

Android-Adding SystemService

From Texas Instruments Wiki

Content is no longer maintained and is being kept for reference only!

This wiki page will demonstrate - "How to add system service to android framework". Example - "Adding a Bluetooth HID service" - taken as reference of understanding.This will also help to add support for more bluetooth profiles into android framework.

Contents

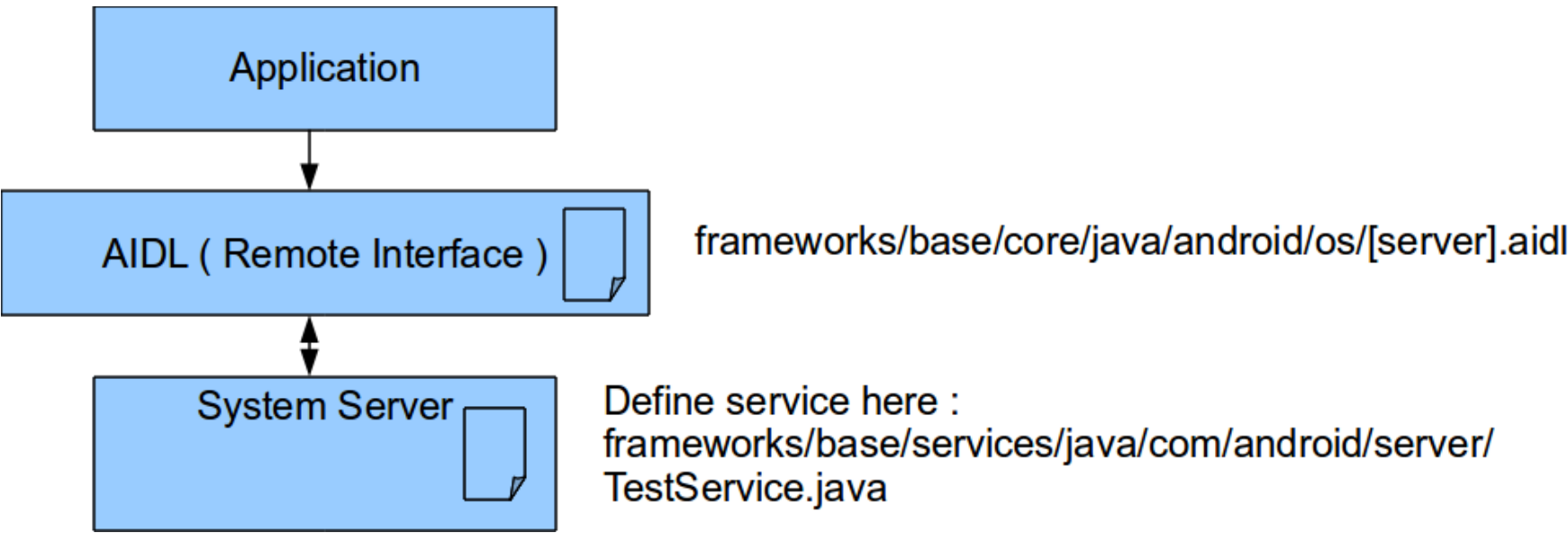
- 1 What is service?
- 2 Service layer
- 3 Create service
- 4 Register service
- 5 Expose service
- 6 Add [service].aidl for build
- 7 Using service
- 8 References
- 9 Support

What is service?

As per the definition given at <http://developer.android.com/guide/topics/fundamentals/services.html>

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

Service layer



Create service

- Add your code to frameworks/base/services/java/com/android/server/

```
/*TestService.java */
package com.android.server;
import android.content.Context;
import android.os.Handler;
import android.os.ITestService;
import android.os.Looper;
import android.os.Message;
import android.os.Process;
import android.util.Log;
public class TestService extends ITestService.Stub {
    private static final String TAG = "TestService";
    private TestWorkerThread mWorker;
    private TestWorkerHandler mHandler;
    private Context mContext;
    public TestService(Context context) {
        super();
        mContext = context;
        mWorker = new TestWorkerThread("TestServiceWorker");
        mWorker.start();
        Log.i(TAG, "Spawned worker thread");
    }

    public void setValue(int val) {
```

```
Log.i(TAG, "setValue " + val);
Message msg = Message.obtain();
msg.what = TestWorkerHandler.MESSAGE_SET;
msg.arg1 = val;
mHandler.sendMessage(msg);
}

private class TestWorkerThread extends Thread {
    public TestWorkerThread(String name) {
        super(name);
    }
    public void run() {
        Looper.prepare();
        mHandler = new TestWorkerHandler();
        Looper.loop();
    }
}

private class TestWorkerHandler extends Handler {
    private static final int MESSAGE_SET = 0;
    @Override
    public void handleMessage(Message msg) {
        try {
            if (msg.what == MESSAGE_SET) {
                Log.i(TAG, "set message received: " + msg.arg1);
            }
        } catch (Exception e) {
            // Log, don't crash!
            Log.e(TAG, "Exception in TestWorkerHandler.handleMessage:", e);
        }
    }
}
```

