Android—广播（Broadcast）—广播的注意事项及相关问题分析

原创 怒放的程序员 2017-08-20 10:50:43

1.Intent.FLAG_RECEIVER_REGISTERED_ONLY

从前面两节的分析可以知道，对于静态注册的广播接收者，如果其所在进程不存在，ActivityManagerService是会先把它所在进程先启动，然后将广播发送给此广播接收者，鉴于此机制，有的应用为了保证自己进程被杀死后能被重新创建，所以会静态注册一些系统广播（例如电池电量变化的广播）的接收者，这样可以达到重启的目的，早期的Android版本上确实可以这么做。

后来Google应该也是意识到这种漏洞，所以对于一些系统广播，在其发送的时候，添加上了*FLAG_RECEIVER_REGISTERED_ONLY*的flag，这样就保证了这类广播只能通过动态注册的广播接收者来接收,例如下面是发送电池电量变化ACTION_BATTERY_CHANGED的广播时添加了FLAG_RECEIVER_REGISTERED_ONLY。

BatteryService.java

```
final Intent intent = new Intent(Intent.ACTION_BATTERY_CHANGED);
intent.addFlags(Intent.FLAG_RECEIVER_REGISTERED_ONLY
        | Intent.FLAG_RECEIVER_REPLACE_PENDING);
...
ActivityManagerNative.broadcastStickyIntent(intent, null, UserHandle.USER_ALL);
```

所以如果你也只想让你的广播只能动态注册的接收者收到，只要调用Intent.addFlags(Intent.FLAG_RECEIVER_REGISTERED_ONLY即可。

下面是一些常见的只支持动态注册接收者接收的系统广播：

广播名称	广播含义
Intent.ACTION_TIME_TICK	表示当前时间已经改变了，每分钟发送一次
Intent.ACTION_BATTERY_CHANGED	当前手机电量发生改变时发送，可以读取当前充电状态，电池电量等其他相关信息
Intent.ACTION_SCREEN_ON	屏幕被点亮时发送
Intent.ACTION_SCREEN_OF	屏幕息屏时发送
Intent.ACTION_CONFIGURATION_CHANGED	当前系统的Configuration发生改变时，例如屏幕方向，系统语言等发生改变了
“android.intent.action.ANR”	出现anr时发送的广播

2.前台广播和后台广播

从前面的分析可以知道，AMS是先通过把广播添加到对应的队列中，然后异步方式从队列里面取广播然后真正的开始发送广播，而目前AMS中有两个队列，在AMS的构造方式中有如下的部分：

ActivityManagerService.java

```
static final int BROADCAST_FG_TIMEOUT = 10*1000;
static final int BROADCAST_BG_TIMEOUT = 60*1000;

mFgBroadcastQueue = new BroadcastQueue(this, mHandler,
                "foreground", BROADCAST_FG_TIMEOUT, false);
mBgBroadcastQueue = new BroadcastQueue(this, mHandler,
                "background", BROADCAST_BG_TIMEOUT, true);
```

BroadcastQueue.java

```
BroadcastQueue(ActivityManagerService service, Handler handler,
            String name, long timeoutPeriod, boolean allowDelayBehindServices) {
    mService = service;
    mHandler = new BroadcastHandler(handler.getLooper());
    mQueueName = name;
    mTimeoutPeriod = timeoutPeriod;
    mDelayBehindServices = allowDelayBehindServices;
}
```

AMS构造方法中构造了两个队列，一个前台队列，一个后台队列。结合其构造方法可知，两者区别在于一个是设置的广播超时时间不同，前台是10s，后台是60s，另外一个是是否要等待后台服务处理完，前台广播是不用等待的，后台广播需要等待。

想要让广播能放到前台队列中，只需调用Intent.addFlags(Intent.FLAG_RECEIVER_FOREGROUND)即可。由于一般普通广播都默认是放在后台队列中的，所以会导致后台队列中的广播比较多，相对处理时间较长，所以为了能够让发送的广播优先被处理，将其设置为前台广播也算一个解决办法。

3.广播的接收延迟问题

广播使用起来成本比较低，用途也比较广泛，特别是可以突破进程限制，让所有相关的应用都可以收到广播，缺点也很明显，就是不够及时，谁也没法保证一定能在某个时间段内一定能收到某个广播，也就是广播的接收延时问题不可避免。

