As an Android developer, I don’ t want to downgrade Samsung. I just want to find out what caused the accident and what we can learn from it. I think that it’ s necessary for the Android system to go wrong. From a technical point of view, why is there such a problem?

Conclusion first, for melon-eating people who don’ t like to read long texts, just look at the conclusion:

"The performance of this accident was that some of the Samsung system's mobile phone system's key system service process repeated Crash and forced to enter the Recovery interface. The key system service refers to Samsung's SystemUI process. When the SystemUI process initialized the AOD plug-in, AOD plug-in initialization error caused Crash Because it is a system service, after Crash reaches a certain number of times, it will be forced to enter the Recovery interface, so most users see the Recovery interface (pictured below)

``The full name of AOD is Always On Display, the Chinese translation is the screen display, which is the service that displays the time, weather, patterns, etc. on the screen after you press the power key to lock the screen.This function is only available on some high-end models.''

``The reason for the AOD plug-in Crash is that May 23, 2020 is leap April. When AOD displays the lunar calendar, it needs to display leap April, so it will go to the branch condition that shows the leap April, which is generally difficult to enter. After entering this condition, you need to obtain the common_data_leap_month string field (the corresponding value of this string field is "leap"), but due to a bug in the code compilation, the code cannot find this field in the compiled apk, so the process directly reports After FATAL EXCEPETION, the process restarts. After restarting, the initialization still needs to obtain this field, and then restart, so repeatedly, and finally trigger the system's self-help measures, enter the Recovery interface.

"This is why this problem only occurs for Chinese users, because AOD needs to display "Leap" April on May 23, but did not find the word "Leap", so it hanged. So it is not the millennium bug, also It’ s not that the server was hacked, and it’ s not Samsung’ s intentionally disgusting people. This kind of bug caused by compilation, and then encountered a leap month that happened once in a few years, it didn’ t come out. I will recognize it when I encounter it. I honestly apologize, not shameful.”

analysis

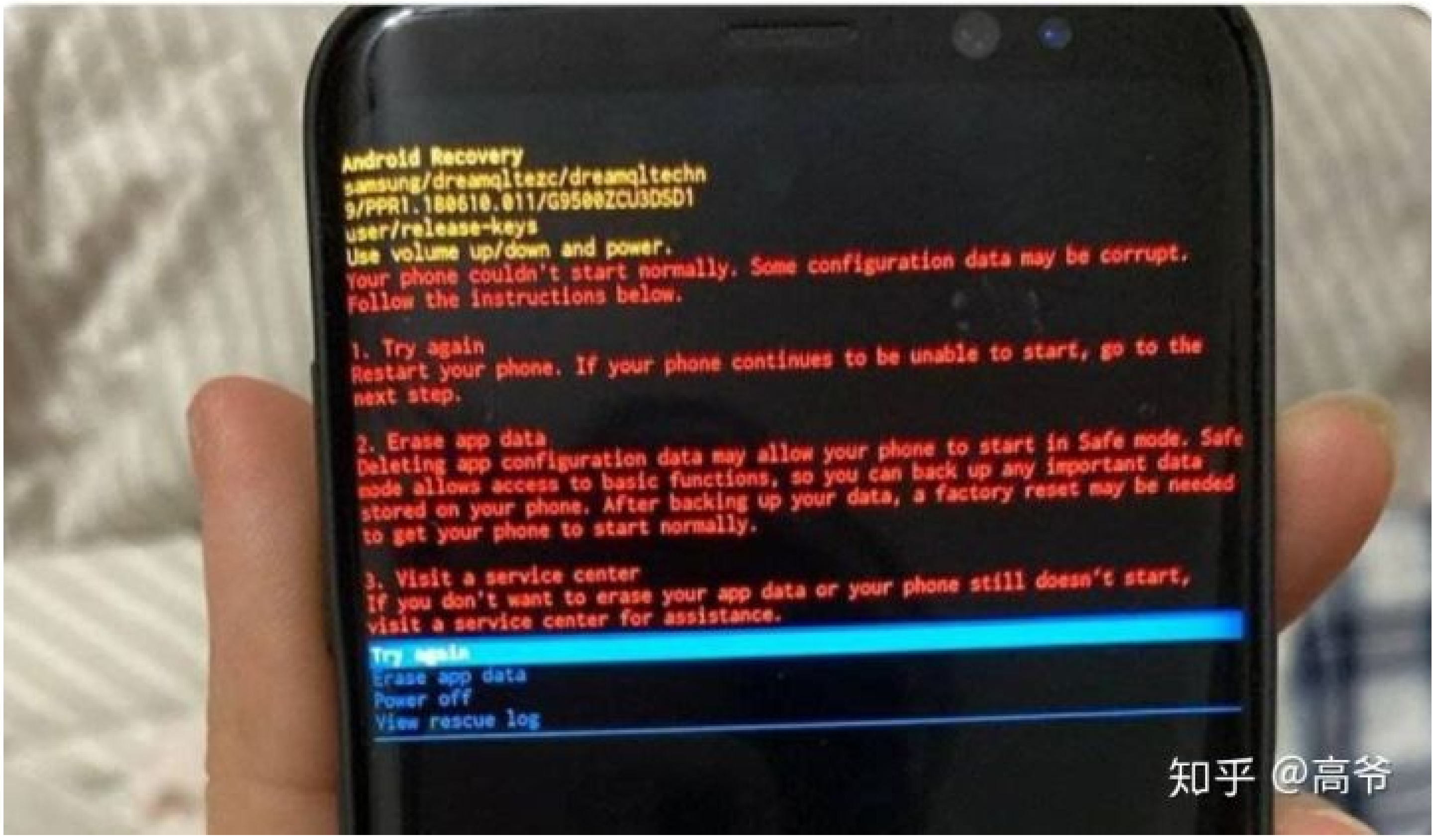
The people who eat melons can come back. Interested Android developers can continue to look down. Although the content is simple, I personally think that I can still take a look.

