Android DropBox机制

Last edited by **caoquanli** 1 month ago

Android 8.1 DropBox机制

Table of Contents

1. 概述

1.1. 何为DropBox?

- DropBox类似一个Android日志文件的百宝箱，记录了从java层到native层，再到kernel层的各种日志信息。相对于anr、tombstone、logd、kmsg等具有特定场合的日志记录机制，DropBox记录的日志更加强调全面性和持久性，但也因为存储容量的限制会一定程度上进行文件压缩或者抛弃一部分细节内容。

1.2. DropBox文件

- 在/data/system/dropbox目录下的DropBox日志文件有以下几种命名格式的文件："TAG@xxx.txt"，"TAG@xxx.dat"，"TAG@xxx.dat.gz"，"TAG@xxx.txt.gz"，"TAG@xxx.lost"；TAG为日志类型标签，xxx是文件创建的时间戳；这些文件中，前四个分别是文本、二进制数据、压缩二进制数据和压缩文本格式的dropbox日志文件，最后一个以".lost"为后缀的文件实际是DropBoxManagerService为腾出内存空间而清理掉旧的文件时创建的一个空文件，用于标记这是个被清理的文件。

- TAG标签基本标明了日志文件的类型，目前查到的有二十多种，主要分为两类，一类系统开机后主动检测相关日志目录而生成的，一类是系统运行过程中发生异常时生成的。

- 以"SYSTEM_"为前缀的，除SYSTEM_TOMBSTONE之外都是在系统启动之后主动进行相关体重目录的检查而生成的日志记录文件，包括SYSTEM_LAST_KMSG、SYSTEM_AUDIT、SYSTEM_BOOT、SYSTEM_RESTART、SYSTEM_RECOVERY_LOG、SYSTEM_RECOVERY_KMSG、SYSTEM_FSCK这些TAG。SYSTEM_TOMBSTONE不仅在系统启动之后就主动检测一次/data/tombstones目录，在后期系统运行过程中出现新的墓碑文件时也会同步生成一个对应的dropbox日志文件。

- 以"system_server_"、"system_app_"和"data_app_"为前缀的，分别代表了system_service进程、系统app、第三方app，包括了wtf、crash、native_crash、strictmode、lowmem、anr、watchdog 这些应用场合，都是在系统运行过程中发生一些异常而调用了ActivityManagerService(AMS)中的接口进行文件打印的。其他的，包括BATTERY_DISCHARGE_INFO、netstats_error、storage_benchmark、storage_trim、netstats_error、netstats_dump等应用场合则是直接调用的DropBoxManager的接口进行文件打印。

1.3. DropBoxManager(DBM)

- DBM是dropbox文件打印和信息获取的接口管理类，这里介绍addText()、addData()、addFile()三个打印接口和一个isTagEnabled()标签的判断接口。

- 其中addText()用于打印字符串到".txt"或者".txt.gz"中；addData()用于打印字节数组，根据输入参数flags类型来判断打印到文本格式还是二进制数据文件中；addFile()直接将其他日志文件打印为dropbox文件，并根据输出参数flags类来决定dropbox文件的格式。

- isTagEnabled()按注释的意思是判断settings设置的列表来判断将要输入的TAG是否合法，不过没有找到注释说的那个列表，c测试随便编写了一个TAG也能正常生成文件。

- 调用DBM接口前，需要通过系统服务来获取DBM实例，示例：

```
        if (ServiceManager.getService(Context.DROPBOX_SERVICE) == null) return;
       final DropBoxManager dbox = mContext.getSystemService(DropBoxManager.class);
       ...
        dbox.addText(dropboxTag, sb.toString());
```

1.4. DropBoxManagerService(DBMS)

- DBMS继承于SystemService，在系统服务(/frameworks/base/services/java/com/android/server/SystemServer.java)中启动。

- DBMS是dropbox文件的管理服务，dropbox文件的创建、压缩、写入和清理都在这个服务里完成。

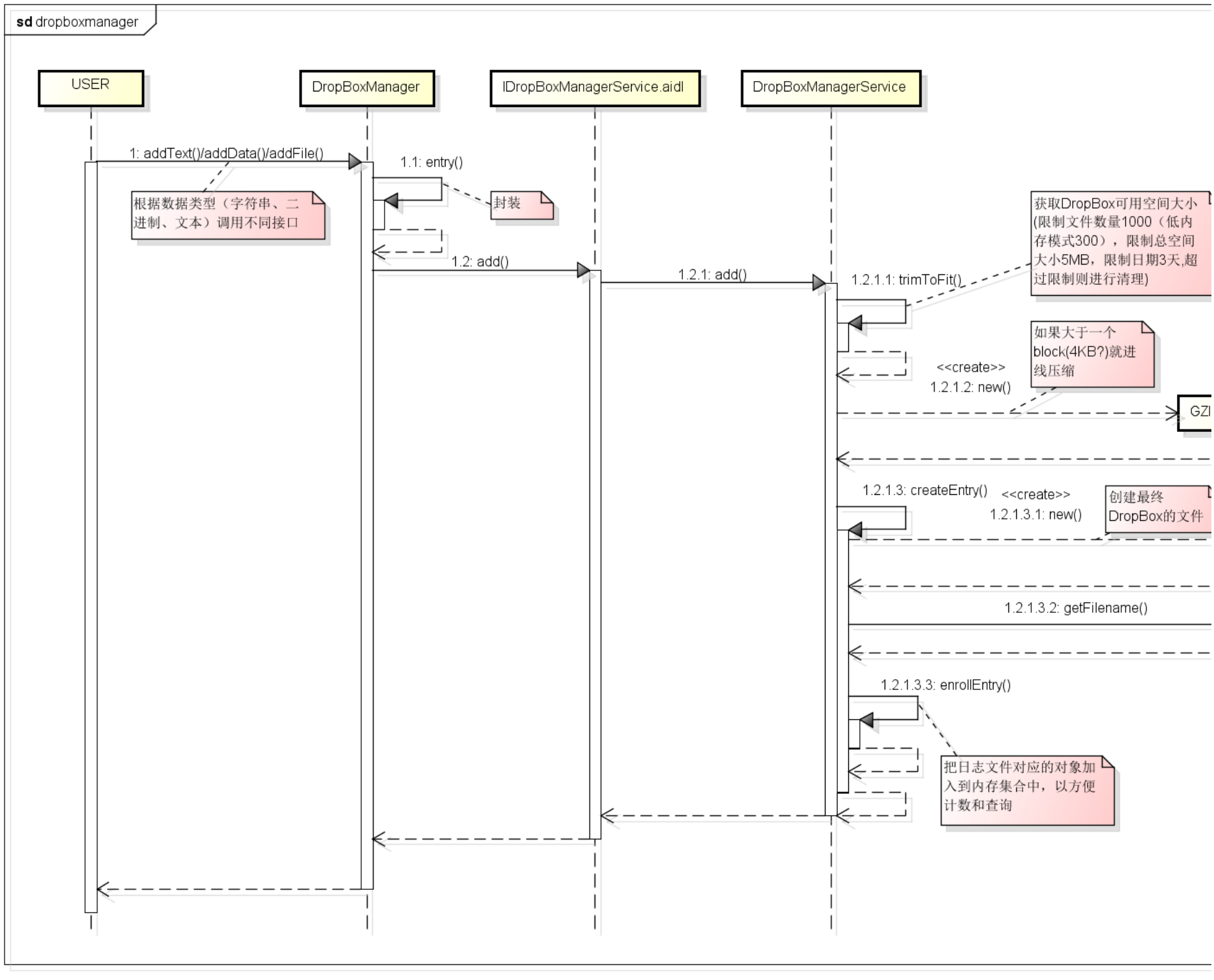
- 当其他进程调用了DBM的接口来打印dropbox文件，会用DBM封装成entry类，然后通过IDropBoxManagerService.aidl调用DBMS的add()接口请求打印文件。

- 在DBMS的add()方法中，会依次做如下工作：判断一下TAG的合法性；调用trimToFit()获取系统分配的剩余可用空间，默认在/data/system/dropbox目录下创建的dropbox文件数量不超过1000（低内存模式不超过300），文件创建时间不超过3天，总大小不超过5MB；判断将要打印的内容是否超过一个block的大小（4KB?），超过则进行压缩处理；创建临时文件，读取数据流；调用createEntry()方法生成最终文件；发送写dropbox结束广播，避免死锁

- createEntry()方法里调用getFile()获取文件名称，调用EntryFile()将前面创建的临时文件命名为获取的文件名称，之后调用enrollEntry()把日志文件对应的对象加入到内存集合中，以方便计数和查询。