既然讲到广播接收延时，不得不先说下BroadcastRecord中关于记录广播的几个时间点：

变量	含义	记录的时机
enqueueClockTime	一次广播插入到广播队列时的时间点	调用enqueueParallelBroadcastLocked()或者enqueueOrderedBroadcastLocked()时被赋值的，值等于System.currentTimeMillis()
dispatchTime	一次广播从广播队列中被取出，准备开始发送	调用processNextBroadcast()时被赋值的，值等于SystemClock.uptimeMillis()
dispatchClockTime	含义同dispatchTime	跟dispatchTime一同被赋值的，只不过值等于System.currentTimeMillis()
receiverTime	一次广播中，开始派发给其中每个接收者时的时间点，主要记录的是有序广播的情况	一处时调用processNextBroadcast()时被赋值的，一处是广播超时时被赋值的，值等于SystemClock.uptimeMillis()
finishTime	一次广播完成时的时间点	调用addBroadcastToHistoryLocked()时被赋值，值等于SystemClock.uptimeMillis()

针对表格内容，说明一下，所谓一次广播，是指用BroadcastRecord对象来表示的，该对象会被插入到广播队列中，插入的时间点存放在enqueueClockTime中，等到准备从队列中取出BroadcastRecord时，将当前的时间点记录在dispatchTime和dispatchClockTime中，每次广播的接收者可能有很多，对于动态注册的无序广播，由于不用关心广播是否都送到，所以不用统计receiverTime， 而对于有序广播来说，当从BroadcastRecord取出一个接收者时，将当前的时间点记录在receiverTime中，最后等到所有接收者都送达完毕，一次广播也就结束了，所以将此时结束的时间点记录在finishTime中。

另外也发现了，在将广播从队列中取出时，分别用了dispatchTime和dispatchClockTime来同时记录，两者获取的值有所不同，之所以要两个变量来记录，是因为dispatchTime会用来评定是否广播超时，而dispatchClockTime会最终记录到广播历史中，所谓广播历史是指AMS内部会将近期发送的广播保存起来，方便debug。

既然提供了这么多的时间相关的成员，那怎么才能显示出来了，方便用来调试呢，这里就要看下面讲的通过dump来打印相关信息了。只有把相关信息打印出来了，才能方便我们断定到底广播超时在哪里，是由谁引起的。

4.dump广播相关的信息

adb shell dumpsys activity b [packagename]

此命令主要是打印应用动态注册的广播接收者，可以通过此命令得知某个应用注册了哪些广播的监听，同时对于之前已经接收过的广播的一些时间信息，方便调试是否接收超时。如果命令后面不接包名，则默认打印手机所有应用的广播注册信息。

