输入框与键盘错位问题技术报告

Android 16 平板横屏 + 微信双栏（平行视界）布局下，IME 弹出时输入框与键盘之间出现约 500 px 空白

项目	内容
报告日期	2026-11-15
报告性质	用户侧实测技术反馈，供厂商定位问题参考
报告对象	搜狗输入法团队（com.sohu.inputmethod.sogou.oem）；联想 / ZUI 系统团队
设备	Lenovo TB322FC（Legion Y700 系列）
系统	Android 16（API 36），ZUXOS 1.5.10.259
构建号	TB322FC_CN_OPEN_USER_Q00041.1_W_ZUXOS_1.5.10.259_ST_260611
屏幕	3040 × 1904 px，440 dpi（约 1105 × 692 dp），横屏
输入法版本	搜狗输入法（联想定制版）1.0.android_pad_lenovo_2024.20260423163610，versionCode 29615556，targetSdk 29
应用版本	微信 8.0.80，versionCode 3220，targetSdk 34

说明：本报告所有窗口与 Insets 数据均来自 adb shell dumpsys window / dumpsys input_method 实测；所有截图均已做隐私脱敏处理（会话列表、聊天内容、头像、账号信息、网络名称均已遮盖或裁剪）。

1 摘要

在上述设备横屏、微信处于**双栏（左侧会话列表 + 右侧聊天区，“平行视界”）**布局时，键盘弹出后微信聊天输入框并未贴靠键盘，而是悬停在屏幕中部。输入框底部与键盘顶部之间存在 **501 px（约 182 dp）** 的纯空白区域，用户输入时视线需要大幅上下移动，交互体验严重受损。

该问题在**微信单栏布局、以及其他应用（如系统设置搜索）中均不复现**；同一场景下更换不同输入法，错位量随 Insets 数值变化，可确认属 **IME Insets 的消费 / 上报配合问题**，而非单一应用渲染缺陷。

2 问题现象



图 1（已脱敏）：微信双栏聊天页 + 搜狗输入法键盘。输入框 y=761…890，键盘顶部 y=1391，二者之间空白 **501 px**。

键盘本身渲染完整、按键响应正常（可正常输入英文与中文候选），仅输入框落点错误。

3 复现步骤

- 1. 设备保持横屏（3040 × 1904）。
- 2. 打开微信 8.0.80，进入任意聊天会话，使界面呈现“左会话列表 + 右聊天区”双栏布局。
- 3. 点击右侧聊天区的输入框，弹出搜狗输入法。
- 4. 观察输入框位置：输入框悬停于屏幕中部，与键盘之间出现约 501 px 空白。

复现率：100%（多次冷启动、重启输入法后一致）。

4 技术分析

4.1 实测窗口与 Insets 数据

搜狗输入法 IME 窗口（dumpsys window，节选）：

```
Window{... u0 InputMethod}  package=com.sohu.inputmethod.sogou.oem
  mAttrs={(0,0)(fillxfill) gr=BOTTOM CENTER_VERTICAL sim={adjust=pan} ty=INPUT_METHOD ...
    fl=NOT_FOCUSABLE LAYOUT_IN_SCREEN SPLIT_TOUCH HARDWARE_ACCELERATED
```

```
DRAWS_SYSTEM_BAR_BACKGROUNDS
    pfl=FIT_INSETS_CONTROLLED
    fitTypes=STATUS_BARS NAVIGATION_BARS
    fitSides=LEFT TOP RIGHT
  Requested w=3040 h=1797
  Frames: parent=[0,107][3040,1904] display=[0,107][3040,1904] frame=[0,107][3040,1904]
  mGivenContentInsets=[0,1284][0,0]  mGivenVisibleInsets=[0,1284][0,0]
  mTouchableInsets=3
  touchable region=SkRegion((0,1391,3040,1904))
```

InputMethodService 自算 Insets（dumpsys input_method）：

```
Last computed insets:
  contentTopInsets=1284  visibleTopInsets=1284
  touchableInsets=3  touchableRegion=SkRegion((0,1284,3040,1665))
```

系统最终下发的 IME InsetsSource：

```
InsetsSource id=3 type=ime frame=[0,1391][3040,1904] visibleFrame=[0,1391][3040,1904] visible=true
InsetsSourceControl ... mSurfacePosition=Point(0, 107)
  mInsetsHint=Insets{left=0, top=0, right=0, bottom=513}
```

微信聊天窗口：

```
Window{... com.tencent.mm/com.tencent.mm.ui.chatting.ChattingUI}
  mAttrs={(0,0)(fillxfill) sim={state=hidden adjust=resize forwardNavigation} ...}
  Frames: parent=[912,0][3040,1904] display=[912,0][3040,1904] frame=[912,0][3040,1904]
  mWindowingMode=multi-window（ActivityEmbedding 内嵌 Activity）
  mLastReportedActivityWindowInfo=ActivityWindowInfo{isEmbedded=true,
      taskBounds=Rect(0,0-3040,1904), taskFragmentBounds=Rect(912,0-3040,1904)}
```

