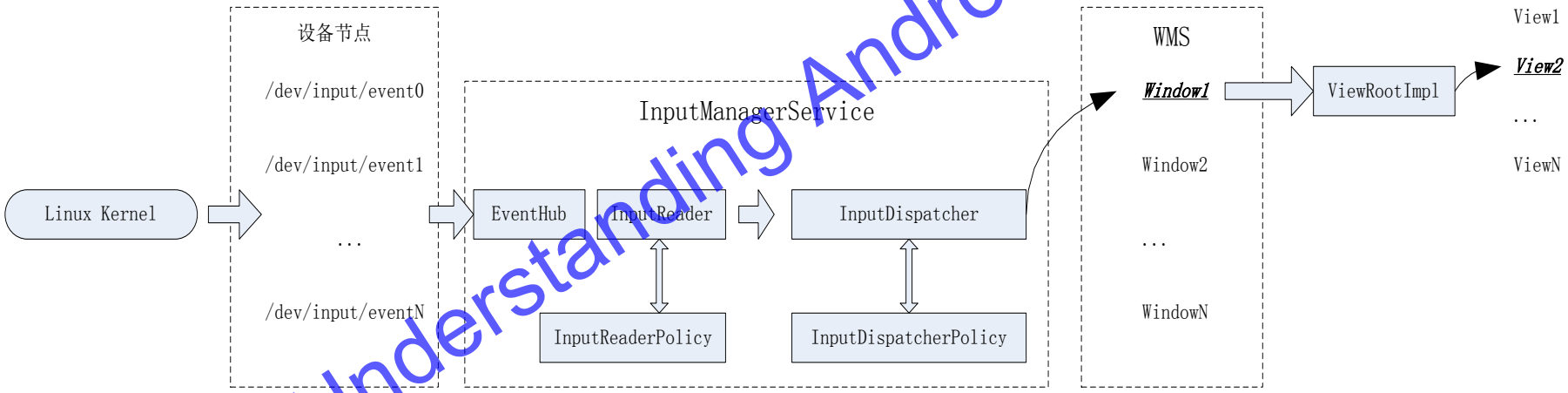
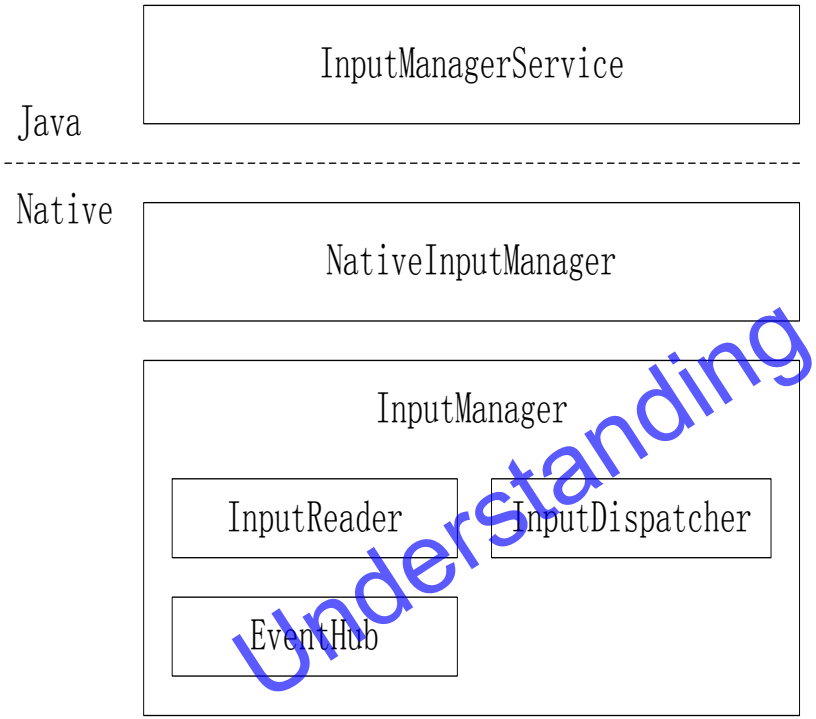
输入系统