For Samsung developers, analyzing this Crash is very simple, just log in the monitor. From the later analysis, this problem is quickly found and repaired (it lasted about half a year); But for other developers interested in this issue, you need to use other tools

However, the analysis process is also very simple. Here, I will record my analysis ideas and tools used for everyone's convenience.

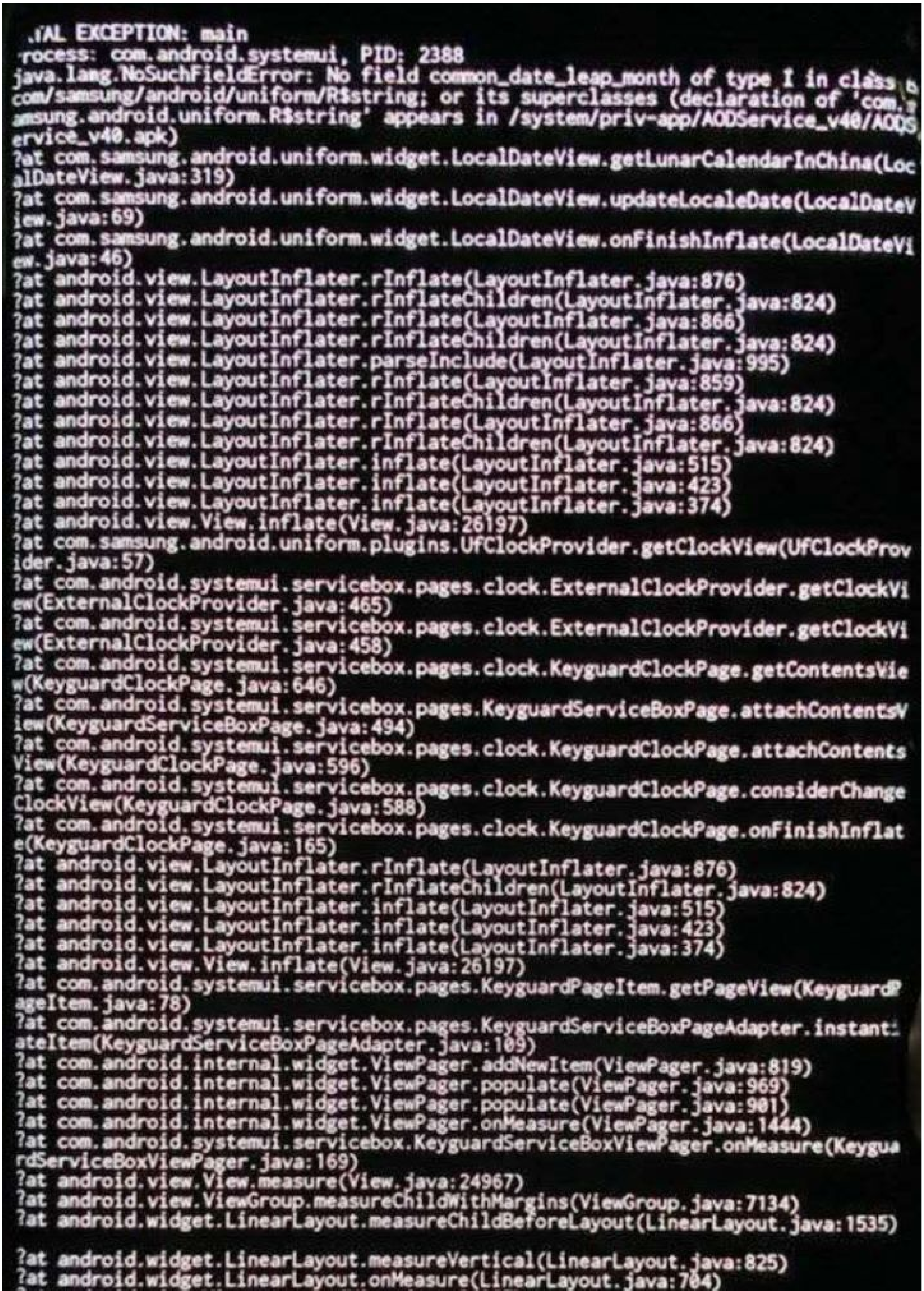
Start with phenomena

The above conclusion says that frequent Crash of the Persist process will cause the system to trigger self-help and enter the Recovery interface, so many user feedback screenshots you see are the Recovery interface, as follows

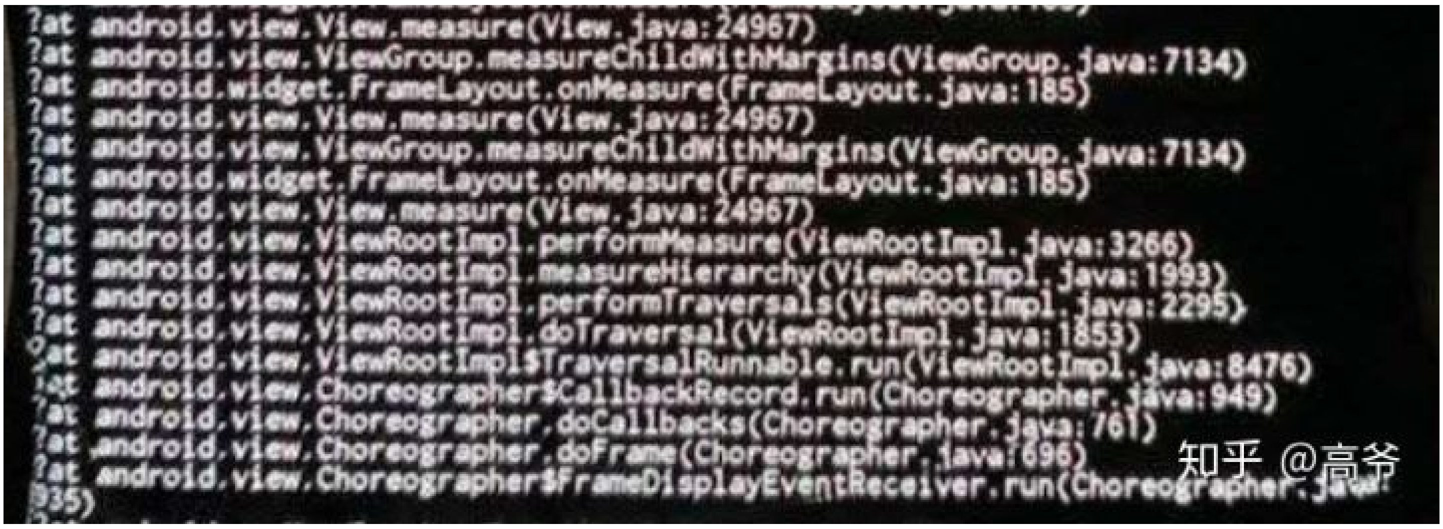


```
Android Recovery
samsung/dreamqltezc/dreamqltechn
9/PPR1.180610.011/G9500ZCU3DSD1
user/release-keys
Use volume up/down and power.
Your phone couldn't start normally. Some configuration data may be corrupt.
Follow the instructions below.

1. Try again
Restart your phone. If your phone continues to be unable to start, go to the
next step.

2. Erase app data
Deleting app configuration data may allow your phone to start in Safe mode. Safe
mode allows access to basic functions, so you can back up any important data
stored on your phone. After backing up your data, a factory reset may be needed
to get your phone to start normally.

3. Visit a service center
If you don't want to erase your app data or your phone still doesn't start,
visit a service center for assistance.
Try again
Erase app data
Power off
View rescue log
```