Register service

- Register service in SystemServer.java

```
/*
 * go to function "@Override public void run()"
 * .....
 * Add following block after line "if (factoryTest != SystemServer.FACTORY_TEST_LOW_LEVEL) {"
 */

try {
    Slog.i(TAG, "Test Service");
    ServiceManager.addService("Test", new TestService(context));
} catch (Throwable e) {
    Slog.e(TAG, "Failure starting TestService Service", e);
}
```

Expose service

- A service can expose set of functions that can be access by other process/application. Exposed functions are required to be declared in .aidl file at following location

frameworks/base/core/java/android/os/[server].aidl

```
/*
 * aidl file : frameworks/base/core/java/android/os/ITestService.aidl
 * This file contains definitions of functions which are exposed by service
 */
package android.os;
interface ITestService {
    /**
     * {@hide}
     */
    void setValue(int val);
}
```

Add [service].aidl for build

```
/*
 * open frameworks/base/Android.mk and add following line
 */
...
core/java/android/os/IPowerManager.aidl \
core/java/android/os/ITestService.aidl \
core/java/android/os/IRemoteCallback.aidl \
...
```

- Rebuild the framework/base or android system.Service is now ready to use by other application/process.

Using service

To use service

- first get service handle using "ServiceManager.getService()" api
- use service handle to call set of functions exposed by service

Below is the sample code to use service.

```
/*
 * HelloServer.java
 */
package com.Test.helloserver;
import android.app.Activity;
import android.os.Bundle;
import android.os.ServiceManager;
import android.os.ITestService;
import android.util.Log;
public class HelloServer extends Activity {
    private static final String DTAG = "HelloServer";
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);


        ITestService om = ITestService.Stub.asInterface(ServiceManager.getService("Test"));
        try {
            Log.d(DTAG, "Going to call service");
            om.setValue(20);
            Log.d(DTAG, "Service called succesfully");
        }
        catch (Exception e) {
            Log.d(DTAG, "FAILED to call service");
            e.printStackTrace();
        }
    }
}
```

References

- <http://developer.android.com/reference/android/app/Service.html>
- <http://developer.android.com/guide/topics/fundamentals/services.html>
- <http://www.opersys.com/blog/esc-india-2011-wrapup>

Support

For community support join <http://groups.google.com/group/rowboat>
For IRC #rowboat on irc.freenode.net



Engage in the
TI E2E™ Community
Ask questions, share knowledge, explore ideas
and help solve problems with fellow engineers

For technical support please post your questions at <http://e2e.ti.com>. Please post only comments about the article Android-Adding SystemService here.

Links

Amplifiers & Linear (http://www.ti.com/lsds/ti/analog/amplifier_and_linear.page)	DLP & MEMS (http://www.ti.com/lsds/ti/analog/mems/mems.page)	Processors (http://www.ti.com/lsds/ti/dsp/embedded_processor.page)	Switches & Multiplexers (http://www.ti.com/lsds/ti/analog/switches_and_multiplexers.page)
Audio (http://www.ti.com/lsds/ti/analog/audio/audio_overview.page)	High-Reliability (http://www.ti.com/lsds/ti/analog/high_reliability.page)	■ ARM Processors (http://www.ti.com/lsds/ti/dsp/arm.page)	Temperature Sensors & Control ICs (http://www.ti.com/lsds/ti/analog/temperature_sensor.page)
Broadband RF/IF & Digital Radio (http://www.ti.com/lsds/ti/analog/rfif.page)	Interface (http://www.ti.com/lsds/ti/analog/interface/interface.page)	■ Digital Signal Processors (DSP) (http://www.ti.com/lsds/ti/dsp/home.page)	Wireless Connectivity (http://focus.ti.com/docs/wirelessoverview.tsp?familyId=2003&sectionId=646&tabId=2735)
Clocks & Timers (http://www.ti.com/lsds/ti/analog/clocksandtimers/clocks_and_timers.page)	Logic (http://www.ti.com/lsds/ti/logic/home_overview.page)	■ Microcontrollers (MCU) (http://www.ti.com/lsds/ti/microcontroller/home.page)	
Data Converters (http://www.ti.com/lsds/ti/analog/dataconverters/data_converter.page)	Power Management (http://www.ti.com/lsds/ti/analog/powermanagement/power_portal.page)	■ OMAP Applications Processors (http://www.ti.com/lsds/ti/omap-applications-processors/the-omap-experience.page)	

Retrieved from "http://processors.wiki.ti.com/index.php?title=Android-Adding_SystemService&oldid=217993"

Categories: AM389x | Android | Sitara Android | TI816x | TIDM37x | TIOMAP3

- This page was last modified on 28 June 2016, at 10:41.

- This page has been accessed 25,830 times.
- Content is available under Creative Commons Attribution-ShareAlike unless otherwise noted.