数据解读：

- 搜狗窗口顶部为 **y=107**（拟合状态栏），窗口高度 1797；
- contentTopInsets = 1284（相对窗口）→ 键盘可见顶部 = 107 + 1284 = **1391**，与截图完全一致；
- 系统据此下发 ime inset bottom = 1904 − 1391 = **513**，与物理键盘高度一致，属正确上报；
- 微信窗口为全高 1904，理论上应把输入框放在 1391 上方，但实际落在 890。

4.2 交叉对照实验

在同一聊天页、同一设备上替换输入法，得到如下对照（输入框落点以截图测量）：

输入法	ime inset (px)	visibleFrame.top	微信输入框底 (px)	表现
搜狗定制版	513	1391	890	悬空 501 px
AOSP 拉丁输入法	1035	869	未随键盘上移	被键盘遮挡
悬浮式输入法（inset frame 为空）	0	1111	1110	正确贴合

关键规律：只有当 IME 的 **InsetsSource.frame 坍缩为空（inset = 0）、而 visibleFrame 仍保留真实键盘顶部**时，输入框才落在正确位置。由此推断微信当前的落点计算为：

```
composerBottom ≈ visibleFrame.top – imeInsetHeight
```

在正常输入法下 imeInsetHeight ≈ 键盘高度，而 visibleFrame.top 本身已反映键盘遮挡高度，**二者重复扣减**，导致输入框上移约一个键盘高度。

4.3 其他应用正常（排除输入法整体缺陷）

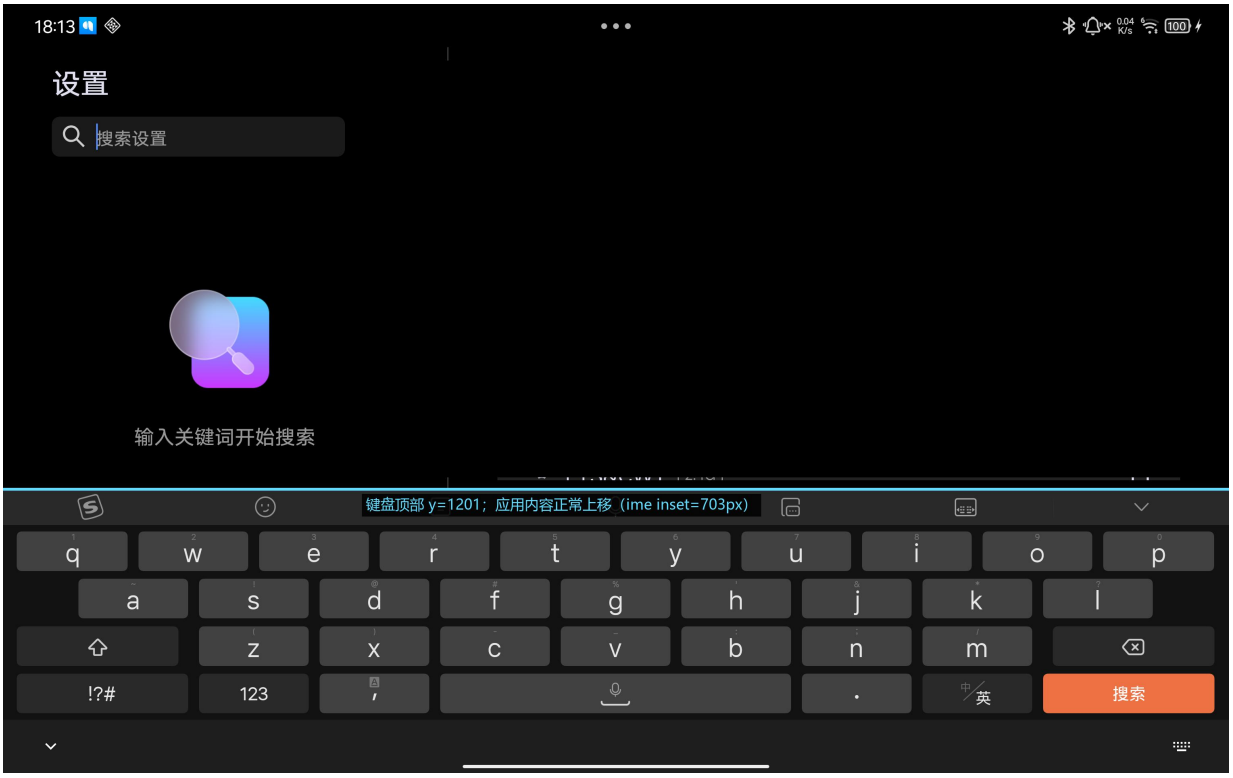


图 2（已脱敏）：系统设置搜索页 + 同一搜狗输入法，ime inset = 703 px，内容正常上移，无错位。

```
InsetsSource id=3 type=ime frame=[0,1201][3040,1904]   → ime inset = 703
```