However, there are user interfaces that directly display the error message (I guess it is a function added by Samsung itself, let me know if you have any trouble), this interface is very important for us to analyze the code



```
.TAL EXCEPTION: main
rocess: com.android.systemui, PID: 2388
java.lang.NoSuchFieldError: No field common_date_leap_month of type I in class
com/samsung/android/uniform/R$string; or its superclasses (declaration of 'com.
amsung.android.uniform.R$string' appears in /system/priv-app/AODService_v40/AODS
ervice_v40.apk)
?at com.samsung.android.uniform.widget.LocalDateView.getLunarCalendarInChina(Loc
alDateView.java:319)
?at com.samsung.android.uniform.widget.LocalDateView.updateLocaleDate(LocalDateV
iew.java:69)
?at com.samsung.android.uniform.widget.LocalDateView.onFinishInflate(LocalDateVi
ew.java:46)
?at android.view.LayoutInflater.rInflate(LayoutInflater.java:876)
?at android.view.LayoutInflater.rInflateChildren(LayoutInflater.java:824)
?at android.view.LayoutInflater.rInflate(LayoutInflater.java:866)
?at android.view.LayoutInflater.rInflateChildren(LayoutInflater.java:824)
?at android.view.LayoutInflater.parseInclude(LayoutInflater.java:995)
?at android.view.LayoutInflater.rInflate(LayoutInflater.java:859)
?at android.view.LayoutInflater.rInflateChildren(LayoutInflater.java:824)
?at android.view.LayoutInflater.rInflate(LayoutInflater.java:866)
?at android.view.LayoutInflater.rInflateChildren(LayoutInflater.java:824)
?at android.view.LayoutInflater.inflate(LayoutInflater.java:515)
?at android.view.LayoutInflater.inflate(LayoutInflater.java:423)
?at android.view.LayoutInflater.inflate(LayoutInflater.java:374)
?at android.view.View.inflate(View.java:26197)
?at com.samsung.android.uniform.plugins.UfClockProvider.getClockView(UfClockProv
ider.java:57)
?at com.android.systemui.servicebox.pages.clock.ExternalClockProvider.getClockVi
ew(ExternalClockProvider.java:465)
?at com.android.systemui.servicebox.pages.clock.ExternalClockProvider.getClockVi
ew(ExternalClockProvider.java:458)
?at com.android.systemui.servicebox.pages.clock.KeyguardClockPage.getContentsVie
w(KeyguardClockPage.java:646)
?at com.android.systemui.servicebox.pages.KeyguardServiceBoxPage.attachContentsV
iew(KeyguardServiceBoxPage.java:494)
?at com.android.systemui.servicebox.pages.clock.KeyguardClockPage.attachContents
View(KeyguardClockPage.java:596)
?at com.android.systemui.servicebox.pages.clock.KeyguardClockPage.considerChange
ClockView(KeyguardClockPage.java:588)
?at com.android.systemui.servicebox.pages.clock.KeyguardClockPage.onFinishInflat
e(KeyguardClockPage.java:165)
?at android.view.LayoutInflater.rInflate(LayoutInflater.java:876)
?at android.view.LayoutInflater.rInflateChildren(LayoutInflater.java:824)
?at android.view.LayoutInflater.inflate(LayoutInflater.java:515)
?at android.view.LayoutInflater.inflate(LayoutInflater.java:423)
?at android.view.LayoutInflater.inflate(LayoutInflater.java:374)
?at android.view.View.inflate(View.java:26197)
?at com.android.systemui.servicebox.pages.KeyguardPageItem.getPageView(KeyguardP
ageItem.java:78)
?at com.android.systemui.servicebox.pages.KeyguardServiceBoxPageAdapter.instanti
ateItem(KeyguardServiceBoxPageAdapter.java:109)
?at com.android.internal.widget.ViewPager.addNewItem(ViewPager.java:819)
?at com.android.internal.widget.ViewPager.populate(ViewPager.java:969)
?at com.android.internal.widget.ViewPager.populate(ViewPager.java:901)
?at com.android.internal.widget.ViewPager.onMeasure(ViewPager.java:1444)
?at com.android.systemui.servicebox.KeyguardServiceBoxViewPager.onMeasure(Keygua
rdServiceBoxViewPager.java:169)
?at android.view.View.measure(View.java:24967)
?at android.view.ViewGroup.measureChildWithMargins(ViewGroup.java:7134)
?at android.widget.LinearLayout.measureChildBeforeLayout(LinearLayout.java:1535)

?at android.widget.LinearLayout.measureVertical(LinearLayout.java:825)
?at android.widget.LinearLayout.onMeasure(LinearLayout.java:704)
?at android.view.View.measure(View.java:24967)
?at android.view.ViewGroup.measureChildWithMargins(ViewGroup.java:7134)
?at android.widget.FrameLayout.onMeasure(FrameLayout.java:185)
```



```
?at android.view.View.measure(View.java:24967)
?at android.view.ViewGroup.measureChildWithMargins(ViewGroup.java:7134)
?at android.widget.FrameLayout.onMeasure(FrameLayout.java:185)
?at android.view.View.measure(View.java:24967)
?at android.view.ViewGroup.measureChildWithMargins(ViewGroup.java:7134)
?at android.widget.FrameLayout.onMeasure(FrameLayout.java:185)
?at android.view.View.measure(View.java:24967)
?at android.view.ViewRootImpl.performMeasure(ViewRootImpl.java:3266)
?at android.view.ViewRootImpl.measureHierarchy(ViewRootImpl.java:1993)
?at android.view.ViewRootImpl.performTraversals(ViewRootImpl.java:2295)
?at android.view.ViewRootImpl.doTraversal(ViewRootImpl.java:1853)
?at android.view.ViewRootImpl$TraversalRunnable.run(ViewRootImpl.java:8476)
?at android.view.Choreographer$CallbackRecord.run(Choreographer.java:949)
?at android.view.Choreographer.doCallbacks(Choreographer.java:761)
?at android.view.Choreographer.doFrame(Choreographer.java:696)
?at android.view.Choreographer$FrameDisplayEventReceiver.run(Choreographer.java:
935)
```