- ".lost"后缀空文件在trimToFit()方法中清理旧文件时创建，用于标记被清理的文件。

- 从DBM接口的调用，到dropbox文件的创建写入流程可以用如下时序图表示：



- 参考资料： Android DropBoxManager服务分析

2. 应用场景

2.1. wtf

2.1.1. 背景

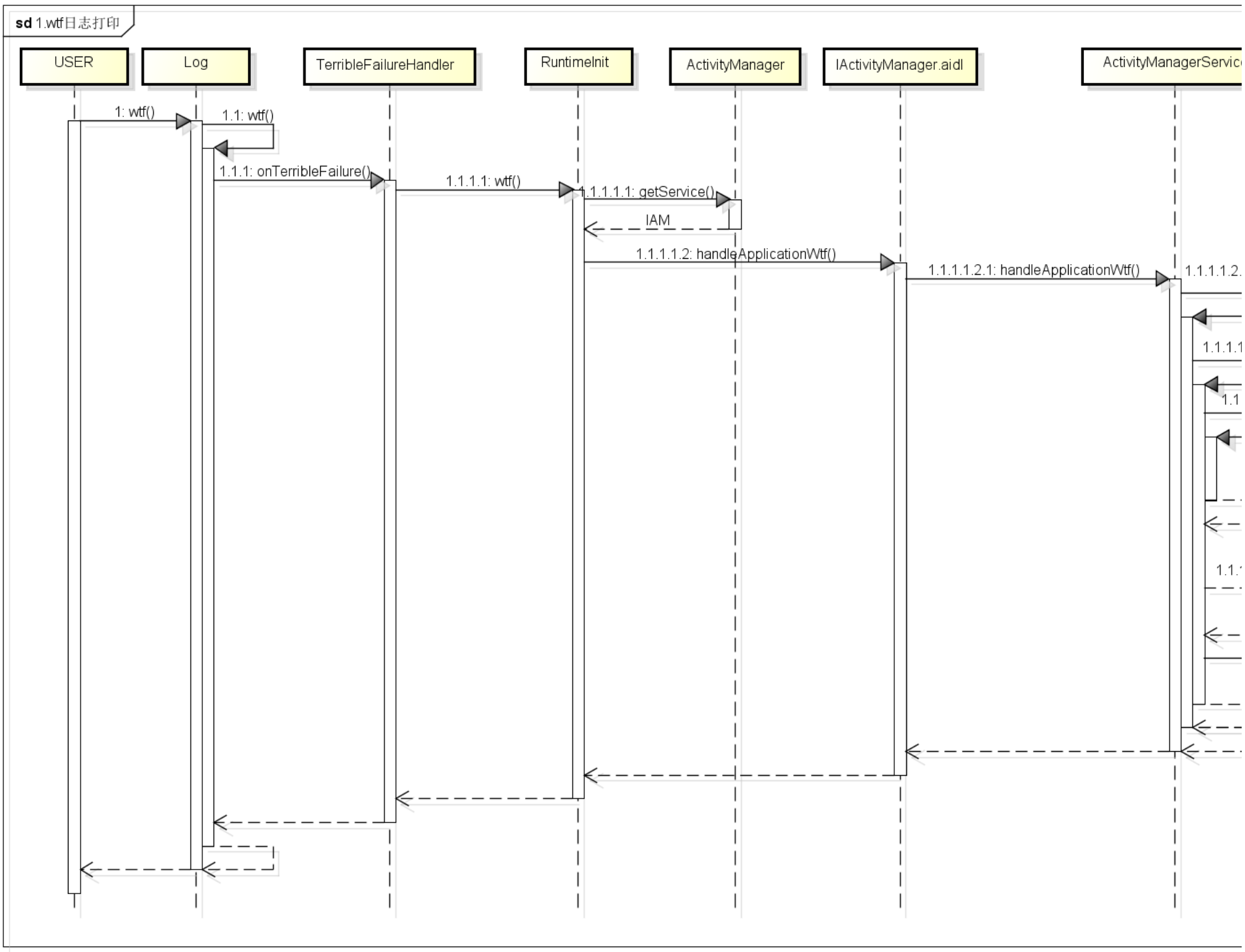
- WTF全称"What a Terrible Failure"，是在程序运行中发生了开发时所预见的不该发生的错误时，应用程序主动进行报告的一种日志。

2.1.2. 机制

- 在JAVA层打印log的接口类android.util.Log中提供了一个静态的wtf函数，当错误case发生时，应用程序会调用该接口函数链接到AMS，之后通过AMS的addErrorToDropBox()方法调用DBM的addText()接口创建wtf记录文件。
- 完整TAG包括"system_server_wtf"、"system_app_wtf"和"data_app_wtf"，分别表示不同类别的应用。

2.1.3. 时序

- 调用wtf接口打印dropbox日志文件时序如下：



2.2. app_crash

2.2.1. 背景

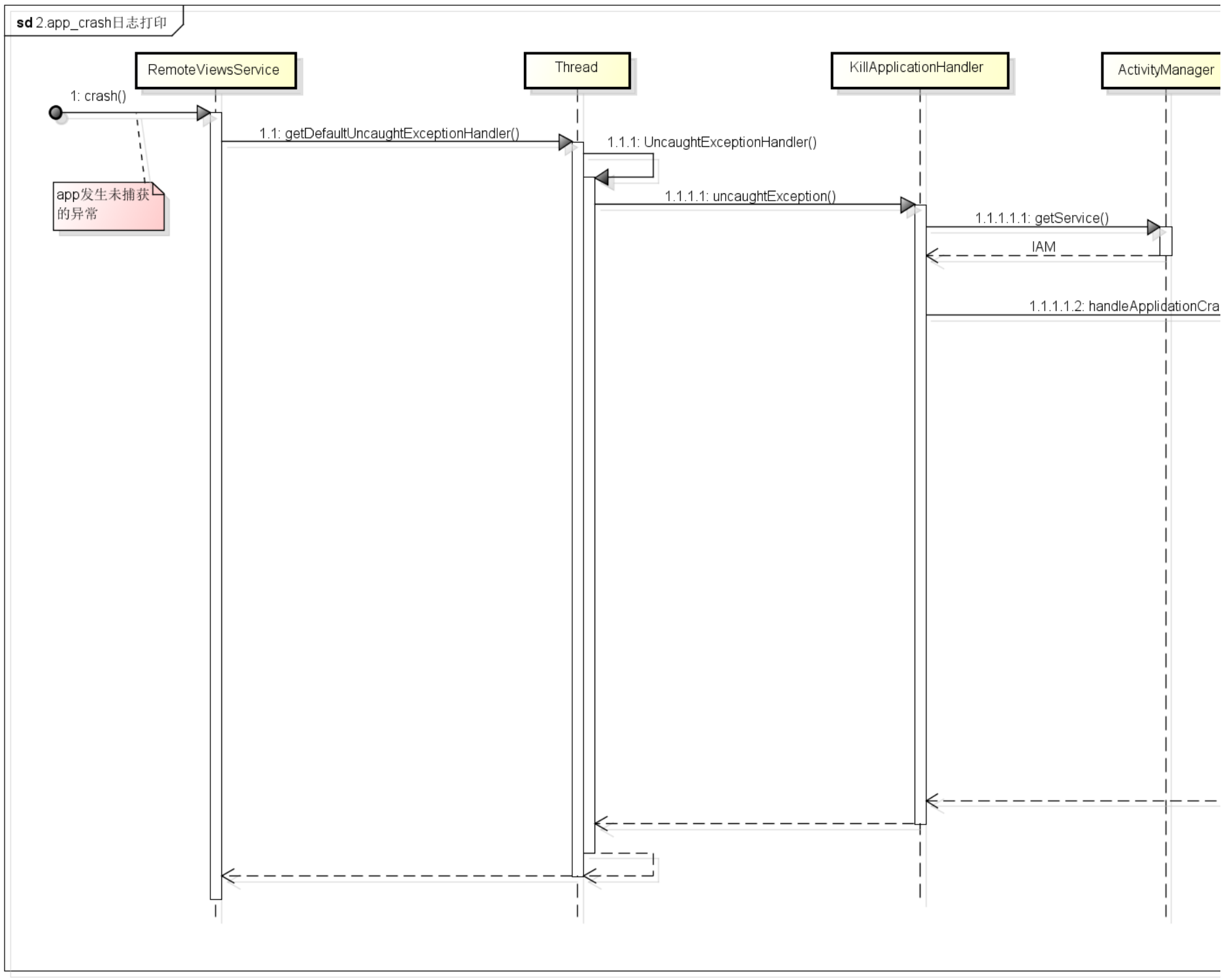
- 当JAVA层应用发生未捕获（catch）到的crash时，由AMS记录本次异常到DropBox中，之后弹窗提示强制退出。

2.2.2. 机制

- java层的应用进程都由zygote进程fork()而来，而zygote进程在开机时在RuntimeInit中设置了一个默认的异常捕获器，应用也就都继承了这个捕获器。
- 当应用运行过程中出现crash，而其本身又没有设置捕获器去捕捉这个异常时，就会响应前面那个默认的捕获器。默认的捕获器中会调用AMS的方法去记录这次crash信息。
- 注意如果生成的是"system_server_crash"开头的dropbox文件，表示发生crash的是system_server进程。
- 参考资料： Android crash