输入系统简介



IMS的组成结构

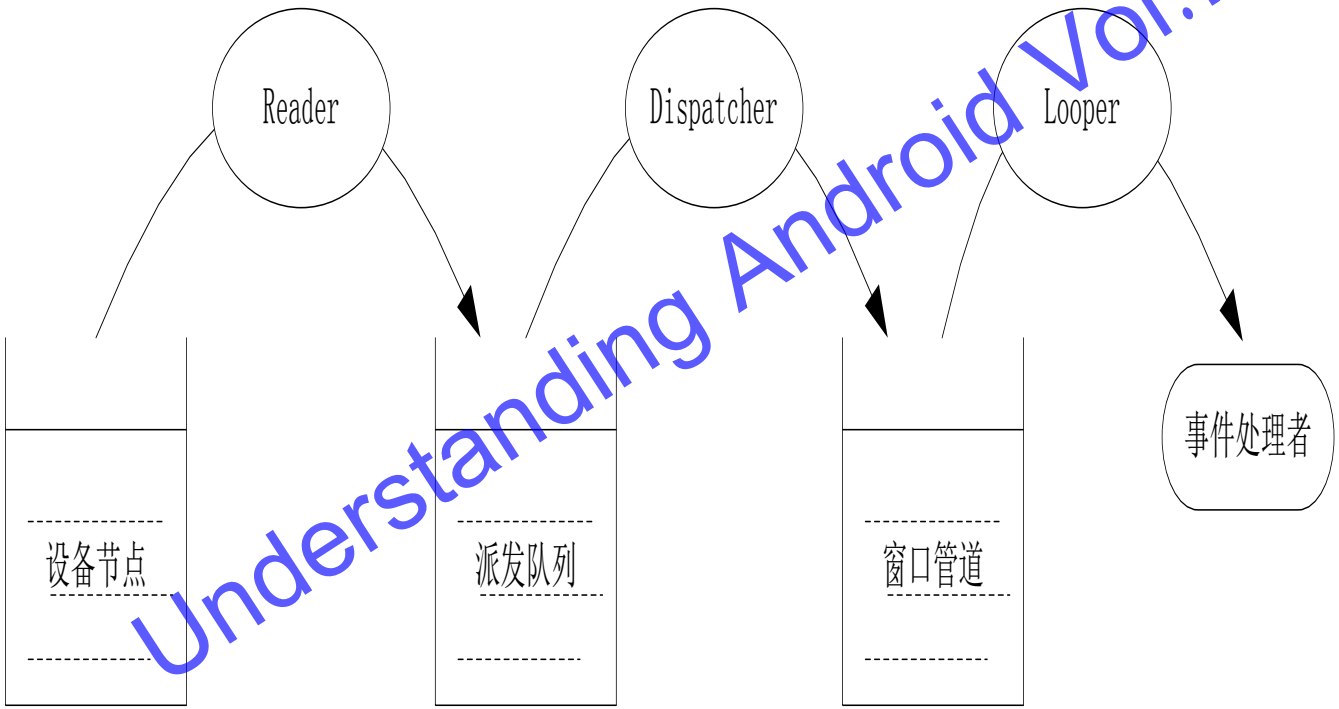


Java层的IMS的主要工作是为ReaderPolicy与DispatcherPolicy提供实现，以及与Android其他系统服务进行协作，其中最主要的协作者是WMS。

NativeInputManager位于IMS的JNI层，负责Native层的组件与Java层的IMS的相互通讯。同时，它为InputReader及InputDispatcher提供了策略请求的接口。策略请求被它转发给Java层的IMS，由IMS进行最终的定夺。

InputManager是InputReader与InputDispatcher的运行容器，它创建了两个线程分别承载InputReader与InputDispatcher的运行。