Developers are most familiar with this stack. This is a one-frame rendering process. During the initialization of AOD's LocalDateView, an error occurs when calling the getLunarCalendarInChina method. LunarCalendar means lunar. string value.

Then the problem is very clear, we only need to check the following points

1. common_data_leap_month The code logic that appears in the string field
2. common_data_leap_month this string field was not found to cause the operation error

Analysis code

First look at the code logic that appears in the common_data_leap_month field. Since the function stack is listed above, then we need to directly view the code to analyze the logic generated by this problem. How do I get the code? Naturally need to decompile, the recommended decompilation tool: TTDeDroid

Decompilation requires Samsung AOD code, you can search for Always-On-Display in ApkMirror, you can find the corresponding file, you can see that Samsung's AOD update frequency is still very frequent, through user feedback can know that not all users There is this problem, and there is no problem to update to the new version, then we speculate that the problem lies in the old version (from the stack to guess that it should be V4.0 version)

Normal version_V5.2.05

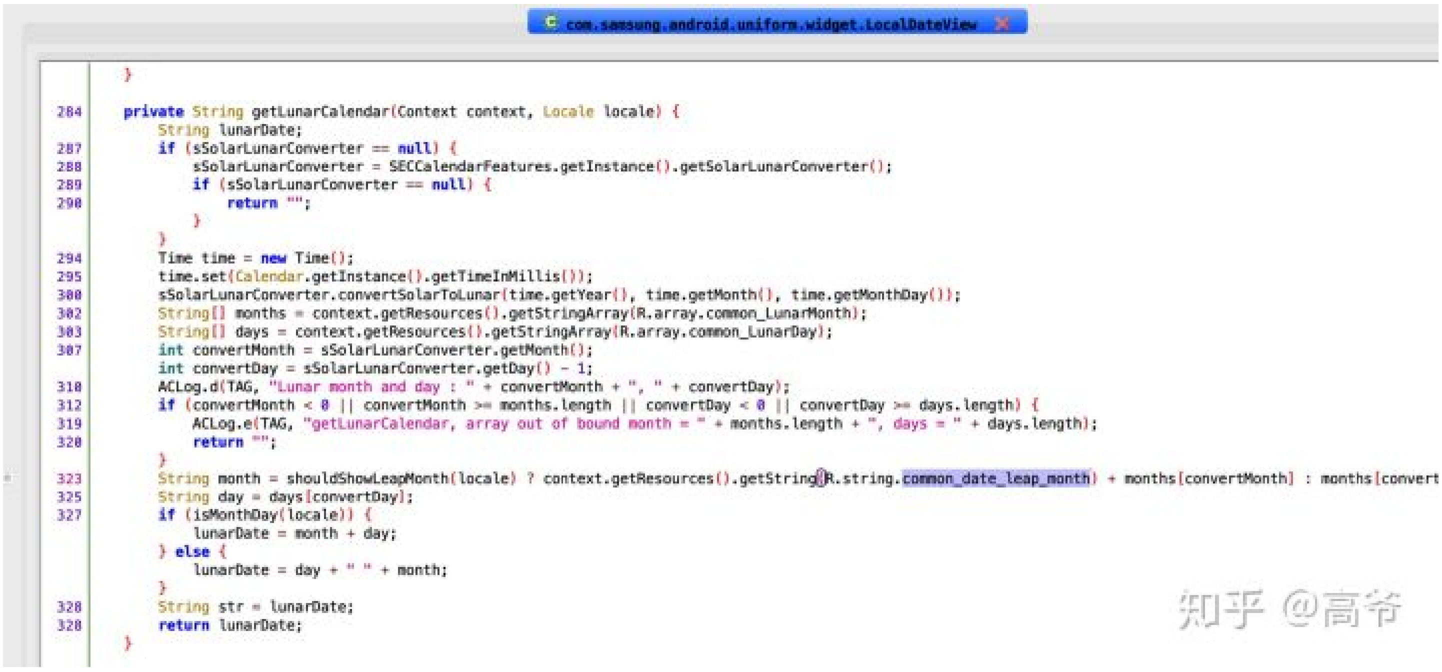
The latest version is normal, without Crash

First let's take a look at the logic of the latest version of this code

```
String month = shouldShowLeapMonth(locale) ? context.getResources().getString(R.string.
```

