sadamoo的专栏

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

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(0) 收藏 举报

分类： android camera（27）

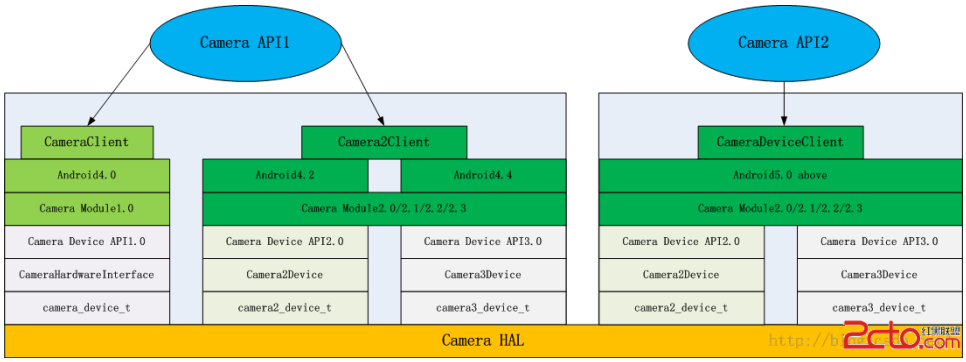
前沿：

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Framewrok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ADIL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:

（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：

```
interface
ICameraDeviceUser
{
    /**
     *
 Keep up-to-date with frameworks/av/include/camera/camera2/ICameraI
     */
    void
disconnect();

    //
 ints here are status_t

    //
 non-negative value is the requestId. negative value is status_t
    int
submitRequest(in CaptureRequest request, boolean
streaming);

    int
cancelRequest(int
requestId);

    int
deleteStream(int
streamId);

    //
 non-negative value is the stream ID. negative value is status_t
    int
createStream(int
```

```
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:

```
interface
ICameraService
{
    /**
     *
 Keep up-to-date with frameworks/av/include/camera/ICameraService.h
     */
    int
getNumberOfCameras();

    //
 rest of 'int' return values in this file are actually status_t

    int
getCameraInfo(int
cameraId, out CameraInfo info);

    int
connect(ICameraClient client, int
cameraId,
                    String
 clientPackageName,
                    int
```

```
clientUid,
                    //
 Container for an ICamera object
                    out
 BinderHolder device);


    int
connectPro(IProCameraCallbacks callbacks, int
cameraId,
                                String
 clientPackageName,
                                int
clientUid,
                                //
 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消失了的各种Stream创建者

在之前的博文中，一直都在重点强调Camera2Client下出现了各种，目前看来这些只能停留在API1.0的世界里面了，随着时间的推移Android版本的升级也许会慢慢的消逝，也就直接告诉我们HAL1.0的CameraHardwareInterface的实现方式将不复存在，当然一切还得取决于厂商的实现方式。

在这里要重点说明的是在Camera2Client下出现了CallbackProcessor、FrameProcessor、StreamingProcessor等模块，每个模块负责处理不同的业务以及相关底层视频图像数据的处理与回调，其中对于数据的处理通过建立CPUConsumer与Surface的架构，更多的是以一种Consumer的角度实现对Buffer的queue与dequeue相关的操作，最终实现Camera3Device标准下的处理逻辑。

而在API2中在Framework层中，这些模块将不再被使用，代替他们的是在Android5.0中的Java层中出现的各种Consumer，类似与Preview模式下的SurfaceFlinger在Java层中的surfaceview，这种模式是通过建立不同类型的Consumer，然后在Native层建立一个BufferQueue，并将这个BufferQueue的IGraphicBufferConsumer用于构建CPUConsumer，将IGraphicBufferProducer通过createStream给CameraDevice增加一个Stream。

