

[登录](#) | [注册](#)

sadamoo的专栏

[目录视图](#)[摘要视图](#)[RSS 订阅](#)

个人资料



sadamoo

访问: 131455次

积分: 1705

等级: [BLOG > 4](#)

排名: 第16466名

原创: 7篇 转载: 211篇

译文: 0篇 评论: 5条

文章搜索

文章分类

[linux 设备驱动模型 \(13\)](#)[android camera \(28\)](#)[linux input \(3\)](#)[linux i2c \(12\)](#)[linux lcd \(1\)](#)[linux ipc \(3\)](#)[android input \(1\)](#)[linux 内存管理 \(19\)](#)[android multimedia \(36\)](#)[alsa \(15\)](#)[android audio \(4\)](#)[android class \(1\)](#)[android reboot \(2\)](#)[android miracast \(6\)](#)[linux socket \(5\)](#)[linux pipe \(2\)](#)[android wifi \(3\)](#)[mp4 \(3\)](#)[【公告】博客系统优化升级](#) [【收藏】Html5 精品资源汇集](#) [博乐招募开始啦](#)

Android Camera API2.0下全新的Camera FW/HAL架构简述

2016-03-03 12:00

295人阅读

[评论](#) [收藏](#) [分享](#)[分类](#): [android camera \(27\)](#)

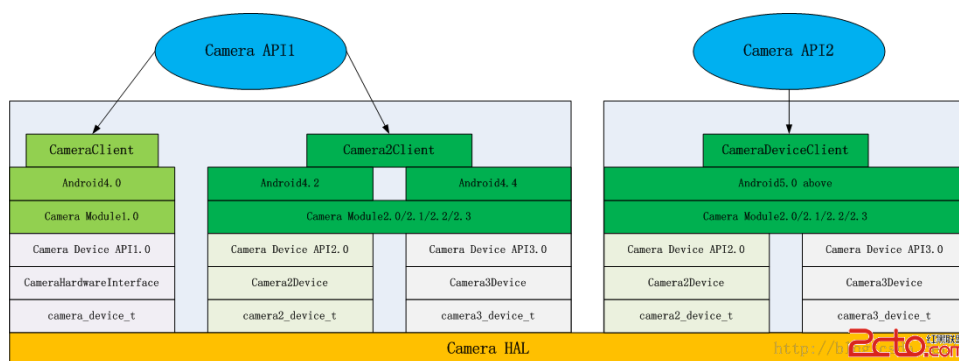
前沿:

前面博文大多少总结的是Camera HAL1到HAL3的系统架构，但这些架构对于Camera APP开发来说依旧还是处于Camera API1.0的标准。而随着Camera3、HAL3.0等的不断更新，Google先是在Framework中更改了整个架构从而去匹配Camera API1.0的处理逻辑，随着时间的推移，Google直接对Camera API进行了全新的升级，去除了原先的Camera.Java的相关接口，取而代之的是设计了Camera API2来完全匹配之前设计的Camera3以及HAL3，这样的好处是整个架构看起来会更简单。

本文主要简单的说明一下API2.0下Camera在Framework层中的处理逻辑，以及对比之前API1.0下他放弃了什么，同时增加了什么？

1. 全新的Camera API2.0

在API2.0中你再也看不得之前的startPreview、takePicture、AutoFocus等标准的操作接口，取而代之的是出现了大量涉及到CaptureRequest/CaptureResult相关的API，Google 在API Level21中即Android5.0版本中开始使用，并deprecate旧的Camera.java相关的接口。



2. AIDL技术在CameraService中的出现

AIDL是Android Java层实现C/S架构的一种方式，在Native Binder机制的帮助下，在Java层直接建立一种进程间通信。在Camera API2.0下可以看到大量的AIDL处理方式在Java层中出现，替代之前API1.0下都需要进入了Native层来完成通信。

对于CameraService而言，无论是哪种架构或者方式，都是应该满足下面的几个过程：

文章存档

2016年06月 (8)
2016年04月 (1)
2016年03月 (4)
2015年12月 (9)
2015年08月 (1)

展开

阅读排行

Android中基于NuPlayer (4525)
我对linux理解之i2c 二 (3983)
Android WifiDisplay分析 (3090)
Android WifiDisplay分析 (2999)
闲聊linux中的input设备 (2810)
Android4.0 input touch解 (2121)
device_register和驱动dri (2091)
android audio (2079)
Android WifiDisplay分析 (1975)
Bootloader之uBoot简介 (1959)

评论排行

Android4.0 input touch解 (1)
Linux内存寻址和内存管理 (1)
android多媒体框架之流媒体 (1)
WifiP2pService的启动以及 (1)
ALSA声卡驱动中的DAPM (1)
Linux的i2c驱动详解 (0)
基本的数据结构学习笔记 (0)
Linux设备驱动模型学习之 (0)
使用Camera2 替代过时 (0)
Linux设备驱动模型之底层 (0)