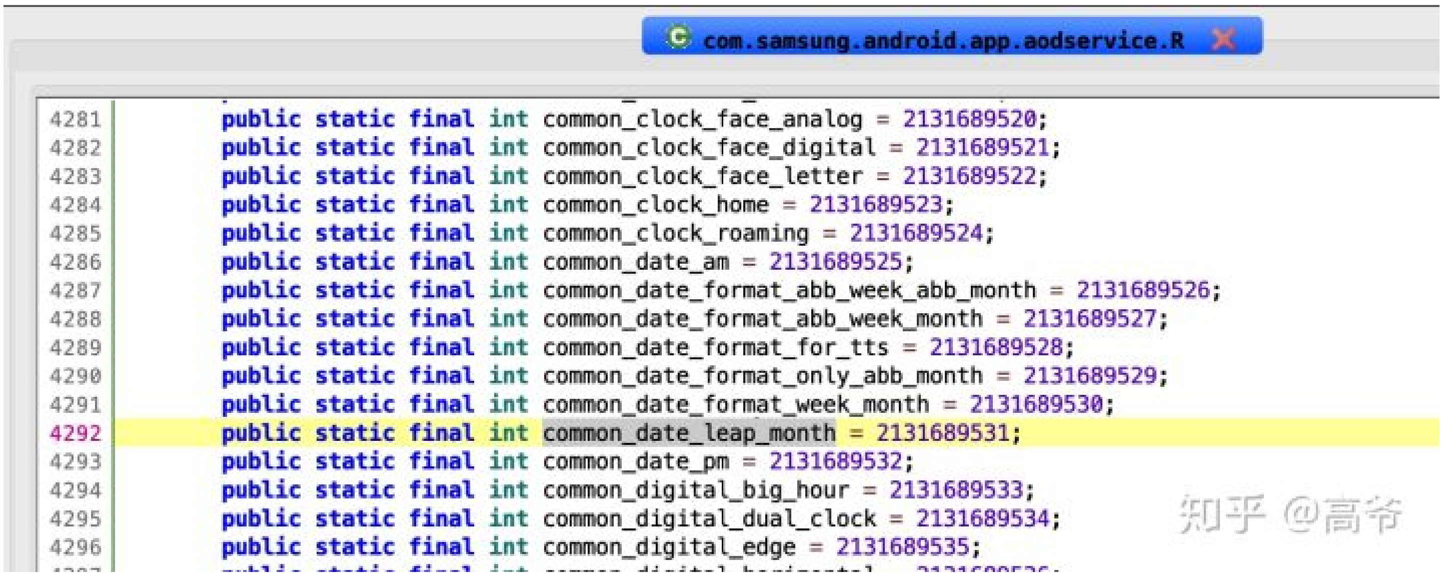
This means that if you need to display the leap month, then take the value of common_date_leap_month, and search for common_date_leap_month globally and find that this value is defined in the latest version.

Here you can see the code logic of the common_data_leap_month field: ``**Only when the leap month needs to be displayed, the value of the common_date_leap_month field will be obtained, and the acquisition of this value will not be triggered 99.9% of the time''** .

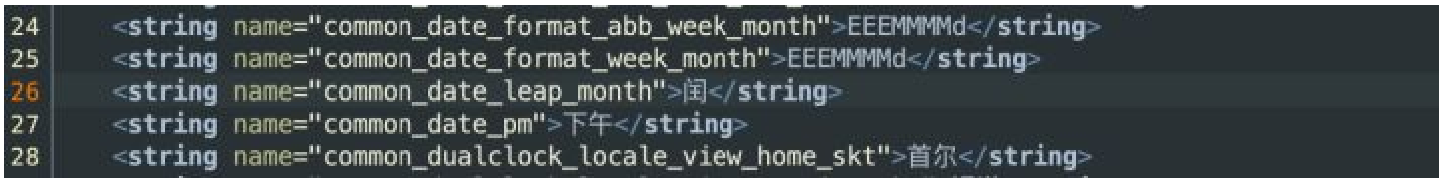


```java
com.samsung.android.uniform.widget.LocalDateView

    }

    private String getLunarCalendar(Context context, Locale locale) {
        String lunarDate;
        if (sSolarLunarConverter == null) {
            sSolarLunarConverter = SECCalendarFeatures.getInstance().getSolarLunarConverter();
            if (sSolarLunarConverter == null) {
                return "";
            }
        }
        Time time = new Time();
        time.set(Calendar.getInstance().getTimeInMillis());
        sSolarLunarConverter.convertSolarToLunar(time.getYear(), time.getMonth(), time.getMonthDay());
        String[] months = context.getResources().getStringArray(R.array.common_LunarMonth);
        String[] days = context.getResources().getStringArray(R.array.common_LunarDay);
        int convertMonth = sSolarLunarConverter.getMonth();
        int convertDay = sSolarLunarConverter.getDay() - 1;
        ACLog.d(TAG, "Lunar month and day : " + convertMonth + ", " + convertDay);
        if (convertMonth < 0 || convertMonth >= months.length || convertDay < 0 || convertDay >= days.length) {
            ACLog.e(TAG, "getLunarCalendar, array out of bound month = " + months.length + ", days = " + days.length);
            return "";
        }
        String month = shouldShowLeapMonth(locale) ? context.getResources().getString(R.string.common_date_leap_month) + months[convertMonth] : months[convert
        String day = days[convertDay];
        if (isMonthDay(locale)) {
            lunarDate = month + day;
        } else {
            lunarDate = day + " " + month;
        }
        String str = lunarDate;
        return lunarDate;
    }
```

The common_date_leap_month exists in the R.java file, indicating that it exists, check the corresponding string.xml also has the definition of this field



```java
com.samsung.android.app.aodservice.R

        public static final int common_clock_face_analog = 2131689520;
        public static final int common_clock_face_digital = 2131689521;
        public static final int common_clock_face_letter = 2131689522;
        public static final int common_clock_home = 2131689523;
        public static final int common_clock_roaming = 2131689524;
        public static final int common_date_am = 2131689525;
        public static final int common_date_format_abb_week_abb_month = 2131689526;
        public static final int common_date_format_abb_week_month = 2131689527;
        public static final int common_date_format_for_tts = 2131689528;
        public static final int common_date_format_only_abb_month = 2131689529;
        public static final int common_date_format_week_month = 2131689530;
        public static final int common_date_leap_month = 2131689531;
        public static final int common_date_pm = 2131689532;
        public static final int common_digital_big_hour = 2131689533;
        public static final int common_digital_dual_clock = 2131689534;
        public static final int common_digital_edge = 2131689535;
```