输入系统的工作原理



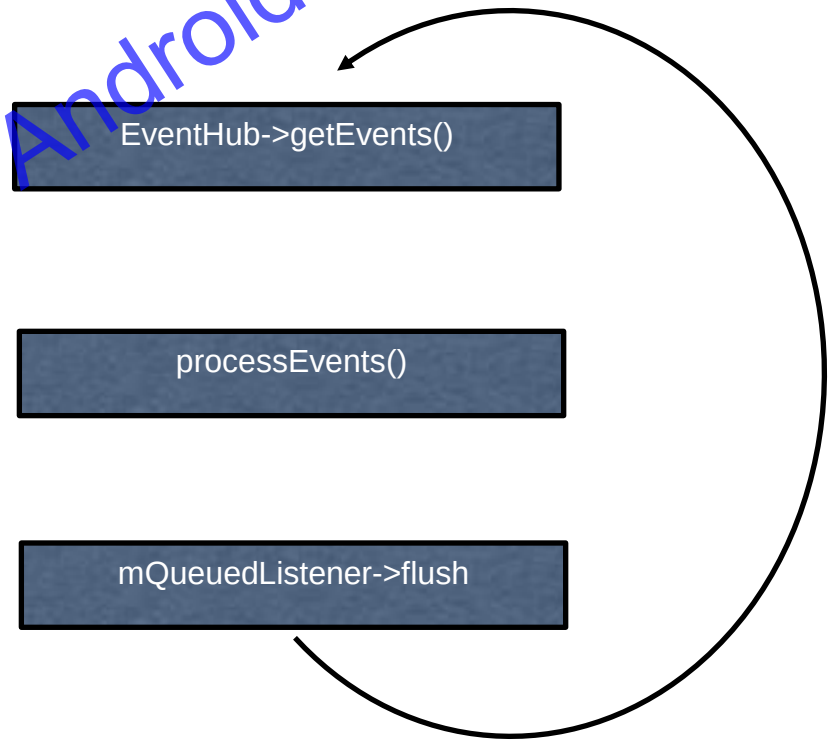
INotify与Epoll

- 通过inotify_init()创建一个inotify对象。
- 通过inotify_add_watch将一个或多个监听添加到inotify对象中。
- 通过read()函数从inotify对象中读取监听事件。当没有新事件发生时，inotify对象中无任何可读数据。

