mOutputStream, bundle.getByteArray("output_buf")));

                } else {
                     Log.e(Constants.LOGTAG, "LigoService: the output stream is null!");
                }
            }
        }

        return START_NOT_STICKY;
    }

    // accessory as arg?
    private void openAccessory() {
        Log.d(Constants.LOGTAG, "LigoService.openAccessory()");
        mParcelFileDescriptor = mUsbManager.openAccessory(mAccessory);

        if (mParcelFileDescriptor != null) {
            Log.d(Constants.LOGTAG, "LigoService: Accessory successfully opened");

            FileDescriptor fd = mParcelFileDescriptor.getFileDescriptor();

            mInputStream = new FileInputStream(fd);
            mOutputStream =
                    new BufferedOutputStream(new FileOutputStream(fd));

            mPool.execute(new UsbReader(mInputStream));

        } else {
            Log.e(Constants.LOGTAG, "LigoService: Failed to open the accessory");
        }
    }

    private void closeAccessory() {
        Log.d(Constants.LOGTAG, "LigoService.closeAccessory()");

        mPool.shutdown();

        if (mAccessory != null) {

            try {
                mParcelFileDescriptor.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            mParcelFileDescriptor = null;
            mOutputStream         = null;
            mInputStream          = null;
        }
    }

    private class AccessoryInitializer extends AsyncTask<Void, Void, Void> {
        protected Void doInBackground(Void... arg0) {
            Log.d(Constants.LOGTAG, "AccessoryInitializer.doInBackground()");
            openAccessory();
            return null;
        }
    }

    private class UsbWriter implements Runnable {
        private final BufferedOutputStream mOutput;
        private final byte[] mData;

        public UsbWriter (BufferedOutputStream output, byte[] data) {
            mOutput = output;
            mData = data;
        }

        public void run () {
            try {
                mOutput.write(mData, 0, mData.length);
                mOutput.flush();
            } catch (IOException e) {
                Log.e(Constants.LOGTAG, "UsbWriter IOException");
                Log.e(Constants.LOGTAG, e.getMessage());
            }
        }
    }

    private class UsbReader implements Runnable {
        private FileInputStream mInput;
        private static final int BUFFER_SIZE = 8192;

        public UsbReader(FileInputStream input) {
            mInput = input;
        }

        public void run() {

            byte[] inputBuf = new byte[BUFFER_SIZE];
            try {
                while (mInput.read(inputBuf) != -1) {

                    Log.d(Constants.LOGTAG, "Read some bytes");

                    Intent readIntent = new Intent(Constants.ACTION_READ_DATA);
                    Bundle bundle = new Bundle();
                    bundle.putByteArray("input_buf", inputBuf);
                    readIntent.putExtras(bundle);
                    sendBroadcast(readIntent);

                    // blank the byte array
                    inputBuf = new byte[BUFFER_SIZE];
                }
            } catch (IOException e) {
                Log.e(Constants.LOGTAG, "UsbReader IOException");
                Log.e(Constants.LOGTAG, e.getMessage());
             }

        }
    }

}
```