2.2.3. 时序

- 打印java层应用crash日志文件时序如下：



2.3. app_native_crash

2.3.1. 背景

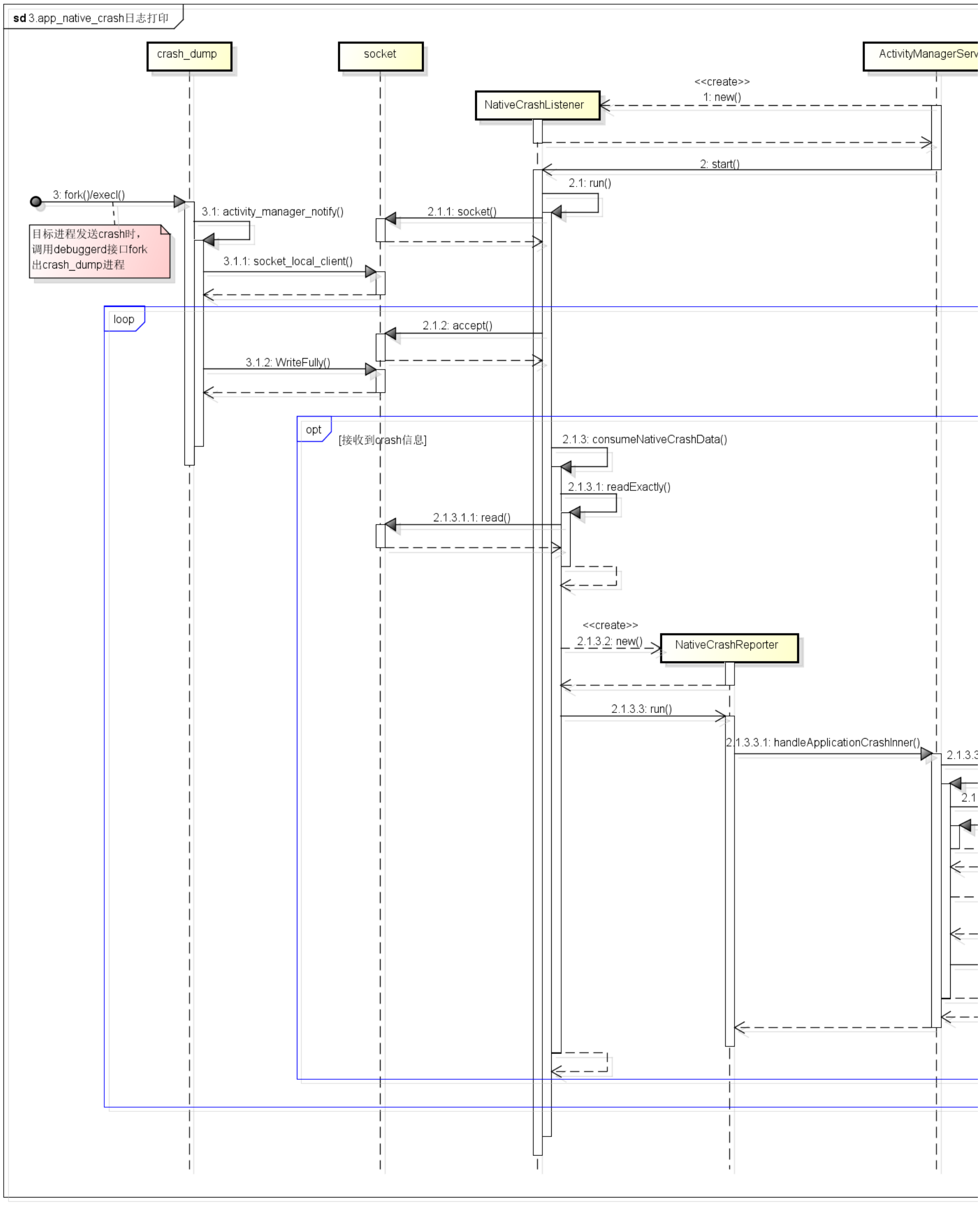
- native层进程发生crash，并通过native层debuggerd模块产生墓碑文件时，会被AMS设置的NativeCrashListener线程监听到，进而调用DBM接口生成dropbox文件记录应用相关的堆栈。

2.3.2. 机制

- AMS中单独启动了一个NativeCrashListener线程，该线程会通过socket节点"/data/system/ndebugsocket"循环监听debuggerd模块中crash_dump进程的通知。
- 当app应用调用的native层程序发生crash时，会被内核监测并发送相关信号给目标进程，进而启动crash_dump进程dump墓碑文件，之后通过套接字通知NativeCrashListener线程。NativeCrashListener线程收到通知后，通过AMS调用DBM接口生成dropbox文件记录app应用相关的堆栈。
- *使用kill -6 <pid> 测试可以看到，只有app应用进程才会产生此类DropBox文件，普通的native层进程不会，只是监听app应用进程调用到native层程序发生的crash？暂时还没从代码里没有找到过滤的地方，后续继续调查。
- debuggerd模块中crash_dump进程启动的时机参考 Android tombstone产生机制与分析技巧 一文第三节

2.3.3. 时序

- 打印app应用native层的crash日志文件时序如下：



2.4. strictmode

2.4.1. 背景

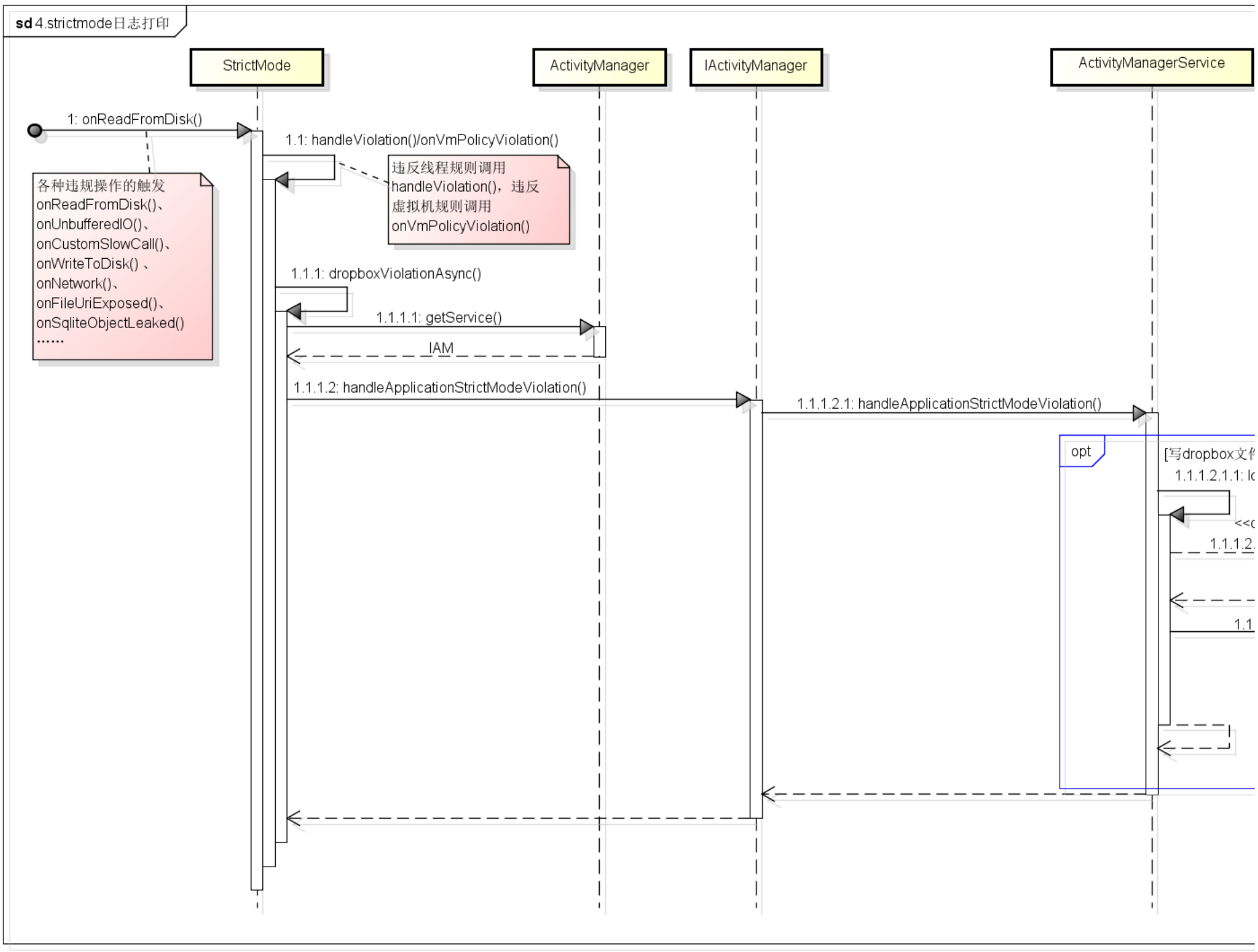
- 一些在主线程中进行网络、文档等耗时操作的不规范行为，会严重影响应用的响应能力，甚至导致ANR(应用程序不响应)的出现。因此，Android设置了strictmode严格模式来检测记录这些不规范行为。

2.4.2. 机制

- strictmode的检测分为ThreadPolicy线程策略检测和VmPolicy虚拟机策略检测。在应用的onCreate()方法中开启相关的检测并且开启写违规记录到dropbox日志文件的"惩罚"。当进程运行中出现的违规行为便会触发相应的函数，然后调用AMS去打印堆栈到dropbox文件。
- ThreadPolicy线程策略检测示例：自定义的耗时调用 使用detectCustomSlowCalls()开启，违规时触发onCustomSlowCall()；磁盘读取操作 使用detectDiskReads()开启，违规时触发onReadFromDisk()；磁盘写入操作 使用detectDiskWrites()开启，违规时触发onWriteToDisk()；网络操作 使用detectNetwork()开启，违规时触发onNetwork()。
- VmPolicy虚拟机策略检测示例：网络加密访问使用detectCleartextNetwork()开启，违规时触发onCleartextNetworkDetected()；文件共享使用detectFileUriExposure开启，违规时触发onFileUriExposed()；数据库泄露 使用detectLeakedSqlLiteObjects()开启，违规时触发onSqliteObjectLeaked()。
- 参考资料： StrictMode机制以及使用场景