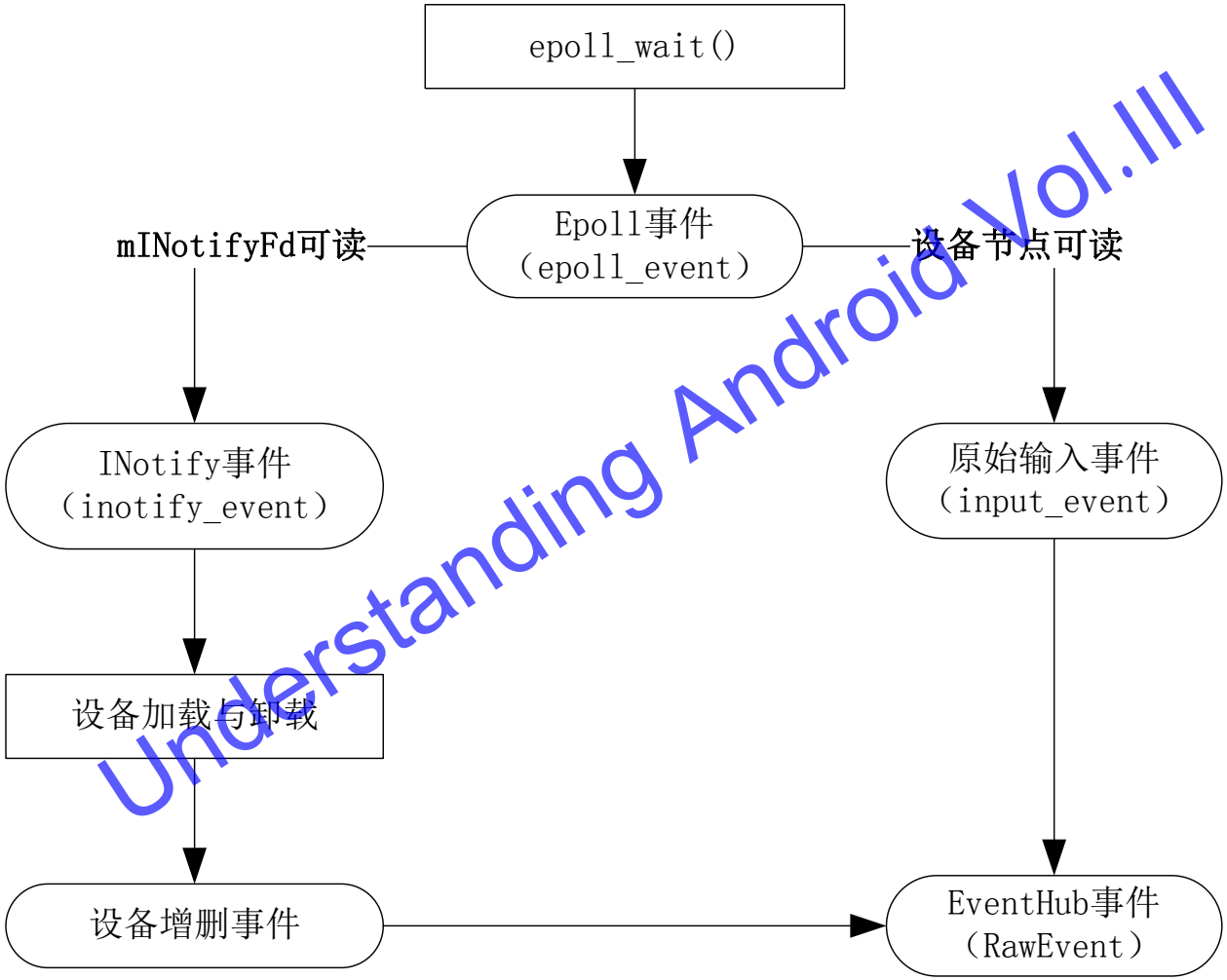
- epoll_create(int max_fds)：创建一个epoll对象的描述符，之后对epoll的操作均使用这个描述符完成。max_fds参数表示了此epoll对象可以监听的描述符的最大数量。
- epoll_ctl (int epfd, int op, int fd, struct epoll_event *event)：用于管理注册事件的函数。这个函数可以增加/删除/修改事件的注册。
- int epoll_wait(int epfd, struct epoll_event * events, int maxevents, int timeout)：用于等待事件的到来。当此函数返回时，events数组参数中将会包含产生事件的文件描述符。

InputReader的线程循环

```
bool InputReaderThread::threadLoop() {
    mReader->loopOnce(); // 执行
InputReader的loopOnce()函数
    return true;
}
```