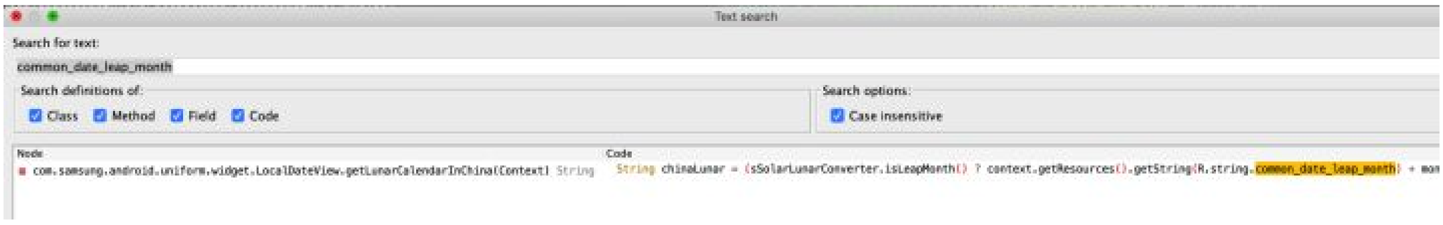


```xml
    <string name="common_date_format_abb_week_month">EEEMMMd</string>
    <string name="common_date_format_week_month">EEEMMMMd</string>
    <string name="common_date_leap_month">闰</string>
    <string name="common_date_pm">下午</string>
    <string name="common_dualclock_locale_view_home_skt">首尔</string>
```

The problematic version_V4.1.70

Since there is no problem with the new version, and we also know the code logic of the common_date_leap_month field, then we look at the reason why the common_date_leap_month field is not found from the old version.

The old version found here is problematic. Users who use this version (these versions) will have frequent crashes on the 23rd. The reason why I think he is problematic is because the global search for the common_date_leap_month field found that There is no corresponding field in the R file, and there is no corresponding field and its corresponding value in the corresponding string.xml, that is, the code here is only used, and there is no definition or assignment (how did it be compiled??)



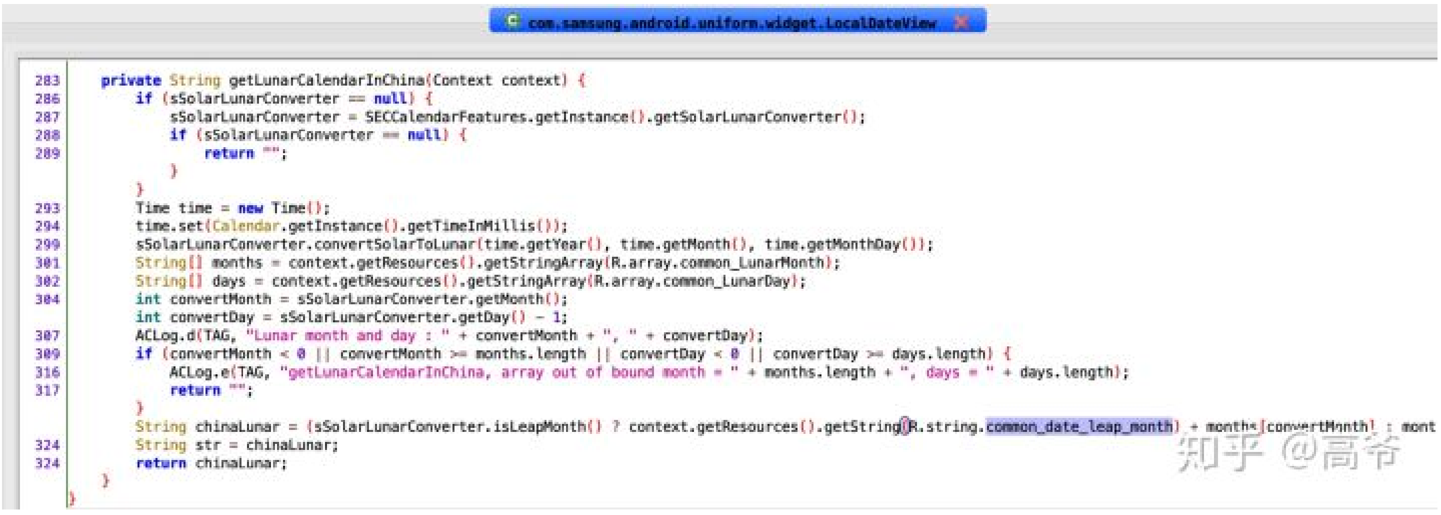
Search for text: common_date_leap_month

Search definitions of: [✓ Class] [✓ Method] [✓ Field] [✓ Code]

Search options: [✓ Case insensitive]

```
Node                                                                          Code
com.samsung.android.uniform.widget.LocalDateView.getLunarCalendarInChina(Context) String    String chinaLunar = (sSolarLunarConverter.isLeapMonth() ? context.getResources().getString(R.string.common_date_leap_month) + mon
```

The corresponding code logic above is as follows, you can see that the function name and the number of lines are consistent with the error report, InChina...