2.4.3. 时序

- strictmode模式打印违规信息到dropbox日志文件时序如下：



2.5. lowmem

2.5.1. 背景

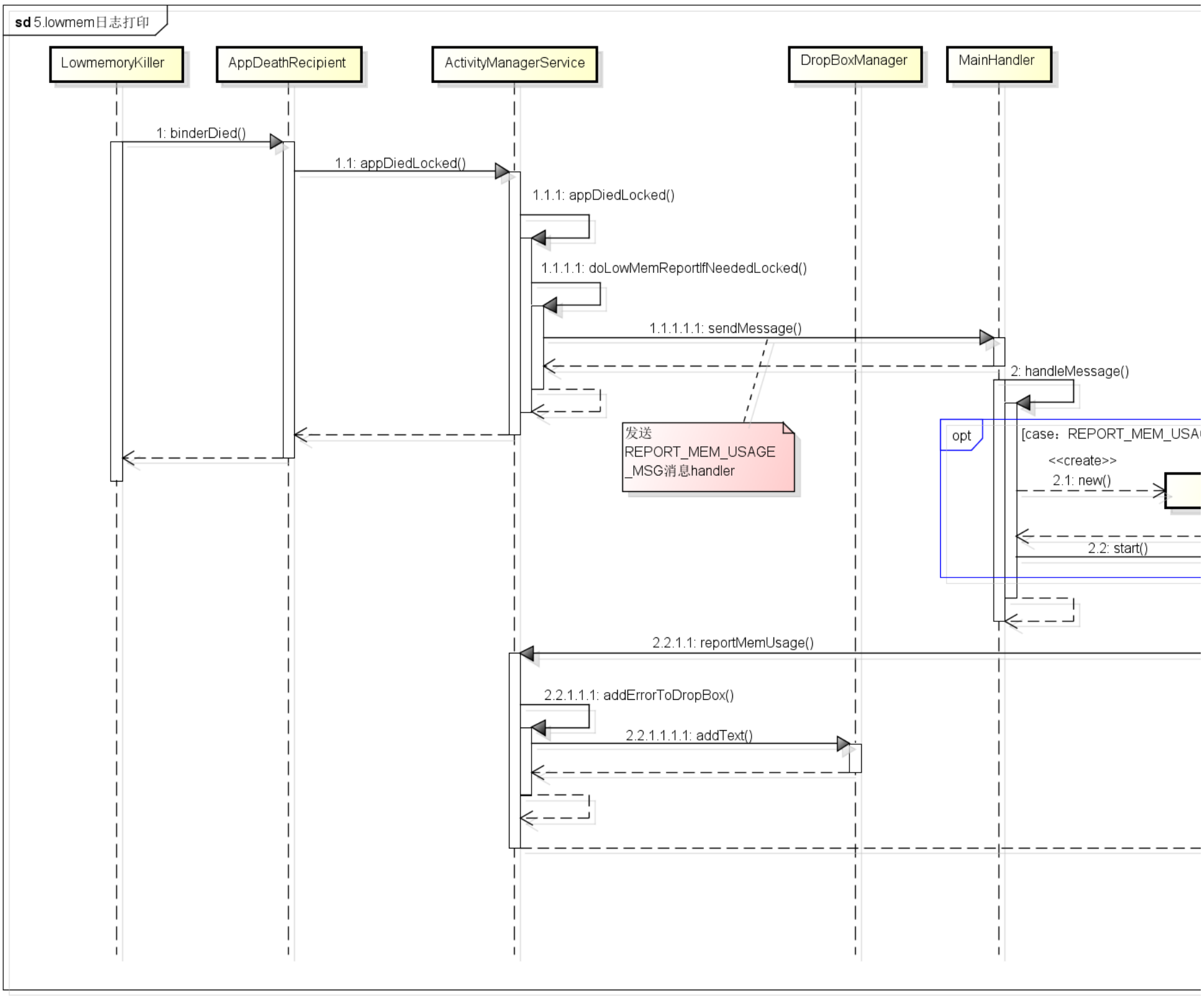
- Android系统运行过程中，启动过多的应用会导致可用内存变少，这个时候需要主动关闭后台应用来释放内存，当没有可关闭的后台应用时，内存则得不到有效释放，这个异常情况将会被记录到dropbox文件中。

2.5.2. 机制

- Android系统运行过程中，当检测到内存不足时，会触发进程回收机制LowmemoryKiller。
- LowmemoryKiller杀死优先级较低的进程后，然后通过binderDied调用appDiedLocked()方法来告知AMS来告知进程已经死亡。appDiedLocked()方法里判定进程由于内存过低而死亡后，会调用doLowMemReportIfNeededLocked()方法进一步判断内存情况。
- doLowMemReportIfNeededLocked()方法判断后台进程都死亡之后内存是否得到释放，当内存依然不足时，则会发送“REPORT_MEM_USAGE_MSG”消息报告内存不足情况。
- 消息的处理在MainHandler的handleMessage()方法里，在收到REPORT_MEM_USAGE_MSG消息之后调用reportMemUsage收集内存的具体使用情况，并通过AMS的addErrorToDropBox接口记录信息到dropbox文件中。
- 除此之外，在ActivityManagerShellCommand.java中处理shell命令“am kill-all”时会调用到AMS中的killAllBackgroundProcesses()，该方法会杀死所有的后台进程，然后调用doLowMemReportIfNeededLocked来判断内存是否有效释放。
- 参考资料： Android Low Memory Killer

2.5.3. 时序

- 使用dropbox日志记录内存不足情况的时序如下：



2.6. anr

2.6.1. 背景

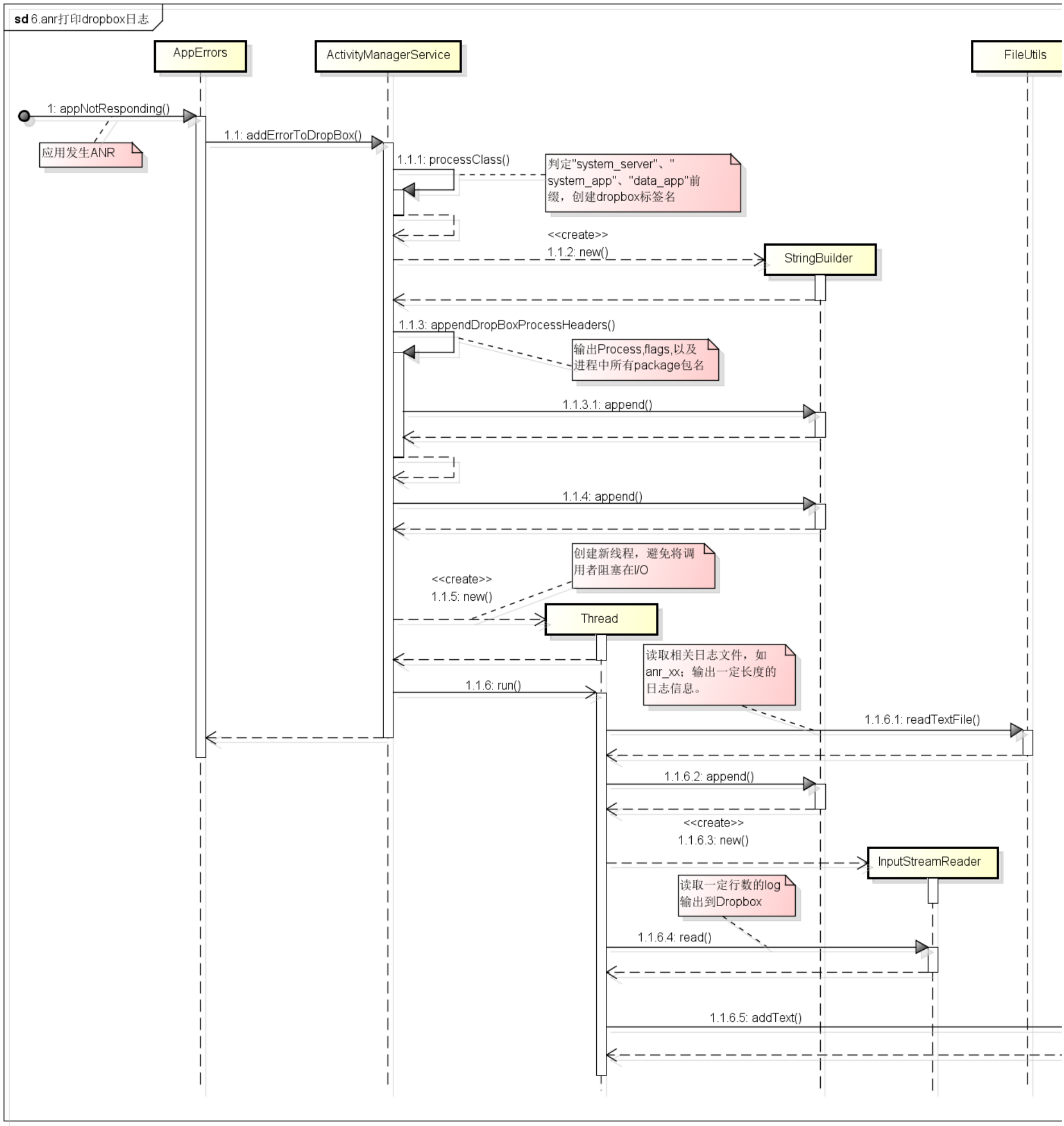
- android系统要求应用主线程不做耗时操作，因此会通过AMS设置一个时间阈值来监测相关应用执行过程是否超时，当应用不响应或者响应时间过程时，会产生ANR现象。
- ANR发生后，会通过log、anr文件、dropbox文件来记录ANR信息。

2.6.2. 机制

- 当应用发生ANR时，会通过AMS最终调用到AppErrors中的appNotResponding()方法来处理ANR信息的dump流程。
- 在appNotResponding()中完成anr文件的dump之后，会将anr文件直接传递到AMS的addErrorToDropBox()方法中进行处理。
- 在addErrorToDropBox()方法中判断anr文件大小，当文件大小超过了单个dropbox文件可用大小（默认单个dropbox文件信息量最多为192KB，包含了一些文件头信息，可用小于192KB），会抛弃anr文件的后面的内容。之后调用DBM的接口addText()方法打印dropbox文件。