下面是打印微信的一些广播注册信息。

```
{ ~ }  » adb shell dumpsys activity b com.tencent.mm
ACTIVITY MANAGER BROADCAST STATE (dumpsys activity broadcasts)
  //表明下面是打印应用通过动态注册的广播接收者，数据来源是AMS中的mRegisteredReceivers，下面依次遍历mRegisteredReceivers，打印
  Registered Receivers:
  //打印具体某个ReceiverList的信息
  * ReceiverList{de21349 8100 com.tencent.mm:push/10119/u0 remote:c1e4550}
    //打印注册此广播接收者的进程的信息，可以获取pid，uid等信息
    app=8100:com.tencent.mm:push/u0a119 pid=8100 uid=10119 user=0
    //打印IntentFilter的信息
    Filter #0: BroadcastFilter{e98a44e}
      //打印具体的需要监听的广播
      Action: "com.tencent.mm.WatchDogPushReceiver"
      AutoVerify=false

  * ReceiverList{a9160e1 8155 com.tencent.mm/10119/u0 remote:c5a6a48}
    app=8155:com.tencent.mm/u0a119 pid=8155 uid=10119 user=0
    Filter #0: BroadcastFilter{a70fe06}
      Action: "android.intent.action.PACKAGE_ADDED"
      Scheme: "package"
      AutoVerify=false

    。。。。//中间内容雷同，省略了

 //下面也是打印应用通过动态注册的广播接收者，不过数据来源是AMS的mReceiverResolver，里面存储的是以BroadcastFilter为单位
  Receiver Resolver Table:
    //打印InterFilter设置了Schemes的情况
    Schemes:
        //根据具体的Scheme分类打印，将有相同Schemes的BroadcastFilter都打印出来
        package:
          BroadcastFilter{a70fe06 u0 ReceiverList{a9160e1 8155 com.tencent.mm/10119/u0 remote:c5a6a48}}
```

```
          BroadcastFilter{a977675 u0 ReceiverList{483c9ac 16172 com.tencent.mm:support/10119/u0 remote:3e31b5f}}
          BroadcastFilter{64ecd4 u0 ReceiverList{6e3b227 16310 com.tencent.mm:tools/10119/u0 remote:905bce6}}
        file:
          BroadcastFilter{a977675 u0 ReceiverList{483c9ac 16172 com.tencent.mm:support/10119/u0 remote:3e31b5f}}
          BroadcastFilter{64ecd4 u0 ReceiverList{6e3b227 16310 com.tencent.mm:tools/10119/u0 remote:905bce6}}

    //打印InterFilter没有设置任何相关data的情况，可以通过此处了解到一个应用中到底有多少filter都添加了同一类广播的监听
    Non-Data Actions:
        android.intent.action.SCREEN_OFF:
          BroadcastFilter{18ffed0 u0 ReceiverList{d801e93 8155 com.tencent.mm/10119/u0 remote:e4fa782}}
          BroadcastFilter{2add5b7 u0 ReceiverList{17af4b6 8155 com.tencent.mm/10119/u0 remote:c234451}}
          BroadcastFilter{8d8a7fe u0 ReceiverList{6b617b9 16172 com.tencent.mm:support/10119/u0 remote:cc6d580}}
          BroadcastFilter{6584f41 u0 ReceiverList{2089628 16310 com.tencent.mm:tools/10119/u0 remote:e999c4b}}
        android.hardware.usb.action.USB_DEVICE_ATTACHED:
          BroadcastFilter{605e699 u0 ReceiverList{a0b13e0 16172 com.tencent.mm:support/10119/u0 remote:af51de3}}
          BroadcastFilter{96a4888 u0 ReceiverList{e431c2b 16310 com.tencent.mm:tools/10119/u0 remote:ce03c7a}}
        com.tencent.mm.ui.ACTION_DEACTIVE:
          BroadcastFilter{35625d2 u0 ReceiverList{2ee585d 8155 com.tencent.mm/10119/u0 remote:d2e234}}
        android.intent.action.ACTION_POWER_DISCONNECTED:
          BroadcastFilter{18ffed0 u0 ReceiverList{d801e93 8155 com.tencent.mm/10119/u0 remote:e4fa782}}
          BroadcastFilter{823c7de u0 ReceiverList{e7c4b19 8155 com.tencent.mm/10119/u0 remote:f823a60}}
