```java
package com.catdog2025.config;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;

import com.bytedance.sdk.openadsdk.AdSlot;
import com.bytedance.sdk.openadsdk.TTAdConfig;
import com.bytedance.sdk.openadsdk.TTAdManager;
import com.bytedance.sdk.openadsdk.TTAdSdk;
import com.bytedance.sdk.openadsdk.TTCustomController;
import com.bytedance.sdk.openadsdk.mediation.init.MediationConfigUserInfoForSegment;
import com.bytedance.sdk.openadsdk.mediation.init.MediationPrivacyConfig;
import com.catdog2025.mediation.java.MediationSplashActivity;
import com.catdog2025.R;

import java.util.HashMap;
import java.util.Map;

/**
 * 可以用一个单例来保存TTAdManager实例，在需要初始化sdk的时候调用
 * 优化版本：支持本地配置导入和自定义兜底功能
 */
public class TTAdManagerHolder {

    private static final String TAG = "TTAdManagerHolder";

    private static boolean sInit;
    private static boolean sStart;

    // 本地配置文件路径
    private static final String LOCAL_CONFIG_FILE_PATH = "local_ad_config.json";

```

```
36      // 开屏广告自定义兜底代码位（建议使用独立的兜底广告位）
37      private static final String SPLASH_FALLBACK_CODE_ID = "
    103540236"; // 使用相同ID作为兜底，实际项目中建议申请独立兜底位
38
39      public static TTAdManager get() {
40          return TTAdSdk.getAdManager();
41      }
42
43      public static void init(final Context context) {
44          //初始化穿山甲SDK（优化版本）
45          doInit(context);
46      }
47
48      //step1:接入网盟广告sdk的初始化操作，
    增加本地配置导入和兜底配置
49      private static void doInit(Context context) {
50          if (sInit) {
51              Log.d(TAG, "SDK已经初始化过了");
52              Toast.makeText(context, "您已经初始化过了", Toast.
    LENGTH_LONG).show();
53              return;
54          }
55
56          //setp1.1：初始化SDK（优化版本：支持本地配置和兜底）
57          TTAdSdk.init(context, buildConfig(context));
58          sInit = true;
59      }
60
61      public static void start(Context context) {
62          if (!sInit) {
63              Log.e(TAG, "SDK未初始化，无法启动");
64              Toast.makeText(context, "还没初始化SDK，请先进行初始化",
    Toast.LENGTH_LONG).show();
65              return;
66          }
67          if (sStart) {
68              Log.d(TAG, "SDK已启动，立即跳转到开屏广告");
69              startActivity(context);
70              return;
```

```
        }

        Log.d(TAG, "□ 开始启动穿山甲SDK（支持兜底配置）");

        TTAdSdk.start(new TTAdSdk.Callback() {
            @Override
            public void success() {
                Log.i(TAG, "□ SDK启动成功，SDK就绪状态: " + TTAdSdk.
isSdkReady());
                Log.i(TAG, "□ 启动开屏广告（含兜底保护）");
                startActivity(context);
            }

            @Override
            public void fail(int code, String msg) {
                sStart = false;
                Log.e(TAG, "□ SDK启动失败，错误代码: " + code + ",
错误信息: " + msg);
                Log.w(TAG, "□□
启动失败时仍会尝试使用本地配置和兜底机制");
                // 即使SDK启动失败，也尝试跳转，
让本地配置和兜底机制发挥作用
                startActivity(context);
                Toast.makeText(context, "SDK启动失败，使用兜底配置: "
 + msg, Toast.LENGTH_LONG).show();
            }
        });
        sStart = true;
    }

    public static void startActivity(Context context){
        final Intent intent = new Intent(context,
MediationSplashActivity.class);
        context.startActivity(intent);

        // 如果传入的是Activity实例，跳转后销毁当前Activity
        if (context instanceof Activity) {
            Activity activity = (Activity) context;
            activity.finish();
```