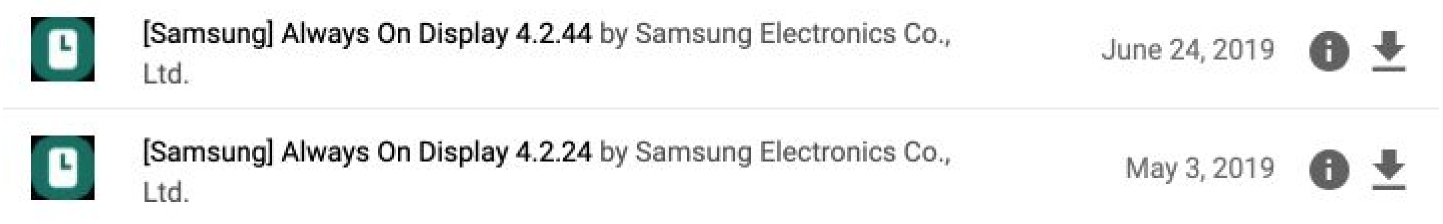
```
283     private String getLunarCalendarInChina(Context context) {
286         if (sSolarLunarConverter == null) {
287             sSolarLunarConverter = SECCalendarFeatures.getInstance().getSolarLunarConverter();
288             if (sSolarLunarConverter == null) {
289                 return "";
                }
            }
293         Time time = new Time();
294         time.set(Calendar.getInstance().getTimeInMillis());
299         sSolarLunarConverter.convertSolarToLunar(time.getYear(), time.getMonth(), time.getMonthDay());
301         String[] months = context.getResources().getStringArray(R.array.common_LunarMonth);
302         String[] days = context.getResources().getStringArray(R.array.common_LunarDay);
304         int convertMonth = sSolarLunarConverter.getMonth();
            int convertDay = sSolarLunarConverter.getDay() - 1;
307         ACLog.d(TAG, "Lunar month and day : " + convertMonth + ", " + convertDay);
309         if (convertMonth < 0 || convertMonth >= months.length || convertDay < 0 || convertDay >= days.length) {
316             ACLog.e(TAG, "getLunarCalendarInChina, array out of bound month = " + months.length + ", days = " + days.length);
317             return "";
            }
            String chinaLunar = (sSolarLunarConverter.isLeapMonth() ? context.getResources().getString(R.string.common_date_leap_month) + months[convertMonth] : mont
324         String str = chinaLunar;
324         return chinaLunar;
        }
    }
```

The corresponding code:

```
private String getLunarCalendarInChina(Context context) {
    if (sSolarLunarConverter == null) {
        sSolarLunarConverter = SECCalendarFeatures.getInstance().getSolarLunarConverter
        if (sSolarLunarConverter == null) {
            return "";
        }
    }
    Time time = new Time();
    time.set(Calendar.getInstance().getTimeInMillis());
    sSolarLunarConverter.convertSolarToLunar(time.getYear(), time.getMonth(), time.getM
    String[] months = context.getResources().getStringArray(R.array.common_LunarMonth);
    String[] days = context.getResources().getStringArray(R.array.common_LunarDay);
    int convertMonth = sSolarLunarConverter.getMonth();
    int convertDay = sSolarLunarConverter.getDay() - 1;
    ACLog.d(TAG, "Lunar month and day : " + convertMonth + ", " + convertDay);
    if (convertMonth < 0 || convertMonth >= months.length || convertDay < 0 || convertD
        ACLog.e(TAG, "getLunarCalendarInChina, array out of bound month = " + months.le
        return "";
    }
    String chinaLunar = (sSolarLunarConverter.isLeapMonth() ? context.getResources().ge
    String str = chinaLunar;
    return chinaLunar;
}
```

When the problem occurred

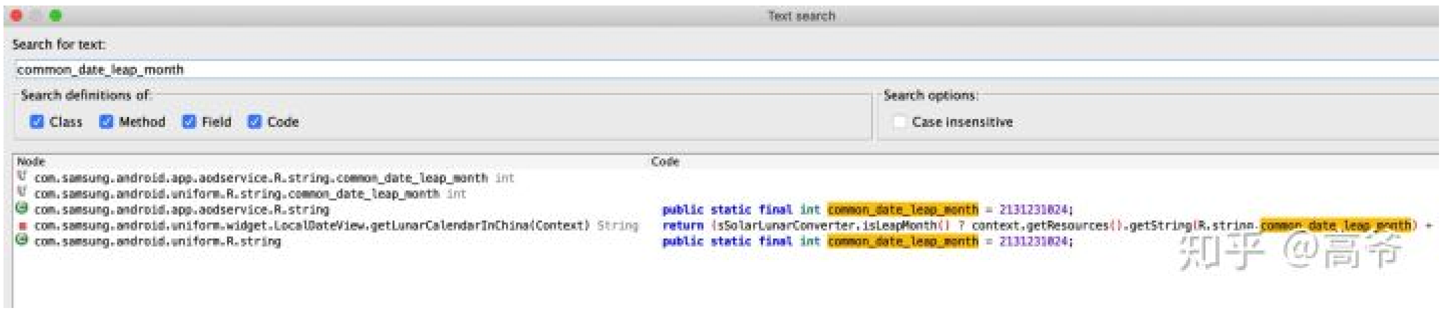
According to my investigation here, I found that this problem actually occurred when the AOD application was upgraded from v3.3.18 to v4.0.57, but there has been no problem in the middle. Without leap April, users will not have problems. , The test was not measured, and it was not fixed until v4.2.44 released on June 24, 2019



[Samsung] Always On Display 4.2.44 by Samsung Electronics Co., Ltd. June 24, 2019

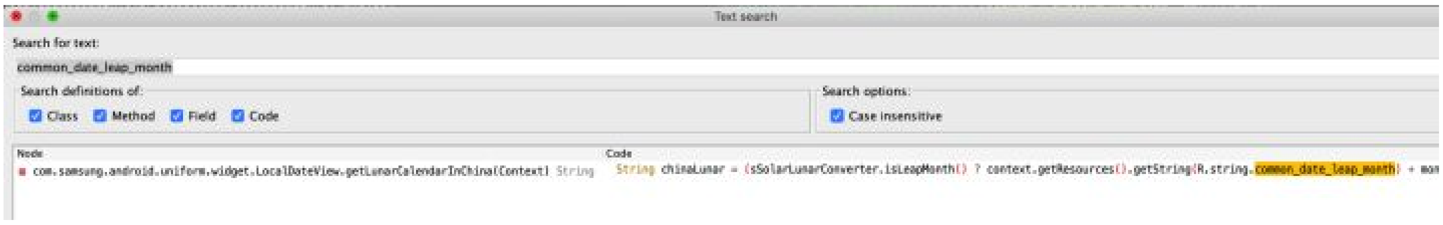
[Samsung] Always On Display 4.2.24 by Samsung Electronics Co., Ltd. May 3, 2019