2.6.3. 时序

- 生成anr的dropbox日志文件时序如下：



2.7. watchdog

2.7.1. 背景

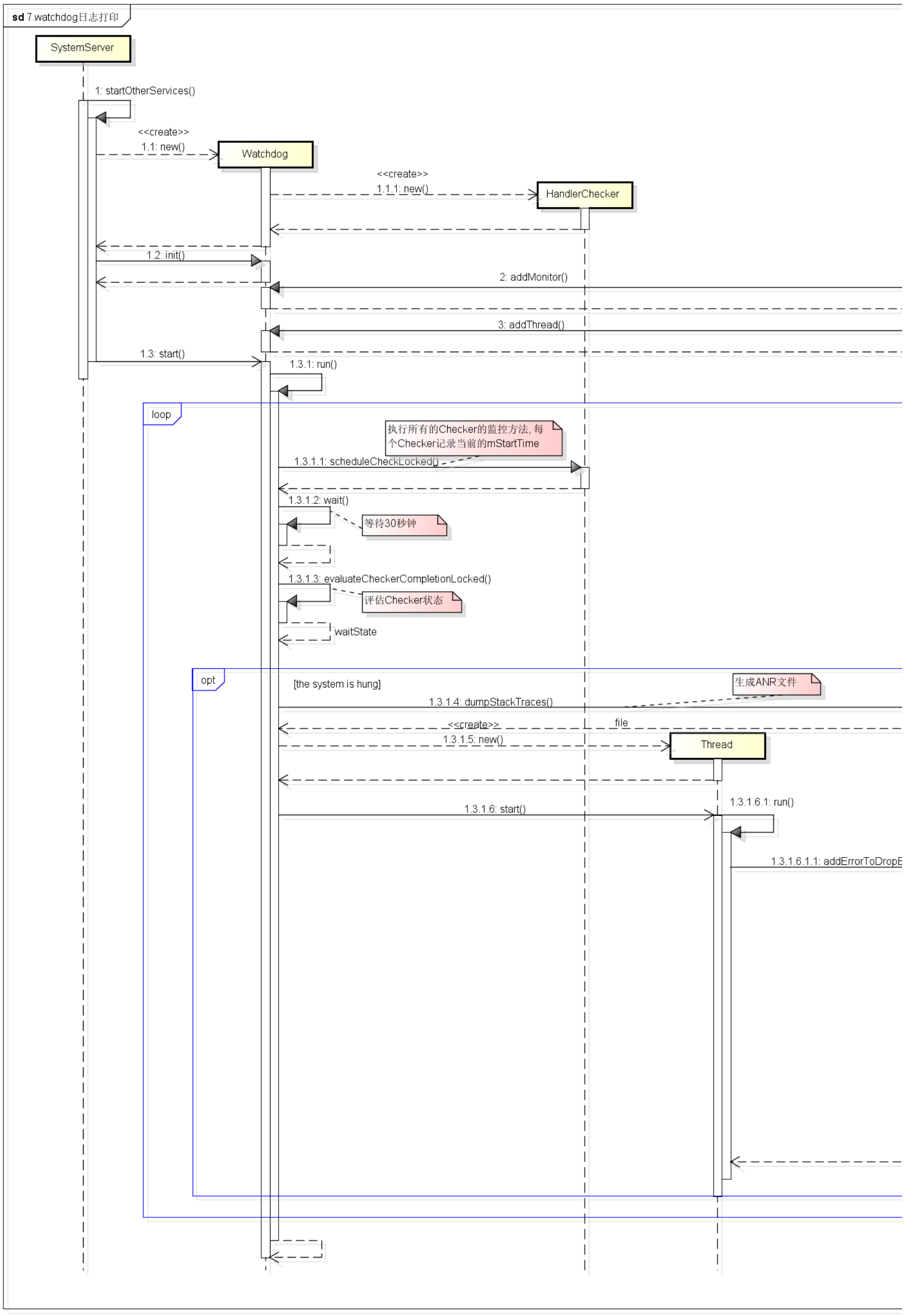
- WatchDog是系统进程的看门狗，用于检测包括AMS、WMS(WindowManagerService)、DMS(DisplayManagerService)等核心服务的主线程运行过程中出现的问题。

2.7.2. 机制

- 在系统服务SystemServer进程启动后，会在其startOtherService()方法中启动Watchdog线程，然后调用Watchdog线程的init()方法去获取AMS对象。
- Watchdog线程主要通过addMonitor()和addThread()方法添加目标服务对象和Handler对象，用来监测死锁和阻塞情况。以AMS为例，AMS在构造的时候调用了这两个方法。
- Watchdog运行之后，开始循环调用HandlerChecker的scheduleCheckLocked()方法，计时30s后，然后通过evaluateCheckerCompletionLocked()方法来判定HandlerChecker完成状态。如果状态为OVERDUE超时状态，则进入超时处理流程。
- 超时处理流程中会dump系统进程的堆栈信息，并创建"watchdogWriteToDropbox"线程，在线程中调用AMS的addErrorToDropBox()方法生成dropbox文件。
- 参考资料： android进阶之Watchdog检查系统异常机制

2.7.3. 时序

- 以AMS为例，WatchDog线程生成dropbox日志文件时序如下：



2.8. SYSTEM_BOOT

2.8.1. 背景

- 每次开机通过dropbox日志文件记录系统硬件、内核版本等信息，示例：

```
 isPrevious: true
Build: mediatek/Leepi_14s/mt2712:8.1.0/OPM1.171019.026/liangc01161529:userdebug/test-keys
Hardware: mt2712
```

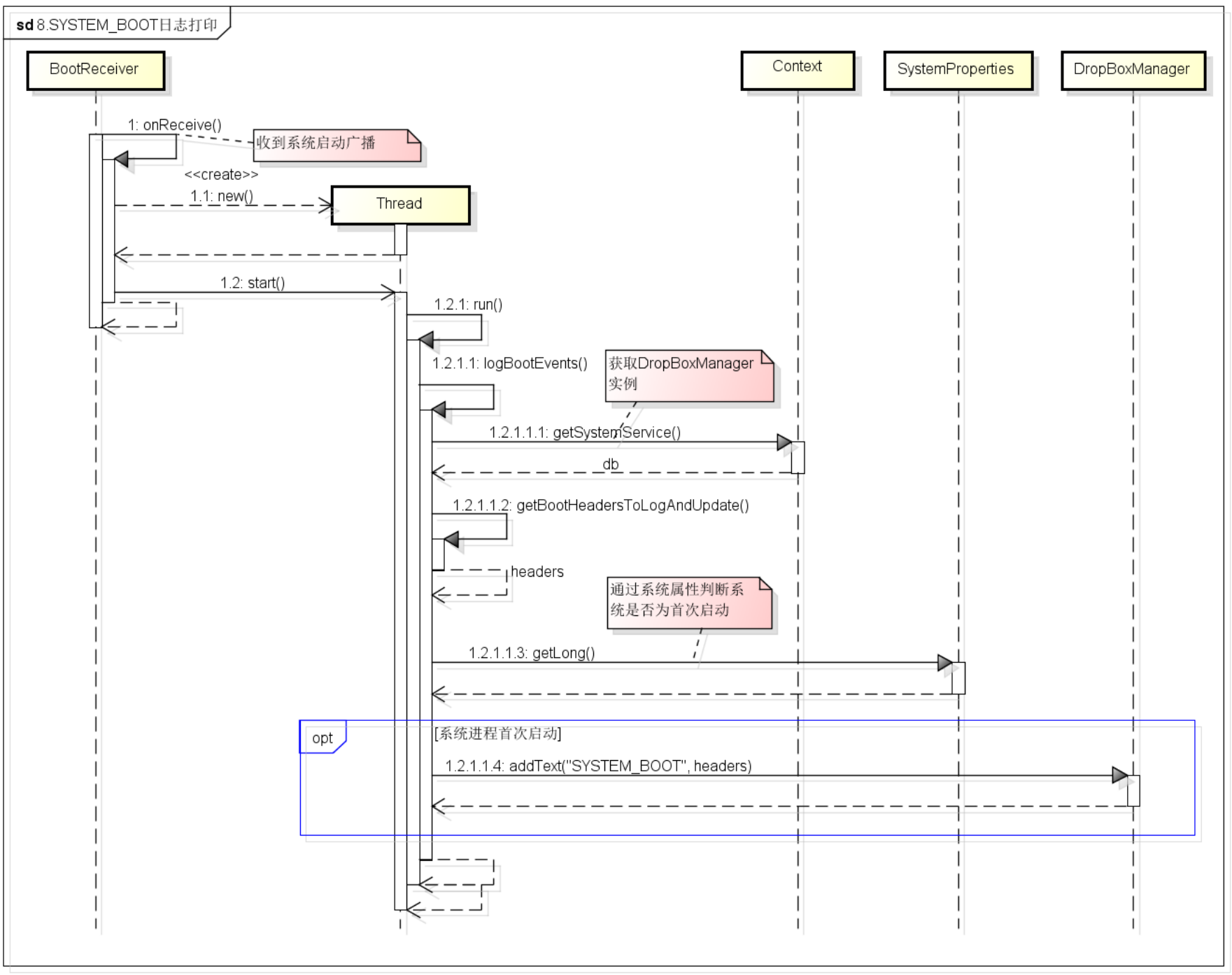
```
Revision: 0
Bootloader: unknown
Radio: unknown
Kernel: Linux version 4.9.90+ (liangchengbin@liangchengbin-HP-PC) (gcc version 4.9.x 20150123 (prerelease) (GCC) ) #1 SMP
```