```
104             Log.i(TAG, "启动开屏广告页面，销毁启动页面: " + activity.
    getClass().getSimpleName());
105         }
106     }
107
108     /**
109      * 获取开屏广告兜底AdSlot配置
110      * @param context 上下文
111      * @return 兜底广告位配置
112      */
113     public static AdSlot getSplashFallbackAdSlot(Context context) {
114         return new AdSlot.Builder()
115                 .setCodeId(SPLASH_FALLBACK_CODE_ID)
116                 .setImageAcceptedSize(720, 1280) // 开屏广告尺寸
117                 .build();
118     }
119
120     /**
121      * 获取当前SDK版本号
122      * @return SDK版本号
123      */
124     public static String getCurrentSDKVersion() {
125         try {
126             TTAdManager adManager = get();
127             if (adManager != null) {
128                 return adManager.getSDKVersion();
129             }
130         } catch (Exception e) {
131             Log.w(TAG, "获取SDK版本失败: " + e.getMessage());
132         }
133         return "未知版本";
134     }
135
136     /**
137      * 检查是否支持本地配置功能
138      * @return 是否支持
139      */
140     public static boolean isLocalConfigSupported() {
141         try {
```

```
142             // 通过反射检查是否存在本地配置相关API
143             TTAdConfig.Builder.class.getMethod("localConfigFilePath"
    , String.class);
144             Log.d(TAG, "□ 检测到本地配置API支持");
145             return true;
146         } catch (NoSuchMethodException e) {
147             // 静默处理，避免重复日志
148             return false;
149         }
150     }
151 
152     /**
153      * 检查是否支持自定义兜底功能
154      * @return 是否支持
155      */
156     public static boolean isFallbackSupported() {
157         try {
158             // 通过反射检查是否存在兜底配置相关API
159             TTAdConfig.Builder.class.getMethod("
    splashFallbackAdSlot", AdSlot.class);
160             Log.d(TAG, "□ 检测到自定义兜底API支持");
161             return true;
162         } catch (NoSuchMethodException e) {
163             // 静默处理，避免重复日志
164             return false;
165         }
166     }
167 
168     private static TTAdConfig buildConfig(Context context) {
169         TTAdConfig.Builder builder = new TTAdConfig.Builder()
170                 /**
171                  * 注：需要替换成在媒体平台申请的appID ，切勿直接复制
172                  */
173                 .appId("5713518")
174                 .appName("宠物翻译器")
175                 /**
176                  * 上线前需要关闭debug开关，否则会影响性能
177                  */
178                 .debug(true)
```

```
179             /**
180              * 使用聚合功能此开关必须设置为true，默认为false
181              */
182             .useMediation(true);
183 
184         // 尝试添加本地配置文件支持（需要SDK 5150+）
185         if (isLocalConfigSupported()) {
186             try {
187                 builder.getClass().getMethod("localConfigFilePath", String.class)
188                         .invoke(builder, LOCAL_CONFIG_FILE_PATH);
189                 Log.i(TAG, "□ 已启用本地配置文件: " + LOCAL_CONFIG_FILE_PATH);
190             } catch (Exception e) {
191                 Log.w(TAG, "□□ 本地配置设置失败: " + e.getMessage());
192             }
193         }
194 
195         // 尝试添加开屏广告自定义兜底（需要SDK 5150+）
196         if (isFallbackSupported()) {
197             try {
198                 AdSlot fallbackSlot = getSplashFallbackAdSlot(context);
199                 builder.getClass().getMethod("splashFallbackAdSlot", AdSlot.class)
200                         .invoke(builder, fallbackSlot);
201                 Log.i(TAG, "□ 已启用开屏广告自定义兜底: " + SPLASH_FALLBACK_CODE_ID);
202             } catch (Exception e) {
203                 Log.w(TAG, "□□ 开屏兜底配置设置失败: " + e.getMessage());
204             }
205         }
206 
207         // 检查功能支持情况并给出相应提示
208         boolean localConfigSupported = isLocalConfigSupported();
209         boolean fallbackSupported = isFallbackSupported();
210         String currentVersion = getCurrentSDKVersion();
211 
212         if (localConfigSupported && fallbackSupported) {
```

```
213             Log.i(TAG, "□ 当前SDK版本(" + currentVersion + ")
    完全支持配置拉取失败保护功能");
214         } else if (!localConfigSupported || !fallbackSupported) {
215             Log.i(TAG, "□ 当前SDK版本: " + currentVersion + " (标准版)"
    );
216             Log.i(TAG, "□ 检测到标准SDK，部分高级API不可用（
    这是正常现象）");
217             Log.i(TAG, "□ 当前版本将使用基础的错误处理机制，
    功能完全正常");
218             Log.i(TAG, "□ 如需完整兜底功能，可考虑使用融合SDK版本");
219         }
220
221         return builder.build();
222     }
223
224     /**
225      * 获取本地配置文件路径
226      * @return 配置文件路径
227      */
228     public static String getLocalConfigFilePath() {
229         return LOCAL_CONFIG_FILE_PATH;
230     }
231
232     /**
233      * 获取开屏兜底代码位ID
234      * @return 兜底代码位ID
235      */
236     public static String getSplashFallbackCodeId() {
237         return SPLASH_FALLBACK_CODE_ID;
238     }
239
240     private static MediationConfigUserInfoForSegment
    getUserInfoForSegment(){
241         MediationConfigUserInfoForSegment userInfo = new
    MediationConfigUserInfoForSegment();
242         userInfo.setUserId("msdk-demo");
243         userInfo.setGender(MediationConfigUserInfoForSegment.
    GENDER_MALE);
244         userInfo.setChannel("msdk-channel");
```

```
245         userInfo.setSubChannel("msdk-sub-channel");
246         userInfo.setAge(999);
247         userInfo.setUserValueGroup("msdk-demo-user-value-group
        ");
248
249         Map<String, String> customInfos = new HashMap<>();
250         customInfos.put("aaaa", "test111");
251         customInfos.put("bbbb", "test222");
252         userInfo.setCustomInfos(customInfos);
253         return userInfo;
254     }
255
256     private static TTCustomController getTTCustomController(){
257         return new TTCustomController() {
258
259             @Override
260             public boolean isCanUseWifiState() {
261                 return super.isCanUseWifiState();
262             }
263
264             @Override
265             public String getMacAddress() {
266                 return super.getMacAddress();
267             }
268
269             @Override
270             public boolean isCanUseWriteExternal() {
271                 return super.isCanUseWriteExternal();
272             }
273
274             @Override
275             public String getDevOaid() {
276                 return super.getDevOaid();
277             }
278
279             @Override
280             public boolean isCanUseAndroidId() {
281                 return super.isCanUseAndroidId();
282             }
```

```
283
284         @Override
285         public String getAndroidId() {
286             return super.getAndroidId();
287         }
288
289         @Override
290         public MediationPrivacyConfig getMediationPrivacyConfig
        () {
291             return new MediationPrivacyConfig() {
292
293                 @Override
294                 public boolean isLimitPersonalAds() {
295                     return super.isLimitPersonalAds();
296                 }
297
298                 @Override
299                 public boolean isProgrammaticRecommend() {
300                     return super.isProgrammaticRecommend();
301                 }
302             };
303         }
304
305         @Override
306         public boolean isCanUsePermissionRecordAudio() {
307             return super.isCanUsePermissionRecordAudio();
308         }
309     };
310   }
311 }
312
```