当然本质上看起来两者实现方式的机制是一样的，都是需要create一个Stream，然后Stream需要对应的ANativeWindow类型的Surface，用于从HAL3中获取数据，一旦获取数据后和这个Surface绑定的Consumer就可以通过OnFrameAvailable()来接收处理buffer。下面的接口说明了在API2下对于不同数据处理模块只需要get一个Surface后通过AIDL实现方式就可以创建一个stream接口，用于数据的接收。

```
?
1  status_t
    CameraDeviceClient::createStream(int
2
   width, int
3
   height, int
4
   format,
5
                          const
6
   sp<igraphicbufferproducer>& bufferProducer)
7
   {
8
       ATRACE_CALL();
9
       ALOGV(%s
10  (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;
```

```
    }
    if (!mDevice.get()) return DEAD_OBJECT;

    // Don't create multiple streams for the same target surface
    {
        ssize_t index = mStreamMap.indexOfKey(bufferProducer->asBinder());
        if (index != NAME_NOT_FOUND) {
            ALOGW("%s: Camera %d: Buffer producer already has a stream for it
                    (ID %zd),
                    __FUNCTION__, mCameraId, index);
            return ALREADY_EXISTS;
        }
    }

   ..............

    int32_t disallowedFlags = GraphicBuffer::USAGE_HW_VIDEO_ENCODER |
                              GRALLOC_USAGE_RENDERSCRIPT;
    int32_t allowedFlags = GraphicBuffer::USAGE_SW_READ_MASK |
                           GraphicBuffer::USAGE_HW_TEXTURE |
                           GraphicBuffer::USAGE_HW_COMPOSER;
    bool flexibleConsumer = (consumerUsage & disallowedFlags) == 0 &&
            (consumerUsage & allowedFlags) != 0;

    sp<ibinder> binder;
    sp anw;
    if (bufferProducer != 0) {
        binder = bufferProducer->asBinder();
```

```
        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);//创


    return
res;
}
</anativewindow></ibinder></igraphicbufferproducer>
```

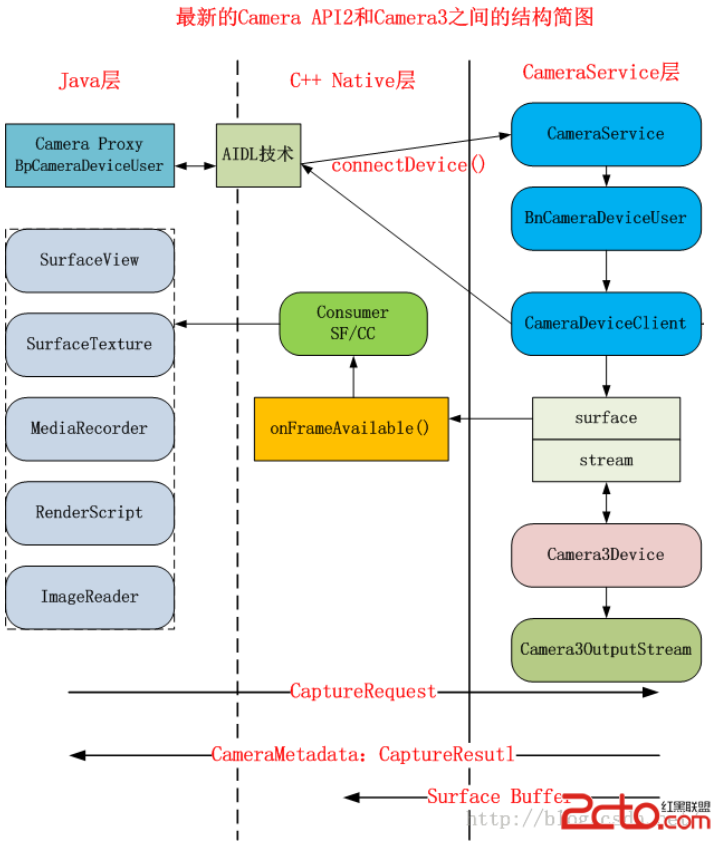
在API2.0下可以看到在Android Java层中提供了不同的Module来处理不同的视频图像数据，这个过程是很类似与Camera2Client下的各种Processor模块的，只是后者是将数据处理打包后再返回到APP中，而前者是直接由Java层的不同模块来异步的响应并直接处理不同类型数据流的到来，如PREVIEW、RRCORD、STILL_CAPTURE、VIDEO_SNAPSHOT、ZERO_SHUTTER_LAG等不同模式的数据流将由MediaRecoder、SurfaceView、ImageReader等来直接处理，总体来说效率会更佳。

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：

```
                                                                    ?
interface
ICameraDeviceCallbacks
{
    /**
     *
 Keep up-to-date with frameworks/av/include/camera/camera2/ICameraD
     */


    oneway
void
onCameraError(int
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



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

猜你在找

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