2.8.2. 机制

- 在BootReceiver类中，有一个继承于广播接收器BroadcastReceiver的方法onReceive()。当系统启动完成，发出开机广播时，onReceive()方法会响应。
- onReceive()方法中未避免主线程阻塞，创建了一个线程并利用logBootEvents()方法记录开机事件。
- logBootEvents()里面判断不是系统进程（system_service）重启之后调用DBM的接口addText()来打印"SYSTEM_BOOT"标签的dropbox日志文件。

2.8.3. 时序

- 生成"SYSTEM_BOOT"标签的dropbox日志文件时序如下：



2.9. SYSTEM_LAST_KMSG

2.9.1. 背景

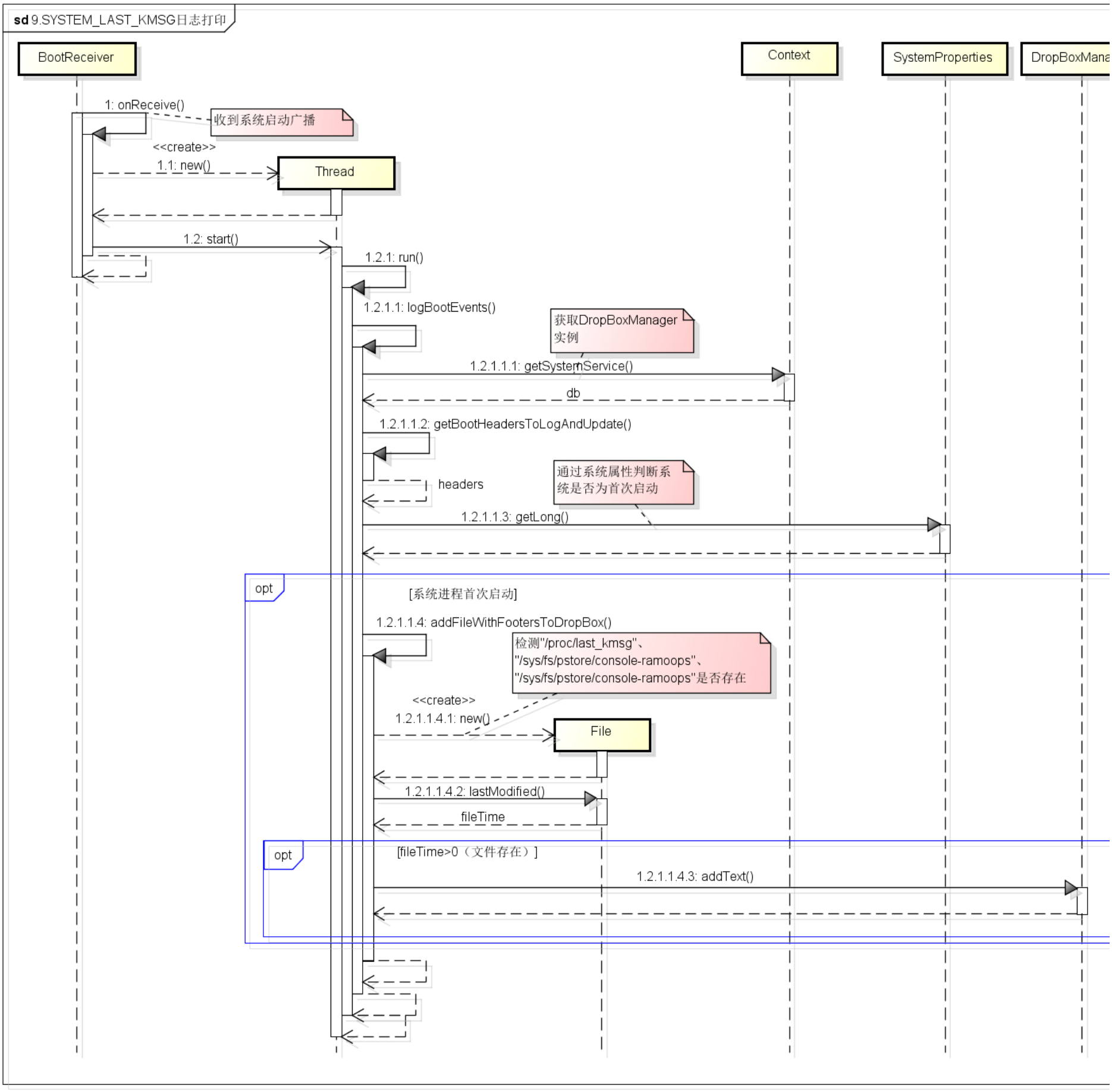
- 当系统异常重启前的一小段时间，会在/proc/last_kmsg节点记录一些内核log，通过这些log可以分析系统重启原因，可能是kernel panic造成，也有可能其他人为重启。
- 参考资料： Android的ram_console和last_kmsg

2.9.2. 机制

- 在系统启动完成发出启动完成广播后，BootReceiver类中的onReceive()开始响应，并创建一个独立线程调用logBootEvents()方法记录开机事件。
- logBootEvents()方法里判断不是系统进程（system_service）重启之后会依次检查"/proc/last_kmsg"、"/sys/fs/pstore/console-ramoops"、"/sys/fs/pstore/console-ramoops-0"节点文件是否存在。
- 如果存在，表示系统是重启，之后会调用addFileWithFootersToDropBox()方法从节点读取信息，并调用DBM的接口addText()来打印"SYSTEM_LAST_KMSG"标签的dropbox日志文件。

2.9.3. 时序

- 生成"SYSTEM_LAST_KMSG"标签的dropbox日志文件时序如下：



powered

2.10. SYSTEM_AUDIT

2.10.1. 背景

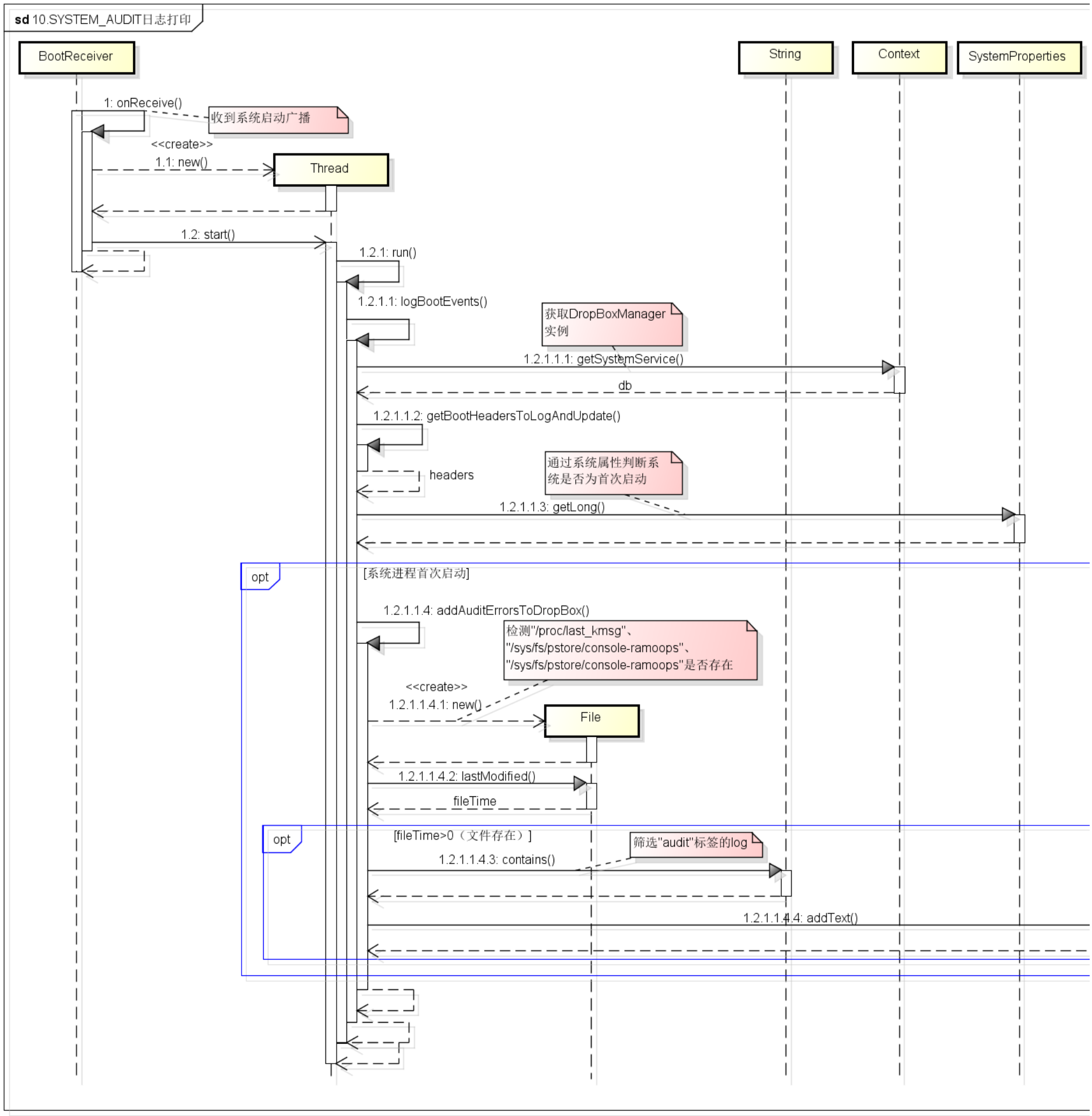
- 在内核log中，有一类标签为"audit"的审计log，用于记录进程访问系统资源的操作。当进程没有权限时，会用“denied”标明。
- "SYSTEM_AUDIT"标签的dropbox日志就是过滤系统重启后的last_kmsg，单独提取"audit"这类log。