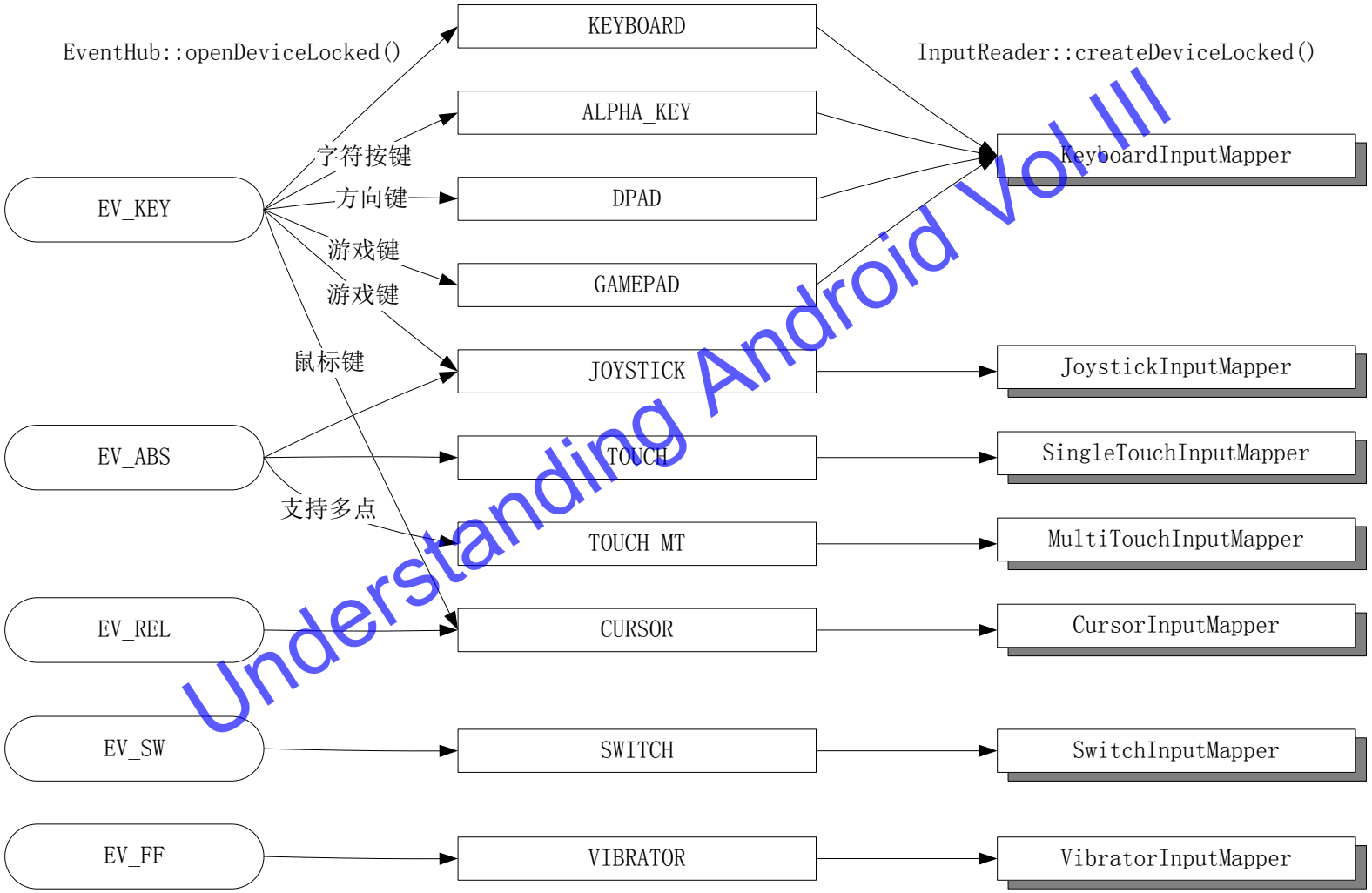
EventHub->getEvent()工作原理



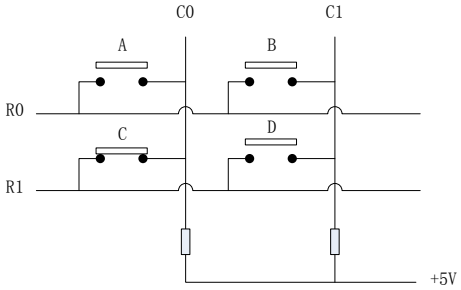
InputReader的事件处理

- 设备增删事件
- 输入事件
  - EV_KEY
  - EV_ABS
  - EV_REL
  - EV_SW

InputReader的事件处理



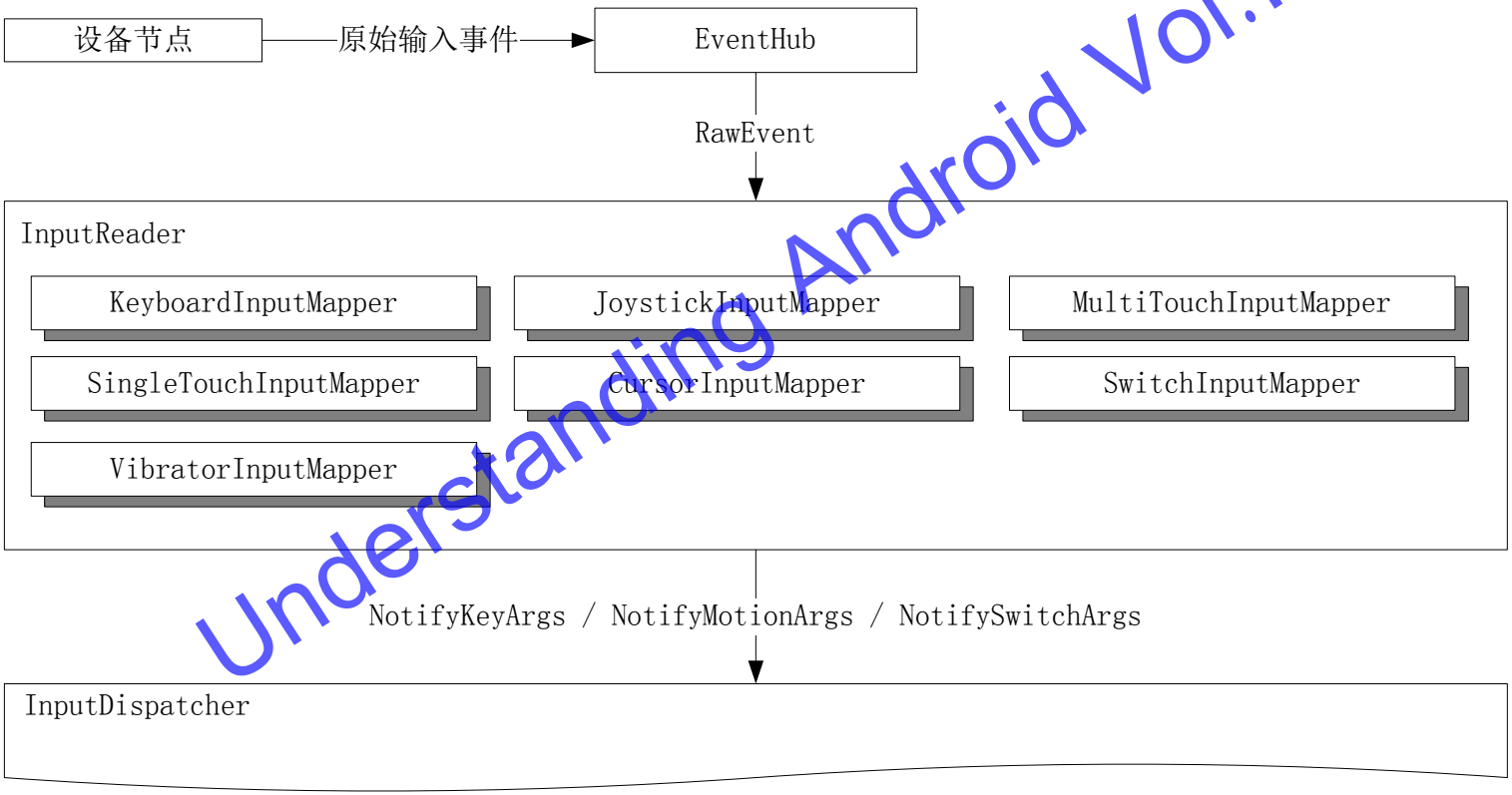
KeyboardInputMapper



键盘布局文件(.kl)


```
NotifyKeyArgs args(when, getDeviceId(), mSource, policyFlags,
        down ? AKEY_EVENT_ACTION_DOWN : AKEY_EVENT_ACTION_UP,
        AKEY_EVENT_FLAG_FROM_SYSTEM, keyCode, scanCode, newMetaState, downTime);
  getListener()->notifyKey(&args);
```