推荐文章

*Android RocooFix 热修复框架
* android6.0源码分析之Camera API2.0下的初始化流程分析
*Android_GestureDetector手势滑动使用
*Android MaterialList源码解析
*Android官方开发文档Training系列课程中文版: 创建自定义View之View的创建

最新评论

WifiP2pService的启动以及P2P全村人的希望: 您好, 我想请问一下, 如何能够设置自己手机发出去的device name。我在做一个小程序, 希望手机检...
ALSA声卡驱动中的DAPM详解之wsc_168: 您好: 现在正在移植wm8962的驱动, 遇到了一些问题, 向您请教一些问题。wm8962芯片已经...
Android4.0 input touch解析尹之梦: 我碰到的好像是这个触

(1) CameraService启动;

(2) 一个Client端通过CameraService Proxy连接到CameraService, 并获得一个CameraClient Proxy。后续通过CameraClient Proxy直接和CameraService来交互。

(3) Client要提供Callback实体接口到Service端, 即每个Service端的CameraClient都需要一个Callback Proxy来完成数据、消息的Callback。

无论Android怎么升级, Camera模块基本都处于这种工作模式下, 只是具体的实现方式不同而已。此外, 上面所提的到C/S架构基本都是通过Binder IPC来实现的。

传统的CameraService架构是在API1.0下请求Service在客户端创建一个Camera, 是一种很明显的C++层的C/S架构, 但在API2.0的架构下原先在Client层的Camera直接是交由Java层CameraDevice来维护, 通过AIDL的处理方式实现接口ICameraDeviceUser, 在Java层维护一个Camera proxy, 好处很明显是响应的速度会更快一些:

```
1  interface
2  ICameraDeviceUser
3  {
4      /**
5       *
6       * Keep up-to-date with frameworks/av/include/camera/camera2/ICameraDeviceUser
7       */
8      void
9      disconnect();
10
11     /**
12     * ints here are status_t
13
14     * //
15     * non-negative value is the requestId. negative value is status_t
16
17     int
18     submitRequest(in CaptureRequest request, boolean
19     streaming);
20
21     int
22     cancelRequest(int
23     requestId);
24
25     int
26     deleteStream(int
27     streamId);
28
29     /**
30     * //
31     * non-negative value is the stream ID. negative value is status_t
32
33     int
34     createStream(int
```

摸屏的问题，那到底该怎么改啊，求指教？

[Linux内存寻址和内存管理](#)
zq606: 学习了

```
width, int

height, int

format, in Surface surface);

int

createDefaultRequest(int

templateId, out CameraMetadataNative request);

int

getCameraInfo(out CameraMetadataNative info);

int

waitUntilIdle();

int

flush();

}
```

同样的我们看到CameraSevice在Android Java层处的ICameraService.AIDL文件：

```
1 interface ?
2 ICameraService
3 {
4     /**
5      *
6      * Keep up-to-date with frameworks/av/include/camera/ICameraService.h
7      */
8     int
9     getNumberOfCameras();
10
11     //
12     rest of 'int' return values in this file are actually status_t
13
14     int
15     getCameraInfo(int
16     cameraId, out CameraInfo info);
17
18     int
19     connect(ICameraClient client, int
20     cameraId,
21     String
22     clientPackageName,
23     int
```

```

    clientUid,
23
    //
24    Container for an ICamera object
25    out
    BinderHolder device);
26
27    int
28    connectPro(IProCameraCallbacks callbacks, int
29    cameraId,
30    String
31    clientPackageName,
32    int
33    clientUid,
34    //
    Container for an IProCameraUser object
    out
    BinderHolder device);

    int
    connectDevice(ICameraDeviceCallbacks callbacks, int
    cameraId,
    String
    clientPackageName,
    int
    clientUid,
    //
    Container for an ICameraDeviceUser object
    out
    BinderHolder device);

    int
    addListener(ICameraServiceListener listener);

    int
    removeListener(ICameraServiceListener listener);

    int
    getCameraCharacteristics(int
    cameraId, out CameraMetadataNative info);
}

```

3.Camera2Client消失, CameraDeviceClient出世

CameraDeviceClient可以说是替代了原先API1.0下升级后的Camera2Client, 此外在API2.0下是不允许Camera HAL Module 版本号为CAMERA_DEVICE_API_VERSION_1_0的, 至于选择使用的是Camera2Device还是Camera3Device来连接HAL3主要通过HAL的CAMERA_DEVICE_API_VERSION来指定。此外HAL中的VERSION必须要在CAMERA_DEVICE_API_VERSION_2_0以上才允许建立CameraDeviceClient。

