Add Hidl Service on Android 8.0 or later

table of Contents

This article uses LED as an example to add HIDL on aosp to familiarize yourself with the whole process.

1. Write and compile the hal file

Create led folder and base version 1.0 in the hardware/interfaces/directory. This version number is divided into two parts, major.minor. If the major version does not change, only api can be added, not modified.

Create ILed.hal and types.hal

hardware/interfaces/led/1.0/ILed.hal

```
package android.hardware.led@1.0;

interface ILed
{
 //get led status
  get() generates (LedStatus result);



 //set led status
  set(LedStatus  val) generates(int32_t  ret);

   getBrightnessRange() generates(bool ret,BrightnessRange range);

     setBrightnessValue(vec<int32_t>  range)  generates(bool ret);
```

```
  on() ;

  off() ;

};
```

hardware/interfaces/led/1.0/types.hal

```
package android.hardware.led@1.0;



enum LedStatus : uint32_t {
      LED_ON,
      LED_OFF,
      LED_BAD_VALUE,
};



struct BrightnessRange {
   uint32_t min;

   uint32_t max;
};
```

After adding, execute hardware/interfaces/update-makefiles.sh to automatically generate a compilation script, and then

Executing mm under the path hardware/interfaces/led/1.0/will generate the required hidl library. Next, we need to implement the hidl interface for the client to call.

2. Implement Hidl Interface

First realize the header file and corresponding cpp of the subclass ledImpl of ILed interface.

hardware/interfaces/led/1.0/default/ledImpl.h

```
#ifndef ANDROID_HARDWARE_LED_V1_0_LED_H
#define ANDROID_HARDWARE_LED_V1_0_LED_H
#include <android/hardware/led/1.0/ILed.h>

#include <hidl/Status.h>

#include <hidl/MQDescriptor.h>
```

```
namespace android {
namespace hardware {
namespace led {
namespace V1_0 {
namespace implementation {

using ::android::hardware::led::V1_0::LedStatus;
using ::android::hardware::led::V1_0::BrightnessRange;
using ::android::hardware::led::V1_0::ILed;
using ::android::hardware::hidl_array;
using ::android::hardware::hidl_string;
using ::android::hardware::hidl_vec;
using ::android::hardware::Return;
using ::android::hardware::Void;
using ::android::sp;


struct ledImpl : public ILed {
    public:
        ledImpl();
        Return<LedStatus>  get() override ;
        Return<int32_t> set(LedStatus val) override;
        Return<void> on() override;
        Return<void> off() override;
        Return<void>
getBrightnessRange(getBrightnessRange_cb          _hidl_cb)
override;
        Return<bool>       setBrightnessValue(const
hidl_vec<int32_t>& range) override;
    private:
        LedStatus state;
};

extern "C" ILed* HIDL_FETCH_ILed(const char* name);
} //namespace implementation
} //namespace V1_0
} //namespace led
} //namespace hardware
} //namespace android

#endif//ANDROID_HARDWARE_LED_V1_0_LED_H
```

hardware/interfaces/led/1.0/default/ledImpl.cpp

```
#define LOG_TAG "LedService"

#include <log/log.h>
#include "ledImpl.h"

namespace android {
namespace hardware {
namespace led {
namespace V1_0 {
namespace implementation {

ledImpl::ledImpl() {
    state = LedStatus::LED_BAD_VALUE;
    ALOGE("ledImpl Init status:%d", state);
}

Return<void> ledImpl::on() {
    state = LedStatus::LED_ON;
    ALOGE("ledImpl on status:%d", state);
  return Void();
}

Return<void> ledImpl::off() {
    state = LedStatus::LED_OFF;
    ALOGE("ledImpl off status:%d", state);
  return Void();
}

Return<LedStatus>  ledImpl::get() {
    return state;
}
Return<int32_t> ledImpl::set(LedStatus val) {
    if(val == LedStatus::LED_OFF || val == LedStatus::LED_ON)
        state = val;
    else
        return -1;
    return 0;
}

Return<void>
ledImpl::getBrightnessRange(getBrightnessRange_cb
```

```
_hidl_cb)
{
    ALOGE("ledImpl getBrightnessRange ");
    BrightnessRange range;
    range.max = 100;
    range.min = 1;
    _hidl_cb(true,range);
    return Void();

}

Return<bool>                ledImpl::setBrightnessValue(const
hidl_vec<int32_t>& range)
{
    ALOGE("ledImpl getBrightnessValue ");
    auto iter = range.begin();
    ALOGE("ledImpl   getBrightnessValue   range.begin:
%d",*iter);
    iter = range.end();
    ALOGE("ledImpl   getBrightnessValue   range.end:
%d",*iter);
    ALOGE("ledImpl   getBrightnessValue   range.size:
%zu",range.size());
    return true;
}

ILed* HIDL_FETCH_ILed(const char */*name*/) {
    ALOGE("ledImpl HIDL_FETCH_ILed ");
  return new ledImpl();
}

} //namespace implementation
} //namespace V1_0
} //namespace led
} //namespace hardware
} //namespace android
```

hardware/interfaces/led/1.0/default/Android.bp

```
cc_library_shared {
  name: "android.hardware.led@1.0-impl",
  defaults: ["hidl_defaults"],
  srcs: ["ledImpl.cpp"],
  shared_libs: [
    "libhidlbase",
```

```
        "libhidltransport",
        "libhardware",
        "liblog",
        "libutils",
        "android.hardware.led@1.0",
    ],
}

cc_binary {
    name: "android.hardware.led@1.0-service",
    init_rc: ["android.hardware.led@1.0-service.rc"],
    srcs: ["service.cpp",
      "ledImpl.cpp"],

    shared_libs: [
        "liblog",
        "libhardware",
        "libhidlbase",
        "libhidltransport",
        "libutils",
        "android.hardware.led@1.0",
    ],

}
```