InputReader总结



InputDispatcher

Understanding Android Vol.III

事件被抛弃

- DROP_REASON_POLICY
- DROP_REASON_APP_SWITCH
- DROP_REASON_BLOCKED
- DROP_REASON_DISABLED
- DROP_REASON_STALE

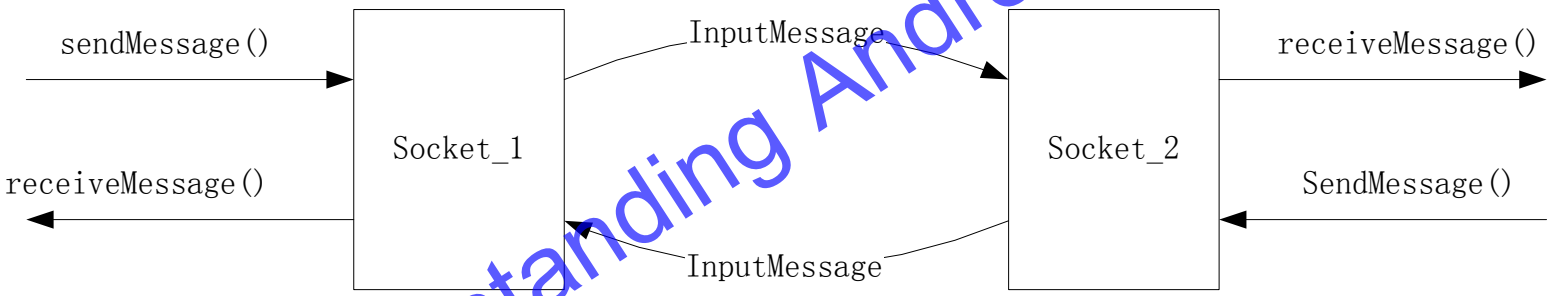
DispatcherPolicy

- interceptMotionBeforeQueueing() （Reader线程）
- interceptKeyBeforeQueueing() (Reader线程）
  - POLICY_FLAG_WOKE_HERE
  - POLICY_FLAG_BRIGHT_HERE
  - POLICY_FLAG_PASS_TO_USER
- interceptKeyBeforeDispatching() (Dispatcher线程）
  - < 0, 策略已经消费这个事件
  - = 0, 立即派发此事件
  - > 0, 等待片刻再向DispatcherPolicy询问派发策略