4. Native消失了的各種Stream創建者

在之前的博文中，一直都在重點強調Camera2Client下出現了各種，目前看來這些只能停留在API1.0的世界里面了，隨着時間的推移Android版本的升級也許會慢慢的消逝，也就直接告訴我們HAL1.0的CameraHardwareInterface的實現方式將不復存在，當然一切還得取決於廠商的實現方式。

在這裡要重點說明的是在Camera2Client下出現了CallbackProcessor、FrameProcessor、StreamingProcessor等模塊，每個模塊負責處理不同的業務以及相關底層視頻圖像數據的處理與回調，其中對於數據的處理通過建立CPUConsumer與Surface的架構，更多的是以一種Consumer的角度實現對Buffer的queue與dequeue相關的操作，最終實現Camera3Device標準下的處理邏輯。

而在API2中在Framework層中，這些模塊將不再被使用，代替他們的是在Android5.0中的Java層中出現的各種Consumer，類似與Preview模式下的SurfaceFlinger在Java層中的surfaceview，這種模式是通過建立不同類型的Consumer，然後在Native層建立一個BufferQueue，並將這個BufferQueue的IGraphicBufferConsumer用於構建CPUConsumer，將IGraphicBufferProducer通過createStream給CameraDevice增加一個Stream。

當然本質上看起來兩者實現方式的機制是一樣的，都是需要create一個Stream，然後Stream需要對應的ANativeWindow類型的Surface，用於從HAL3中獲取數據，一旦獲取數據後和這個Surface綁定的Consumer就可以通過OnFrameAvailable()來接收處理buffer。下面的接口說明了在API2下對於不同數據處理模塊只需要get一個Surface後通過AIDL實現方式就可以創建一個stream接口，用於數據的接收。

```
1  status_t                                     ?
   CameraDeviceClient::createStream(int
2  width, int
3  height, int
4  format,
5
6                                     const
7  sp<igraphicbufferproducer>& bufferProducer)
8  {
9      ATRACE_CALL();
10     ALOGV("%s
(w = %d, h = %d, f = 0x%x), __FUNCTION__, width, height, format);
11
12     status_t
res;
13     if
14     ( (res = checkPid(__FUNCTION__) ) != OK) return
15     res;
16
17     Mutex::Autolock
18     icl(mBinderSerializationLock);
19
20     if
21     (bufferProducer == NULL) {
22         ALOGE("%s:
bufferProducer must not be null,
23         __FUNCTION__);
24         return
25         BAD_VALUE;
```

```
26     }
27     if
28     (!mDevice.get()) return
29     DEAD_OBJECT;
30
31     //
32     Don't create multiple streams for the same target surface
33     {
34         ssize_t
35         index = mStreamMap.indexOfKey(bufferProducer->asBinder());
36         if
37         (index != NAME_NOT_FOUND) {
38             ALOGW("%s:
39             Camera %d: Buffer producer already has a stream for
40             it
41             (ID
42             %zd),
43             mCameraId, index);__FUNCTION__,
44             return
45         ALREADY_EXISTS;
46     }
47     .....
48
49     int32_t
50     disallowedFlags = GraphicBuffer::USAGE_HW_VIDEO_ENCODER |
51                     GRALLOC_USAGE_RENDERSCRIPT;
52     int32_t
53     allowedFlags = GraphicBuffer::USAGE_SW_READ_MASK |
54                  GraphicBuffer::USAGE_HW_TEXTURE
55                  |
56                  GraphicBuffer::USAGE_HW_COMPOSER;
57
58     bool
59     flexibleConsumer = (consumerUsage & disallowedFlags) == 0
60     &&
61         (consumerUsage
62         & allowedFlags) != 0;
63
64     sp<ibinder>
65     binder;
66
67     sp
68     anw;
69
70     if
71     (bufferProducer != 0)
72     {
73         binder
74         = bufferProducer->asBinder();
75     }
```

```

        anw
    = new
    Surface(bufferProducer, useAsync); //创新一个本地的surface, 用于Product

    }

    //
    TODO: remove w,h,f since we are ignoring them

    .....

    res
    = mDevice->createStream(anw, width, height, format, &streamId); //t

    return
    res;
}

</anativewindow></ibinder></igraphicbufferproducer>

```

在API2.0下可以看到在Android Java层中提供了不同的Module来处理不同的视频图像数据，这个过程是很类似与Camera2Client下的各种Processor模块的，只是后者是将数据处理打包后再返回到APP中，而前者是直接由Java层的不同模块来异步的响应并直接处理不同类型数据流的到来，如PREVIEW、RRCORD、STILL_CAPTURE、VIDEO_SNAPSHOT、ZERO_SHUTTER_LAG等不同模式的数据流将由MediaRecorder、SurfaceView、ImageReader等来直接处理，总体来说效率会更佳。