设置页内容正常上移，键盘顶部 1201，无任何空白错位。说明搜狗输入法在普通应用中的 Insets 上报是正常的，问题只在微信双栏（ActivityEmbedding）场景触发。

4.4 AOSP 输入法的对照说明

AOSP 拉丁输入法的 IME 窗口为 frame=[0,0][3040,1904]（全屏、不拟合状态栏），contentTopInsets=762，下发 ime inset=1035。此时微信输入框完全不上移、被键盘遮挡，是另一种错误形态。

两种 inset 区间（513 / 1035）下微信均不能正确布局，说明微信对 IME Insets 的处理在该布局下普遍不成立，而非仅对某一种数值敏感。

5 问题定位小结

环节	观察	判断
搜狗输入法 Insets 上报	inset=513，与物理键盘一致	数据正确；但窗口拟合状态栏（frame.top=107）与 AOSP（0）不同，可能使微信走入不同分支
系统（ZUI）ActivityEmbedding	微信为内嵌 Activity（taskFragmentBounds=[912,0-3040,1904]）	IME Insets 在分栏坐标映射环节建议复核
微信布局计算	输入框位置随 inset 与 visibleFrame 双重变化	重复扣减，是错位的直接原因

6 修复建议

6.1 微信侧（最优解）

IME 的 WindowInsets.Type.ime() 底部值已等价于“键盘遮挡高度”，不应再与 getWindowVisibleDisplayFrame() 叠加使用。建议二选一：

```
// 方案 A：只用 ime insets
Insets ime = rootInsets.getInsets(WindowInsets.Type.ime());
composerBottom = windowHeight - ime.bottom;

// 方案 B：只用 visible display frame
Rect visible = new Rect();
view.getWindowVisibleDisplayFrame(visible);
composerBottom = visible.bottom;
```

同时需确认在 ActivityEmbedding（isEmbedded=true）下，insets 以**所在 taskFragment 的坐标**而非全屏坐标计算。

6.2 搜狗输入法侧（兼容性规避）

在 InputMethodService.onComputeInsets(Insets outInsets) 中，可对微信平板双栏场景做兼容处理：**保持 visibleTopInsets 为真实键盘顶部不变，将 contentTopInsets 提升至不小于窗口高度**，使 IME InsetsSource 的 frame 坍缩为空，同时保留 visibleFrame。

按此思路在实验构建上验证，结果为：



图 3（已脱敏；该次验证键盘为分屏样式，仅用于展示几何关系）：输入框 y=1078…1200，键盘顶部 y=1201，间隙 **1 px**，精确贴合。

该处理仅影响目标应用，其他应用（如图 2 的设置页）Insets 仍为正常值 703，不受影响。副作用是该场景下应用内容不做 resize（键盘悬浮于内容之上），对微信无可见负面影响。

```
@Override public void onComputeInsets(Insets out) {
    super.onComputeInsets(out);               // 原有逻辑保持
    EditorInfo ei = getCurrentInputEditorInfo();
    if (ei != null && "com.tencent.mm".equals(ei.packageName)
            && isPadDualPane()) {             // 建议增加平板/分栏判定
        out.contentTopInsets = decorViewHeight;   // 仅改动此项
    }
}
```

6.3 联想 / ZUI 系统侧

- 复核 ActivityEmbedding（平行视界）下 IME Insets 的坐标变换与分摊逻辑，尤其内嵌 Activity 的 taskFragmentBounds 与全屏 Insets 的换算；
- 与 AOSP 同场景行为做差异比对（本机 AOSP 输入法亦表现异常，指向系统分栏与 IME 的交互）；
- 可考虑在系统层为“微信 + 平板分栏 + IME”建立兼容白名单，作为过渡方案。

7 附录

7.1 关键数值汇总

指标	数值
屏幕分辨率	3040 × 1904 px @ 440 dpi
状态栏高度	107 px
导航 / 手势区	132 px（约 48 dp）
搜狗 IME 窗口 frame	[0,107] – [3040,1904]（高 1797）
contentTopInsets / visibleTopInsets	1284 / 1284
系统下发 ime inset	513 px
键盘实际顶部	1391 px
微信输入框错误落点	761 … 890 px
空白错位量	501 px ≈ 182 dp
非微信应用 ime inset	703 px（正常）

7.2 复现环境取得方式

```
adb shell dumpsys window | grep -E "InputMethod|InsetsSource|ChattingUI|Frames"
adb shell dumpsys input_method | grep -A2 "Last computed insets"
adb shell wm size; adb shell wm density
```

7.3 附件

- fig1-defect.png —— 问题现象（已脱敏）
- fig2-control.png —— 非微信应用正常对照（已脱敏）
- fig3-fixed.png —— 按 6.2 方案验证后的贴合效果（已脱敏）

本报告基于用户侧可复现的实测数据整理，未包含任何账号、会话、联系人等个人信息；所涉应用均为公开发行版本。