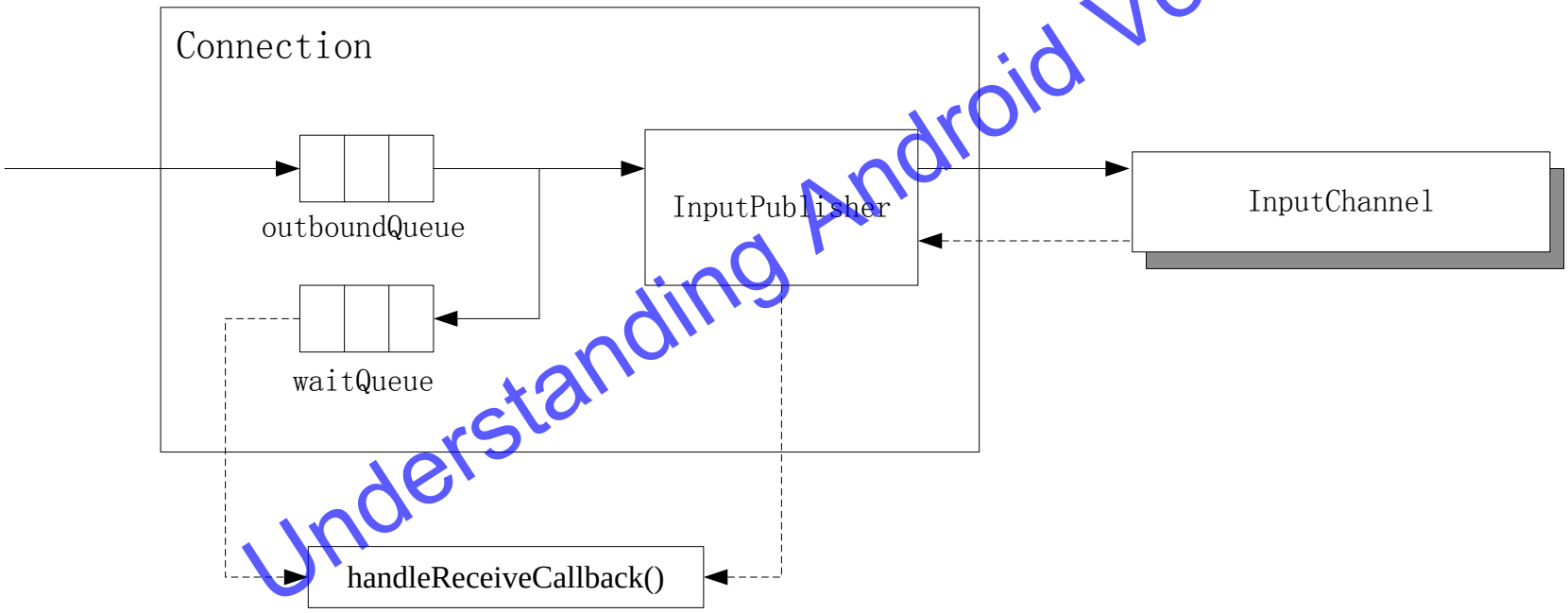
InputFilter

- Sample: Talkback
- 使用者可以通过setInputFilter()函数将自己的IInputFilter对象设置给IMS，从此便可以开始进行输入事件的监听操作
- IMS认为一旦事件被一个InputFilter截获过，则这个事件便被丢弃了

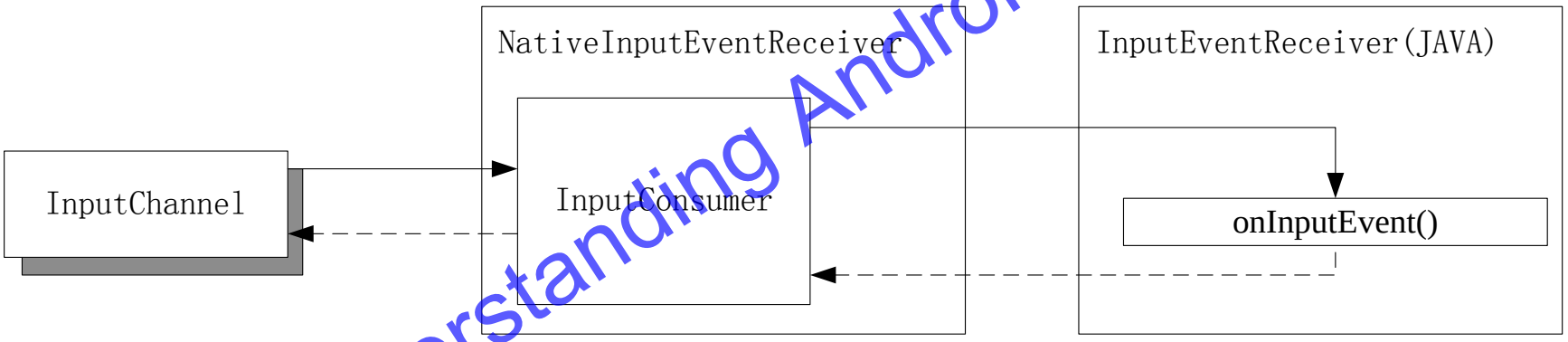
InputChannel



服务端 (InputDispatcher)



客户端（应用进程）



ANR

- 是否可以接受事件？
  - InputPublisher是否阻塞
  - Connection两个队列的状态
    - 必须都为空(Key)
    - 发送队列队首事件在0.5秒内有反馈（Motion)
- 一旦收到来自客户端对事件的反馈，ANR计时重置

焦点窗口的确定

- 窗口不会即将被移除。
- 窗口可见。
- 没有指定FLAG_NOT_FOCUSABLE选项。

以软件方式模拟用户操作

- Sendevent
- InputManagerService.injectInputEvent()

Thanks

Understanding Android Vol.III

for more information, please visit:
http://www.thundersoft.com/

contact us:
biz@thundersoft.com
+86-10-62662686

Address:
4th floor, Taixiang Building 1A#, Longxiang Road, Haidian District Beijing, China, 100191