2.10.2. 机制

- 在系统启动完成发出启动完成广播后，BootReceiver类中的onReceive()开始响应，并创建一个独立线程调用logBootEvents()方法记录开机事件。
- logBootEvents()方法里判断不是系统进程（system_service）重启之后会调用addAuditErrorsToDropBox()方法依次检查"/proc/last_kmsg"、"/sys/fs/pstore/console-ramoops"、 "/sys/fs/pstore/console-ramoops-0"节点文件是否存在，如果存在则过滤出"audit"标签log，并调用DBM的接口addText()来打印"SYSTEM_AUDIT"标签的dropbox日志文件。

2.10.3. 时序

- 生成"SYSTEM_AUDIT"标签的dropbox日志文件时序如下：



2.11. SYSTEM_RESTART

2.11.1. 背景

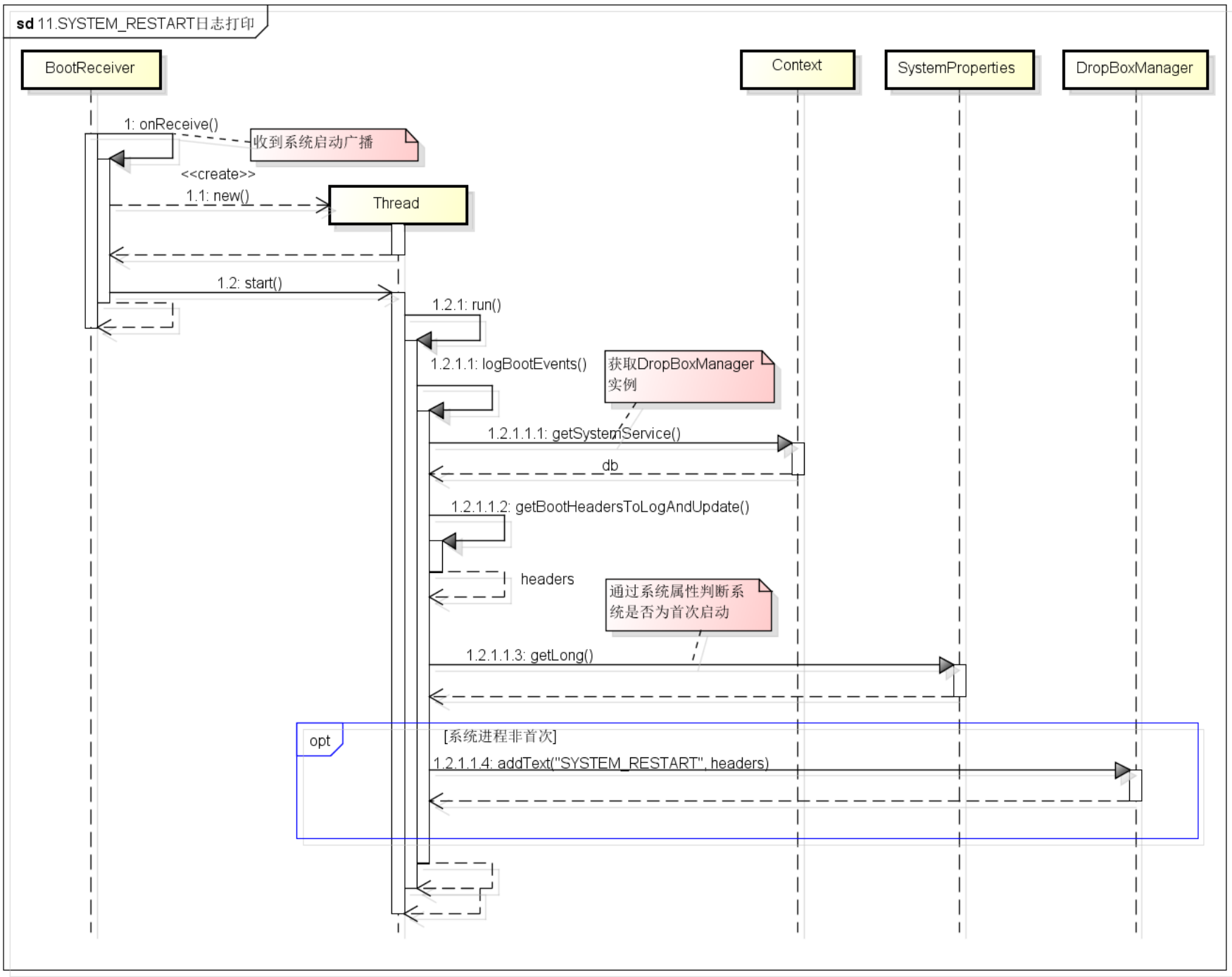
- 系统进程system server开机后正常只会启动一次，当异常发生导致system server重启时，会通过dropbox添加一条重启记录。

2.11.2. 机制

- 当系统进程system server重新启动后，会发送系统启动广播。BootReceiver类中的onReceive()方法响应广播，并创建一个独立线程调用logBootEvents()方法记录开机事件。
- logBootEvents()方法中通过"ro.runtime.firstboot"属性值判断为system server进程重启，直接调用DBM的接口addText()来打印"SYSTEM_RESTART"标签的dropbox日志文件。日志文件的内容格式与"SYSTEM_BOOT"标签的dropbox文件一致。

2.11.3. 时序

- 生成"SYSTEM_RESTART"标签的dropbox日志文件时序如下：



powered by Astah

2.12. SYSTEM_RECOVERY_LOG

2.12.1. 背景

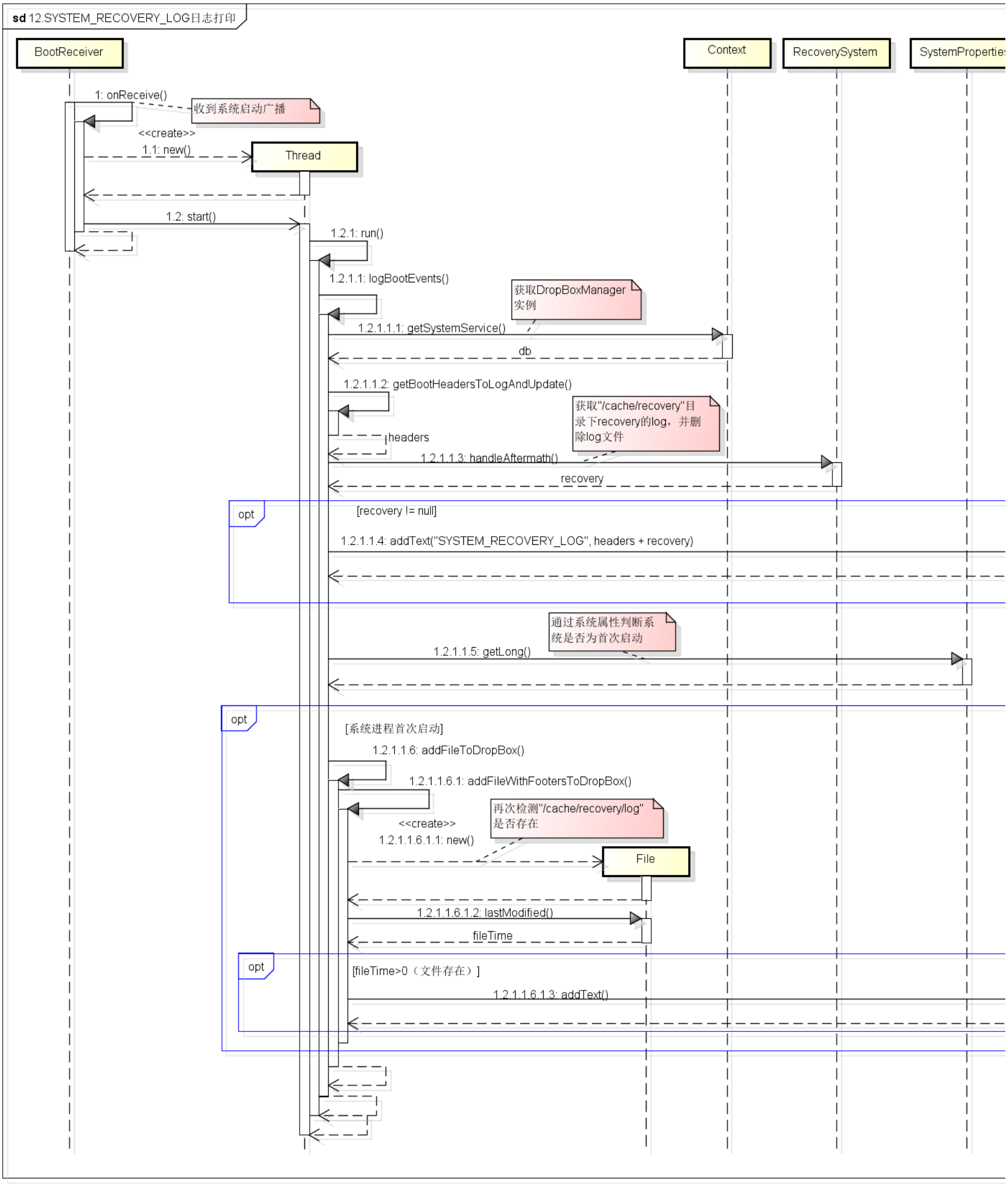
- 系统从recovery模式中启动，会在"/cache/recovery/log"和"/cache/recovery/last_kmsg"中记录recovery流程的日志信息。
- 参考资料： Android 8.0 recovery 流程分析