          BroadcastFilter{2add5b7 u0 ReceiverList{17af4b6 8155 com.tencent.mm/10119/u0 remote:c234451}}
        。。。。//中间内容雷同，省略了

  //打印一些之前跟该应用相关的广播过的广播，从这里可以获取到广播的相关时间点，方便调试
  Historical broadcasts [background]:
  Historical Broadcast background #9:
    BroadcastRecord{ec22df u0 com.tencent.mm.plugin.report.service.KVCommCrossProcessReceiver} to user 0
    Intent { act=com.tencent.mm.plugin.report.service.KVCommCrossProcessReceiver flg=0x10 (has extras) }
      targetComp: {com.tencent.mm/com.tencent.mm.plugin.report.service.KVCommCrossProcessReceiver}
      extras: Bundle[mParcelledData.dataSize=884]
    caller=com.tencent.mm 8100:com.tencent.mm:push/u0a119 pid=8100 uid=10119
    enqueueClockTime=Mon Jul 10 19:34:23 GMT+08:00 2017 dispatchClockTime=Mon Jul 10 19:34:23 GMT+08:00 2017
    enqueueTime=-48s780ms dispatchTime=-48s779ms finishTime=-48s763ms
    Total: +17ms Waiting: +1ms Processing: +16ms
    resultTo=null resultCode=0 resultData=null
    nextReceiver=1 receiver=null
    Receiver #0: ResolveInfo{ea4fb2c com.tencent.mm/.plugin.report.service.KVCommCrossProcessReceiver m=0x0}
      priority=0 preferredOrder=0 match=0x0 specificIndex=-1 isDefault=false
      ActivityInfo:
        name=com.tencent.mm.plugin.report.service.KVCommCrossProcessReceiver
        packageName=com.tencent.mm
        enabled=true exported=false processName=com.tencent.mm
        taskAffinity=com.tencent.mm targetActivity=null persistableMode=PERSIST_ROOT_ONLY
        resizeable=false lockTaskLaunchMode=LOCK_TASK_LAUNCH_MODE_DEFAULT
         needGuestControl=false
        ApplicationInfo:
          packageName=com.tencent.mm
          labelRes=0x7f08198f nonLocalizedLabel=null icon=0x7f020388 banner=0x0
          className=com.tencent.mm.app.Application
          processName=com.tencent.mm
          taskAffinity=com.tencent.mm
          uid=10119 flags=0x38983e44 privateFlags=0x10 theme=0x7f0c003c flagsEx=0x0
          requiresSmallestWidthDp=0 compatibleWidthLimitDp=0 largestWidthLimitDp=0
          sourceDir=/data/app/com.tencent.mm-1/base.apk
          seinfo=default
          dataDir=/data/user/0/com.tencent.mm
          sharedLibraryFiles=[/system/framework/com.google.android.maps.jar]
          enabled=true targetSdkVersion=23 versionCode=1080
          supportsRtl=false
          fullBackupContent=true

  mBroadcastsScheduled [foreground]=false
  mBroadcastsScheduled [background]=false
  mHandler:
    Handler (com.android.server.am.ActivityManagerService$MainHandler) {d3f1de2} @ 7689139
      Looper (ActivityManager, tid 19) {dc6c573}
        Message 0: { when=+3m43s775ms callback=com.android.server.AppOpsService$1 target=com.android.server.am.Activit
        Message 1: { when=+11m0s943ms what=27 target=com.android.server.am.ActivityManagerService$MainHandler }
        (Total messages: 2, polling=true, quitting=false)
{ ~ }  »
```