As of v3.3.18 we can see that the common_date_leap_month field still exists



```
Text search
Search for text:
common_date_leap_month
Search definitions of:                          Search options:
[✓] Class [✓] Method [✓] Field [✓] Code           [ ] Case insensitive

Node                                                                        Code
com.samsung.android.app.aodservice.R.string.common_date_leap_month int
com.samsung.android.uniform.R.string.common_date_leap_month int
com.samsung.android.app.aodservice.R.string                                 public static final int common_date_leap_month = 2131231024;
com.samsung.android.uniform.widget.LocalDateView.getLunarCalendarInChina(Context) String    return (sSolarLunarConverter.isLeapMonth() ? context.getResources().getString(R.string.common_date_leap_month) +
com.samsung.android.uniform.R.string                                        public static final int common_date_leap_month = 2131231024;
```

After upgrading to v4.0.57 (the first version with problems), this field is gone (how did you compile it??)



```
Text search
Search for text:
common_date_leap_month
Search definitions of:                          Search options:
[✓] Class [✓] Method [✓] Field [✓] Code           [✓] Case insensitive

Node                                                                        Code
com.samsung.android.uniform.widget.LocalDateView.getLunarCalendarInChina(Context) String    String chinaLunar = (sSolarLunarConverter.isLeapMonth() ? context.getResources().getString(R.string.common_date_leap_month) + mon
```

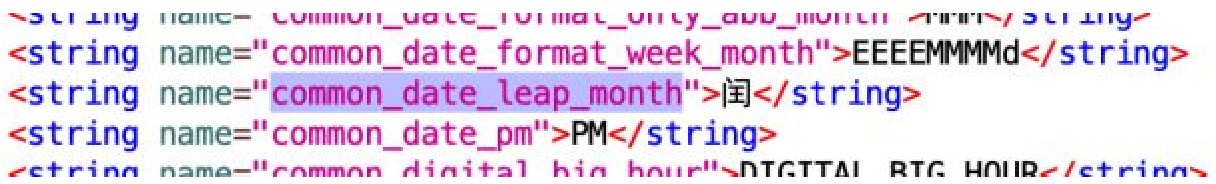
"Take care" :

1. AOD upgrade from v3.3.18 to v4.0.57 introduced this problem (introduced on October 24, 2018)
2. AOD upgrade from v4.2.24 to v4.2.44 solves this problem (fixed on June 24, 2019)

During this period, all models with AOD versions from v4.0.57 to v4.2.44 that have never been upgraded will enter Recovery mode on the day of May 23, 2020.

Compilation problem

A very important point above is the compilation problem. Android developers know that if I write getString(R.string.common_date_leap_month) in the code, then I have to define this common_date_leap_month in strings.xml, and then give him a value, such as " "Leap", so that we can see this field in the R file, we can use the syntax like getString(R.string.common_date_leap_month) to call; otherwise there will be a problem during the compilation phase, the compilation prompt R.string.common_date_leap_month does not exist



```
<string name="common_date_format_week_month">EEEEMMMMd</string>
<string name="common_date_leap_month">闰</string>
<string name="common_date_pm">PM</string>
```

However, through the above analysis, we found that the frequent Crash version is Crash because it can't find the common_date_leap_month field. Since it can't be found, it should be compiled, but since we got the apk, it means that the compilation is no problem. .

If this situation occurs, there are generally the following two situations

1. There are different versions of the same jar package in the project, so different jar packages are used during compilation and runtime
2. The compilation uses Maven, because the dependencies in the project use different versions of the package, the last package is not the version you need

Guess Samsung's problem this time is because of the second situation, the main project and sub-modules use different versions of the package, which can be compiled through, but the final package into the project is not the package at compile time, and the FATAL at runtime appears EXCEPTION: NoSuchFieldError (If you have specific reasons, you can leave a message to discuss a wave)

to sum up

Although the above analysis process is relatively simple, but there are some more tedious work, such as finding versions, decompilation, looking at the code logic, etc. In the end, it can be regarded as the root cause of the problem: the problem caused by the compilation only encountered once in a few years Leap month. So what have we learned from this?

1. Function test: Leap month is a function in the calendar, not a commonly used function, but relatively professional, such as this involves professional, you must be cautious, list all possible situations to test, necessary situations Next, leave it to a professional to evaluate test cases
2. For projects that rely on multiple parties to compile, make sure that the referenced version is consistent with the local version when compiling. For modules that rely on multiple parties, it is best to confirm with the corresponding dependent module before issuing the version each time.
3. SystemUI (lock screen\status bar\gestures\multitasking\AOD, etc.) modules and desktop modules are modules that users can directly feel. These modules have very high requirements for stability, because once these modules occur FATAL, the The impact is very huge, just like Samsung this time, so the developers of these modules are also the hardest, not only to undertake the implementation of some highlight functions, but also to ensure stability, but also in the middle of system development and application development, There is a great coupling on both sides, which is really not easy (the wife has been making this for more than 6 years, adding a drumstick at night...)
4. The system update provided by the manufacturer and the application update (especially the system application) of the manufacturer must be updated in time. Every time the system and system application update generally fix many bugs, enhance stability and performance. There is no profitable pressure on the system and system application. , So the update is mainly to improve the quality, you can safely update.
5. Developers should be curious and awe-inspiring about this kind of things: curiosity can help me learn a lot, and see the essence through phenomena; awe-inspiring can let me know the lack of my knowledge. In the huge Android system, I know nothing but the sea. .
6. This problem is definitely a big accident for Samsung, but it also contributed a classic case. It is estimated that other App or mobile phone manufacturers will include this in the functional test. As for Samsung, the domestic market itself will not work, S20 series I just got warmed up again, and this thing happened again, still the sentence: This is life, confession, apology, not shameful."

Edited on 2020-05-28