2.12.2. 机制

- 在系统启动完成发出启动完成广播后，BootReceiver类中的onReceive()开始响应，并创建一个独立线程调用logBootEvents()方法记录开机事件。
- logBootEvents()方法里判断不是系统进程（system_service）重启之后会检查 "/cache/recovery/log"节点文件是否存在。
- 如果存在说明系统是从recovery模式启动，之后通过addFileToDropBox()方法调用addFileWithFootersToDropBox()方法，读取节点信息后再调用DBM的接口addText()来打印"SYSTEM_RECOVERY_LOG"标签的dropbox日志文件。

2.12.3. 时序

- 生成"SYSTEM_RECOVERY_LOG"标签的dropbox日志文件时序如下：



2.13. SYSTEM_RECOVERY_KMSG

2.13.1. 背景

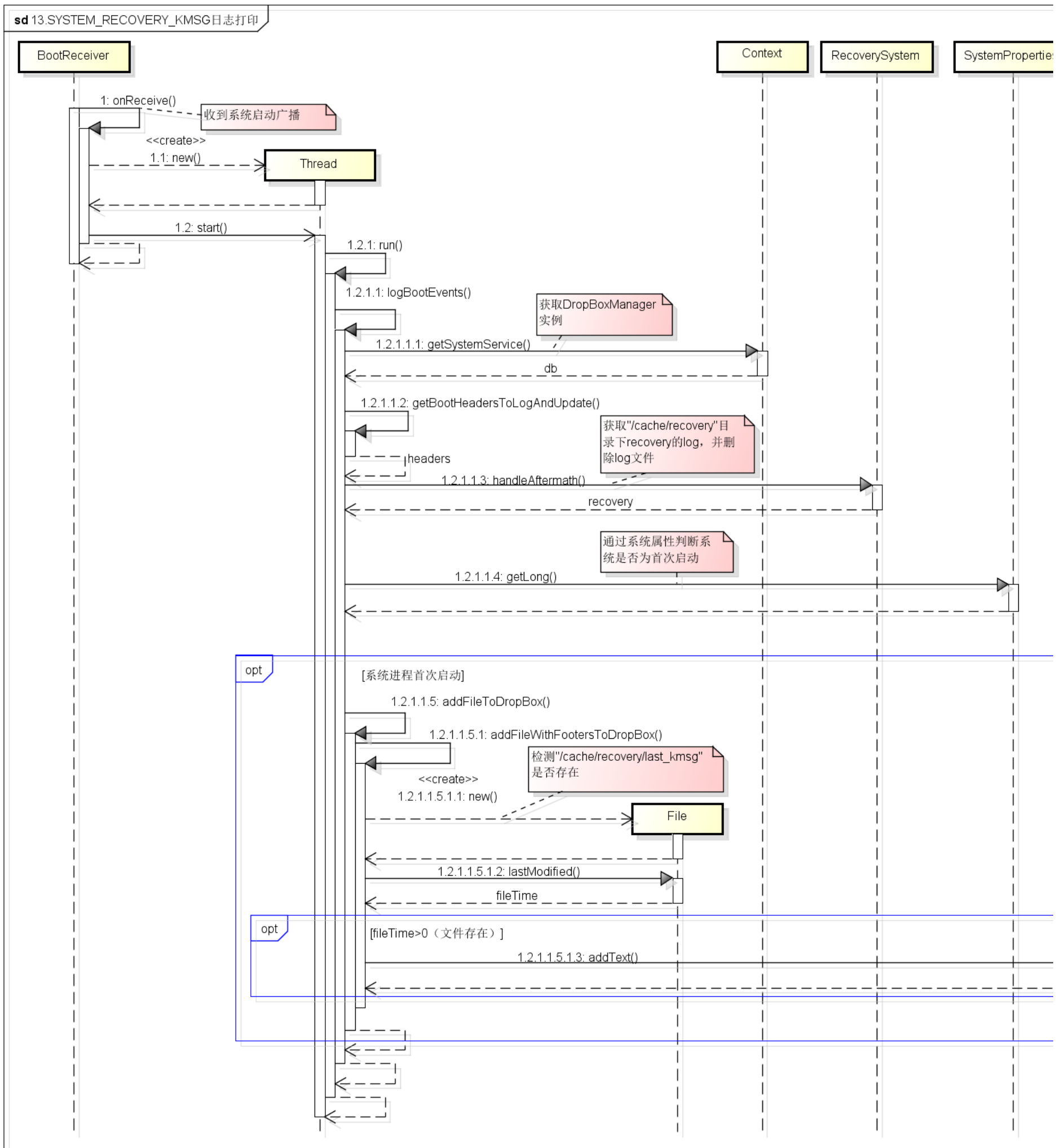
- 系统从recovery模式中启动，会在"/cache/recovery/log"和"/cache/recovery/last_kmsg"中记录recovery流程的日志信息。

2.13.2. 机制

- 在系统启动完成发出启动完成广播后，BootReceiver类中的onReceive()开始响应，并创建一个独立线程调用logBootEvents()方法。
- logBootEvents()方法里判断不是系统进程（system_service）重启之后会检查 "/cache/recovery/last_kmsg"节点文件是否存在。
- 如果存在说明系统是从recovery模式启动，之后通过addFileToDropBox()方法调用addFileWithFootersToDropBox()方法，读取节点信息后再调用DBM的接口addText()来打印"SYSTEM_RECOVERY_KMSG"标签的dropbox日志文件。

2.13.3. 时序

- 生成"SYSTEM_RECOVERY_KMSG"标签的dropbox日志文件时序如下：



2.14. SYSTEM_TOMBSTONE

2.14.1. 背景

- 系统native层进程发生crash时，会通过debuggerd模块在/data/tombstones目录下生成记录目标进程的堆栈、寄存器、内存占用等信息的墓碑文件。dropbox日志文件直接由产生的tombstone文件同步更新。

2.14.2. 机制

- 在系统启动完成发出启动完成广播后，BootReceiver类中的onReceive()开始响应，并创建一个独立线程调用logBootEvents()方法。
- logBootEvents()方法里会检测一次/data/tombstones目录，当有新的墓碑文件时，通过addFileToDropBox()方法调用addFileWithFootersToDropBox()方法来打印"SYSTEM_TOMBSTONE"标签文件。
- 之后构造一个观察器，在系统运行过程中持续监测/data/tombstones目录，当有墓碑文件更新时则继续通过addFileToDropBox()方法调用addFileWithFootersToDropBox()方法来打印"SYSTEM_TOMBSTONE"标签文件。

2.14.3. 时序

- 生成"SYSTEM_TOMBSTONE"标签的dropbox日志文件时序如下：

2.15. SYSTEM_FSCK

2.15.1. 背景

- 当系统启动中文件系统检查到错误时，会在"dev/fscklogs/log"中记录相关内容。

2.15.2. 机制

- 在系统启动完成发出启动完成广播后，BootReceiver类中的onReceive()开始响应，并创建一个独立线程调用logBootEvents()方法。
- logBootEvents()方法中在记录一些启动日志之后，通过addFsckErrorsToDropBoxAndLogFsStat()方法读取"dev/fscklogs/log"文件，当文件存在时则通过addFileToDropBox()方法调用addFileWithFootersToDropBox()方法来打印"SYSTEM_FSCK"标签文件。

2.15.3. 时序

- 生成"SYSTEM_FSCK"标签的dropbox日志文件时序如下：

2.16. BATTERY_DISCHARGE_INFO

2.16.1. 背景

- 电池管理服务主要负责检测电池电量、电池温度、充电动作等，当电池出现温度过高、电量过低且持续放电等情况，会触发报警。

2.16.2. 机制

- BatteryService在系统服务SystemServer启动核心服务的时候启动，构造时创建一些记录电池电量、电池温度、警告阈值等全局变量。之后在onStart()方法中创建BatteryListener向电源属性服务注册监听，用于监听底层电量属性变化。
- 当电量变化时，最终会回调到BinderService的update()方法中。update()方法中调用processValuesLocked()方法来更新电源信息。
- processValuesLocked()方法里会判断当前电量是否过低，以及是否处于充电状态。当电池持续放电导致电量过低且没有连接充电设备时，调用logOutlierLocked()方法记录电池信息。
- logOutlierLocked()方法会通过logBatteryStatsLocked()方法调用DBM的addFile()接口来生成"BATTERY_DISCHARGE_INFO"标签的dropbox文件。
- 参考资料： Android8.0.0-r4——BatteryService

2.16.3. 时序

- 生成"BATTERY_DISCHARGE_INFO"标签的dropbox日志文件时序如下：

2.17. 其他

- 除了上面的应用场合，目前在代码调查中和实际测试过程中还出现了一些其他dropbox文件的标签（后续更新补充）：
- "netstats_error" 网络状态相关(/frameworks/base/services/core/java/com/android/server/net/NetworkStatsService.java)
- "netstats_dump" 网络状态相关(/frameworks/base/services/core/java/com/android/server/net/NetworkStatsRecorder.java)
- "storage_benchmark" 存储相关(/frameworks/base/services/core/java/com/android/server/StorageManagerService.java)
- "storage_trim" 存储相关(/frameworks/base/services/core/java/com/android/server/StorageManagerService.java)
- "restricted_profile_ssaid" Setting相关(/frameworks/base/services/core/java/com/android/server/StorageManagerService.java)
- "incident" 事件报告？(/frameworks/base/cmds/incidentd/src/Reporter.cpp)
- "event_log" 事件相关？暂未看查到打印的地方
- "event_data" 事件相关？暂未查到打印的地方
- "keymaster" 密钥管理相关，暂未查到打印的地方