adb shell dumpsys package [packagename]

这个命令可以打印应用在AndroidManifest中静态注册的四大组件的所有相关信息，当然静态注册的Receiver也在其中了。

这里还是以微信作为例子，直接在终端运行adb shell dumpsys package com.tencent.mm，然后在输出的内容中先搜关键字“Receiver Resolver Table”，根据是否设置了Schemes和Data进行分类，内容如下：

```
Receiver Resolver Table:
  Schemes:
      file:
        e7366c8 com.tencent.mm/.booter.MountReceiver filter e7dc3e2
          Action: "android.intent.action.MEDIA_MOUNTED"
          Action: "android.intent.action.MEDIA_EJECT"
          Action: "android.intent.action.MEDIA_UNMOUNTED"
          Action: "android.intent.action.MEDIA_SHARED"
          Action: "android.intent.action.MEDIA_SCANNER_STARTED"
          Action: "android.intent.action.MEDIA_SCANNER_FINISHED"
          Action: "android.intent.action.MEDIA_REMOVED"
          Action: "android.intent.action.MEDIA_BAD_REMOVAL"
          Scheme: "file"
          AutoVerify=false

  Non-Data Actions:
      android.media.ACTION_SCO_AUDIO_STATE_UPDATED:
        e1368f4 com.tencent.mm/.booter.BluetoothReceiver filter 77373
          Action: "android.media.SCO_AUDIO_STATE_CHANGED"
          Action: "android.media.ACTION_SCO_AUDIO_STATE_UPDATED"
          AutoVerify=false
      com.tencent.mm.permission.MM_AUTO_REPLY_MESSAGE:
        119f251 com.tencent.mm/.plugin.auto.service.MMAutoMessageReplyReceiver filter 10fed3a
          Action: "com.tencent.mm.permission.MM_AUTO_REPLY_MESSAGE"
          AutoVerify=false
      android.bluetooth.adapter.action.STATE_CHANGED:
        eb4facb com.tencent.mm/.booter.BluetoothStateReceiver filter 13864ad
          Action: "android.bluetooth.adapter.action.STATE_CHANGED"
          AutoVerify=false
      com.tencent.mm.Intent.ACTION_CLICK_FILEDOWNLOAD_NOTIFICATION:
        7fe7154 com.tencent.mm/.pluginsdk.ui.FileDownloadNotificationClickReceiver filter 43c715c
          Action: "com.tencent.mm.Intent.ACTION_CLICK_FILEDOWNLOAD_NOTIFICATION"
          AutoVerify=false
      android.net.conn.CONNECTIVITY_CHANGE:
        bce5005 com.tencent.mm/.booter.MMReceivers$ConnectionReceiver filter 8a86dc7
          Action: "android.net.conn.CONNECTIVITY_CHANGE"
          AutoVerify=false
      com.android.vending.INSTALL_REFERRER:
        fb89379 com.tencent.mm/.booter.InstallReceiver filter 64b6230
          Action: "com.android.vending.INSTALL_REFERRER"
          AutoVerify=false
      com.tencent.mm.plugin.openapi.Intent.ACTION_HANDLE_APP_REGISTER:
        a3abcb9 com.tencent.mm/.plugin.base.stub.WXEntryActivity$EntryReceiver filter 4383c2e
          Action: "com.tencent.mm.plugin.openapi.Intent.ACTION_HANDLE_APP_REGISTER"
          Action: "com.tencent.mm.plugin.openapi.Intent.ACTION_HANDLE_APP_UNREGISTER"
          AutoVerify=false
      com.google.android.c2dm.intent.RECEIVE:
        55f8b7b com.tencent.mm/.plugin.gcm.modelgcm.GcmBroadcastReceiver filter 2ebddeb
          Action: "com.google.android.c2dm.intent.RECEIVE"
          Action: "com.google.android.c2dm.intent.REGISTRATION"
          Category: "com.tencent.mm"
          AutoVerify=false
      com.tencent.mm.permission.MM_AUTO_HEARD_MESSAGE:
        ec10b99 com.tencent.mm/.plugin.auto.service.MMAutoMessageHeardReceiver filter 16e7c65
          Action: "com.tencent.mm.permission.MM_AUTO_HEARD_MESSAGE"
          AutoVerify=false
      android.media.SCO_AUDIO_STATE_CHANGED:
        e1368f4 com.tencent.mm/.booter.BluetoothReceiver filter 77373
          Action: "android.media.SCO_AUDIO_STATE_CHANGED"
          Action: "android.media.ACTION_SCO_AUDIO_STATE_UPDATED"
          AutoVerify=false
      android.intent.action.BOOT_COMPLETED:
        e8a1535 com.tencent.mm/.booter.MMReceivers$ExdeviceProcessReceiver filter ecf79e1
          Action: "android.intent.action.BOOT_COMPLETED"
          AutoVerify=false
        f3f5dca com.tencent.mm/.booter.MMReceivers$BootReceiver filter a7fa306
          Action: "android.intent.action.BOOT_COMPLETED"
          AutoVerify=false
      MMBakchatServiceStart:
        7a3c3e9 com.tencent.mm/.plugin.backup.bakoldlogic.bakoldmodel.BakOldUSBReceiver filter 7239ea9
          Action: "MMBakchatServiceStart"
          Action: "MMBakchatServiceStop"
          AutoVerify=false
```

```
    com.tencent.mm.plugin.openapi.Intent.ACTION_HANDLE_APP_UNREGISTER:
      a3abcb9 com.tencent.mm/.plugin.base.stub.WXEntryActivity$EntryReceiver filter 4383c2e
        Action: "com.tencent.mm.plugin.openapi.Intent.ACTION_HANDLE_APP_REGISTER"
        Action: "com.tencent.mm.plugin.openapi.Intent.ACTION_HANDLE_APP_UNREGISTER"
        AutoVerify=false
    android.intent.action.DOWNLOAD_COMPLETE:
      a6efa91 com.tencent.mm/.pluginsdk.model.downloader.FileDownloadReceiver filter 7dbd3cf
        Action: "android.intent.action.DOWNLOAD_COMPLETE"
        AutoVerify=false
    MMBakchatServiceStop:
      7a3c3e9 com.tencent.mm/.plugin.backup.bakoldlogic.bakoldmodel.BakOldUSBReceiver filter 7239ea9
        Action: "MMBakchatServiceStart"
        Action: "MMBakchatServiceStop"
        AutoVerify=false
    com.google.android.c2dm.intent.REGISTRATION:
      55f8b7b com.tencent.mm/.plugin.gcm.modelgcm.GcmBroadcastReceiver filter 2ebddeb
        Action: "com.google.android.c2dm.intent.RECEIVE"
        Action: "com.google.android.c2dm.intent.REGISTRATION"
        Category: "com.tencent.mm"
        AutoVerify=false
    com.tencent.mm..plugin.photoedit.action.clear:
      23401a9 com.tencent.mm/.plugin.photoedit.cache.ArtistCacheManager filter acd4748
        Action: "com.tencent.mm..plugin.photoedit.action.clear"
        AutoVerify=false
    com.tencent.mm.Intent.ACTION_CLICK_FLOW_REPORT:
      1383648 com.tencent.mm/.booter.ClickFlowReceiver filter 3272dc4
        Action: "com.tencent.mm.Intent.ACTION_CLICK_FLOW_REPORT"
        AutoVerify=false
```

adb shell dumpsys activity b history

为了方便调试广播，框架会专门记住最近发送过的广播到相关数据中，然后通过dump信息打印出来，记录的信息分为两部分：

在BroadcastQueue.java中mBroadcastHistory 用于存储最近发送过的广播BroadcastRecord，数组最大值是MAX_BROADCAST_HISTORY，在Android N上通常是50条。

另外在BroadcastQueue.java中分别通过mBroadcastSummaryHistory，mSummaryHistoryEnqueueTime，mSummaryHistoryDispatchTime，mSummaryHistoryFinishTime用来存储最近发送的广播的intent，enqueueClockTime，dispatchClockTime，插入时间System.currentTimeMillis()，数组最大值是MAX_BROADCAST_SUMMARY_HISTORY，在Android N上通常是300条。

dumpsy打印的顺序依次是先打印foreground队列的 mBroadcastHistory 和summaryHistory，之后是background队列的mBroadcastHistory 和summaryHistory,而且是按照时间从最近开始排序。

```
adb shell dumpsys activity b history
ACTIVITY MANAGER BROADCAST STATE (dumpsys activity broadcasts)
  Historical broadcasts [foreground]:
  Historical Broadcast foreground #0:
    BroadcastRecord{e788d4d u-1 android.intent.action.TIME_TICK} to user -1
    Intent { act=android.intent.action.TIME_TICK flg=0x50000014 (has extras) }
      extras: Bundle[{android.intent.extra.ALARM_COUNT=1}]
    caller=android null pid=-1 uid=1000
    enqueueClockTime=2017-09-22 17:06:00 dispatchClockTime=2017-09-22 17:06:00
    dispatchTime=-30s130ms (0 since enq) finishTime=-30s16ms (+114ms since disp)
    Total: +114ms Waiting: 0 Processing: +114ms
    resultTo=null resultCode=0 resultData=null
    resultAbort=false ordered=true sticky=false initialSticky=false
    nextReceiver=14 receiver=null
    Deliver #0: BroadcastFilter{1a7dcc6 u0 ReceiverList{a7182a1 1261 system/1000/u0 local:282fb08}}
    Deliver #1: BroadcastFilter{e0830a1 u0 ReceiverList{cafa108 1261 system/1000/u0 local:55c86ab}}
    Deliver #2: BroadcastFilter{56fb47 u0 ReceiverList{fc1f686 1897 com.android.systemui/10037/u0 remote:c20eb61}}
    Deliver #3: BroadcastFilter{aa9b3f u0 ReceiverList{ed0bb5e 1897 com.android.systemui/10037/u0 remote:bbedc99}}
    Deliver #4: BroadcastFilter{e3551f1 u0 ReceiverList{ee23598 1897 com.android.systemui/10037/u0 remote:dc1537b}}
    Deliver #5: BroadcastFilter{f971399 u0 ReceiverList{9a0d4e0 1897 com.android.systemui/10037/u0 remote:23542e3}}
    Deliver #6: BroadcastFilter{a1bcc2 u0 ReceiverList{8bf350d 1897 com.android.systemui/10037/u0 remote:1ff83a4}}
    Deliver #7: BroadcastFilter{3de536c u0 ReceiverList{aceb41f 1897 com.android.systemui/10037/u0 remote:bc42bbe}}
    Deliver #8: BroadcastFilter{c262524 u0 ReceiverList{aa059b7 1897 com.android.systemui/10037/u0 remote:75508b6}}
    Deliver #9: BroadcastFilter{3d6da53 u0 ReceiverList{aa0fa42 1897 com.android.systemui/10037/u0 remote:966f88d}}
    Deliver #10: BroadcastFilter{7c671d5 u0 ReceiverList{12e168c 2570 com.flyme.systemuitools/10037/u0 remote:5850fbf}
    Deliver #11: BroadcastFilter{38320d2 u0 ReceiverList{2165f5d 2405 com.flyme.telecom.usagedata.service/1001/u0 remo
    Deliver #12: BroadcastFilter{a7d40af u0 ReceiverList{ab9d08e 4977 com.meizu.net.pedometer/10097/u0 remote:4d74889}
    Deliver #13: BroadcastFilter{1da82e8 u0 ReceiverList{780440b 1897 com.android.systemui/10037/u0 remote:67b97da}}

    。。。

  Historical Broadcast foreground #49:
    BroadcastRecord{969e10c u-1 android.intent.action.TIME_TICK} to user -1
    。。。
```

```
Historical broadcasts summary [foreground]:
#0: act=android.intent.action.TIME_TICK flg=0x50000014 (has extras)
  0 dispatch +114ms finish
  enq=2017-09-22 17:06:00 disp=2017-09-22 17:06:00 fin=2017-09-22 17:06:00
  extras: Bundle[{android.intent.extra.ALARM_COUNT=1}]

#199: act=android.hardware.usb.action.USB_STATE flg=0x30000010 (has extras)
  0 dispatch 0 finish
  enq=2017-09-22 14:28:59 disp=2017-09-22 14:28:59 fin=2017-09-22 14:28:59
  extras: Bundle[{host_connected=false, connected=true, unlocked=false, adb=true, configured=true, USB_HW_DISCONNECT


Historical broadcasts [background]:
Historical Broadcast background #0:
  BroadcastRecord{b013415 u-1 android.net.wifi.RSSI_CHANGED} to user -1
  Intent { act=android.net.wifi.RSSI_CHANGED flg=0x4000010 (has extras) }
    targetComp: {com.baidu.wenku/com.baidu.wenku.service.PushReceiver}
    extras: Bundle[{newRssi=-62}]
  caller=android 1261:system/1000 pid=1261 uid=1000
  enqueueClockTime=2017-09-22 17:06:21 dispatchClockTime=2017-09-22 17:06:21
  dispatchTime=-8s612ms (+6ms since enq) finishTime=-8s545ms (+67ms since disp)
  Total: +73ms Waiting: +6ms Processing: +67ms
  resultTo=null resultCode=0 resultData=null
  resultAbort=false ordered=false sticky=true initialSticky=false
  nextReceiver=2 receiver=null
  Deliver #0: (manifest)
    priority=0 preferredOrder=0 match=0x108000 specificIndex=-1 isDefault=false
    ActivityInfo:
      name=com.meizu.broadcast.WifiReceiver
      packageName=com.meizu.monitorphone
      enabled=true exported=true directBootAware=false
      resizeMode=RESIZE_MODE_RESIZEABLE
       needGuestControl=false
  Deliver #1: (manifest)
    priority=0 preferredOrder=0 match=0x108000 specificIndex=-1 isDefault=false
    ActivityInfo:
      name=com.meizu.testdev.woody.receiver.WifiReceiver
      packageName=com.meizu.testdev.woody
      enabled=true exported=true directBootAware=false
      resizeMode=RESIZE_MODE_RESIZEABLE
       needGuestControl=false
 。。。。
Historical Broadcast background #49:
  BroadcastRecord{40c965d u-1 android.intent.action.ACCESS_CONTROL_CHANGED} to user -1

Historical broadcasts summary [background]:
#0: act=android.net.wifi.RSSI_CHANGED flg=0x4000010 (has extras)
  +6ms dispatch +67ms finish
  enq=2017-09-22 17:06:21 disp=2017-09-22 17:06:21 fin=2017-09-22 17:06:21
  extras: Bundle[{newRssi=-62}]

#299: act=android.net.wifi.RSSI_CHANGED flg=0x4000010 (has extras)
  +24ms dispatch +1ms finish
  enq=2017-09-22 15:25:34 disp=2017-09-22 15:25:34 fin=2017-09-22 15:25:34
  extras: Bundle[{newRssi=-61}]
```

5.打开广播相关log

adb shell dumpsys activity log br on

不过这个只限于mtk的代码，Android 原生代码并没有提供命令来打开广播相关log。

参考资料：
http://blog.csdn.net/gemmem/article/details/8859493
http://blog.csdn.net/weihan1314/article/details/7973511/