3. Write hdil service

Next, use the corresponding functions to fill in the stub and set up the daemon. You can use passthrough and binder methods, examples:

hardware/interfaces/led/1.0/default/service.cpp

```
 #define LOG_TAG "android.hardware.led@1.0-service"

#include <android/hardware/led/1.0/ILed.h>
#include <hidl/LegacySupport.h>
#include "ledImpl.h"
using android::hardware::led::V1_0::ILed;
using android::hardware::led::V1_0::implementation::ledImpl;
using
android::hardware::defaultPassthroughServiceImplementati
on;
using android::hardware::configureRpcThreadpool;
using android::hardware::joinRpcThreadpool;
```

```
using android::sp;



int main() {
#if 0
//Passthrough dlopen so method
   return defaultPassthroughServiceImplementation<Iled>();
#else
//Binder way
 sp<ILed> service = new ledImpl();
 configureRpcThreadpool(1, true/*callerWillJoin*/);
  if(android::OK != service->registerAsService())
    return 1;
  joinRpcThreadpool();
#endif
}
```

4. Configure manifest.xml

add the code to the manifest.xm so that hwservicemanager finds the specified hidl service

About HIDL configures

device/<vendorxxx>/<devicexxx>/manifest.xml

```
<hal format="hidl">
  <name>android.hardware.led</name>
            <transport>hwbinder</transport>//hwbinder    or
passthrough (pass-through mode)
  <version>1.0</version>
  <interface>
   <name>ILed</name>
   <instance>default</instance>
  </interface>
 </hal>
```

5. Hidl client call

After hidl service is running, it can be called in two ways, C++ and Java, which is very convenient. Direct access via java saves jni.

5.1 Example of implementing java calling hidl service

Add the following to Android.mk:

```
LOCAL_JAVA_LIBRARIES += android.hardware.led-V1.0-java

```

```
/*
 LOCAL_STATIC_JAVA_LIBRARIES += android.hardware.led-V1.0-java-static
*/
```

Or add the following to Android.bp:

```
 shared_libs: [
 /*… */
  "android.hardware.led-V1.0-java",
 ],
```

The library also has a static version: android.hardware.led-V1.0-java-static.

Add the following to your Java file:

```
 import android.hardware.led.V1_0.ILed;
 ...
 //retry to wait until the service starts up if it is in the manifest
     ILed server = ILed.getService(/* retry */);//throws NoSuchElementException if not available
 server.on();
```

5.2 Example of implementing C++ calling hidl service

First add the HAL library to the makefile:

```
Make: LOCAL_SHARED_LIBRARIES += android.hardware.led@1.0
 Soong: shared_libs: [..., android.hardware.led@1.0]
```

Next, add the HAL header file:

```
 #include <android/hardware/led/1.0/ILed.h>
 …
//in code:
 sp<ILed> client = ILed::getService();
 client->on();
```

Below is my Demo clinet

```
 #define LOG_TAG "LED_CLINET"
#include <android/hardware/led/1.0/ILed.h>
#include <log/log.h>

using android::hardware::led::V1_0::ILed;
using android::hardware::led::V1_0::LedStatus;
```

```
using android::hardware::led::V1_0::BrightnessRange;
using android::hardware::hidl_vec;
using android::sp;



int main(){
    //BrightnessRange range;
    sp<ILed> service = ILed::getService();
    if( service == nullptr ){
        ALOGE("Can't find ILed service...");
        return -1;
    }
    ALOGE("ILed ON");
    service->on();

    ALOGE("ILed OFF");
    service->off();

    ALOGE("ILed set");
    service->set(LedStatus::LED_ON);

    ALOGE("ILed get");
    LedStatus ret = service->get();
    ALOGE("ILed get: %d",ret);

    service->getBrightnessRange([](bool
ret1,BrightnessRange range){
        ALOGE("ILed getBrightnessRange ret: %d",ret1);
        ALOGE("ILed getBrightnessRange Max:
%d",range.max);
        ALOGE("ILed getBrightnessRange Min:
%d",range.min);
    });
    int32_t array[] = {5, 6, 7};
    hidl_vec<int32_t> hv1 = std::vector<int32_t>(array, array
+ 3);
    bool ret2 = service->setBrightnessValue(hv1);
    ALOGE("ILed getBrightnessValue bool: %d",ret2);
    return 0;
}
```

6. github address

Those who need to see the code can go to my github, there are related hidl code, and C++ call examples

gitHub address link (https://github.com/anlory/LedHidl/)