5. FrameProcessorBase依然存在 该类在旧版本中被FrameProcessor用来处理3A相关的信息，主要是Callback每一帧的ExtraResult到APP，也就是3A相关的数据信息。这也是在API1和API2中唯一都需要手动CallBack的模块，其余的数据流都是被上述提到的模块自动处理，其中实现的方式如2小节(3)所描述，其中采用ICameraDeviceCallbacks将每一视频帧数据回传：

```

1  interface
2  ICameraDeviceCallbacks
3  {
4      /**
5       *
6       Keep up-to-date with frameworks/av/include/camera/camera2/ICameraI
7       */
8
9      oneway
10 void onCameraError(int
11 errorCode);

      oneway
void onCameraIdle();

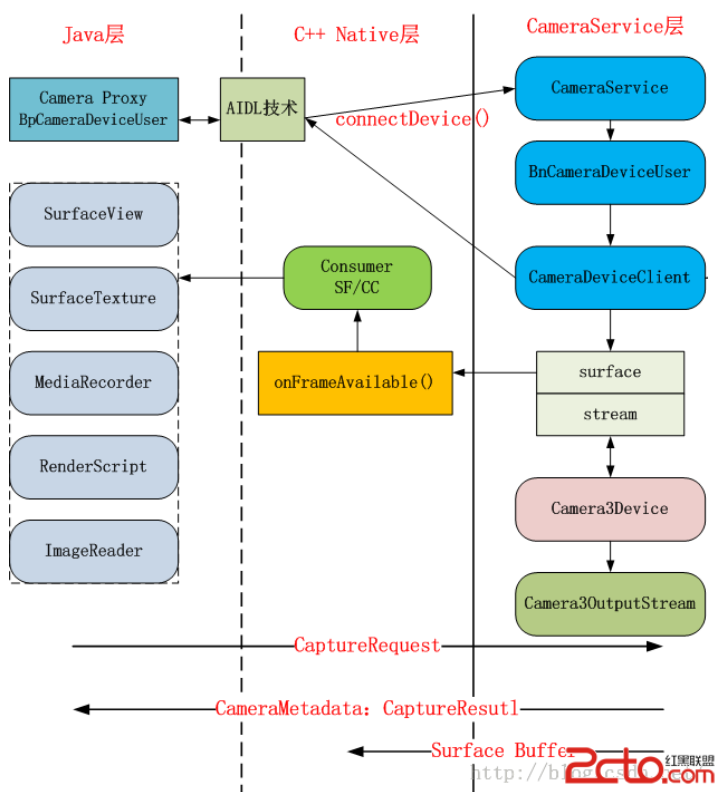
      oneway
void onCaptureStarted(int
requestId, long

```

```
timestamp);  
    oneway  
    void  
    onResultReceived(int  
        requestId, in CameraMetadataNative result);  
}
```

6 小结, 整个API2下Camera3的架构简图

最新的Camera API2和Camera3之间的结构简图



顶 踩
0 0

上一篇 Android Camera HAL V3 Vendor Tag及V1, V3参数转换

下一篇 定语从句的用法讲解

我的同类文章

android camera (27)

- 使用Camera2 替代过时的C... 2016-06-09 阅读 113
 - Android Camera从Camera ... 2016-06-02 阅读 67
 - Android Camera API2中采... 2016-06-02 阅读 77
 - Android5.1中surface和Cpu... 2016-06-01 阅读 56
 - Android Camera API2中采... 2016-03-03 阅读 258
- Android4.2.2 Camer系统架... 2016-06-03 阅读 46
 - Android Camera HAL3中预... 2016-06-02 阅读 84
 - Android Camera HAL3中拍... 2016-06-02 阅读 51
 - Android Camera HAL V3 Ve... 2016-03-03 阅读 137
 - Android Camera HAL3中预... 2016-03-01 阅读 356
- 更多文章

参考知识库

**Android**知识库
12836 关注 | 1500 收录

**大型网站架构**知识库
1931 关注 | 532 收录

**Java EE**知识库
1781 关注 | 618 收录

**Java SE**知识库
9984 关注 | 454 收录

**Java Web**知识库
10300 关注 | 1074 收录

**Docker**知识库
2399 关注 | 187 收录

猜你在找

- Android必备的Java基础知识(二)

Android开发精品课程【Java核心知识】

【Android APP开发】Android高级商业布局快速实现

大数据系统架构

开源大数据技术架构设计
- android camera HAL v30中元数据及其控制

Android Camera的HAL接口

Android Camera portingHAL层移植

android camera HAL v30详细介绍一

Android Camera HAL设计初步



查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题

Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack

VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery

BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity

Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC

coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo

Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr

Angular Cloud Foundry Redis Scala Django Bootstrap

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved 