

Unisoc Confidential For hiar

Android 10.0 联系人应用客制化指导手册

文档版本
发布日期

V1.0
2020-09-01

版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

Unisoc Confidential For hiar

紫光展锐（上海）科技有限公司



前言

概述

本文档提供联系人相关的客户定制说明，包含功能描述和如何客制化。

读者对象


本文档主要适用于 UNISOC 需要定制联系人相关功能的客户。

缩略语

缩略语	英文全名	中文解释
SNE	Second Name Entry	昵称
AAS	Additional number Alpha String	附加号码标签
SIM	Subscriber Identity Module	用户识别模块

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

变更信息

文档版本	发布日期	修改说明
V1.0	2020-09-01	第一次正式发布。

关键字

联系人、客制化。

Unisoc Confidential For hiar

目 录

1 去除默认联系人.....	1
1.1 功能说明	1
1.2 定制方式	1
1.3 验证方式	1
2 支持联系人电话号码片段搜索.....	2
2.1 功能说明	2
2.2 定制方式	2
2.3 验证方式	2
3 SIM 卡联系人支持 SNE 和 AAS	3
3.1 功能说明	3
3.2 定制方式	3
3.3 验证方式	3
4 禁止添加相同的号码.....	4
4.1 功能说明	4
4.2 定制方式	4
4.3 验证方式	8

Unisoc Confidential For hiar

1 去除默认联系人

1.1 功能说明

在 Contacts 中添加一条可定制的默认联系人，不能编辑和删除。

1.2 定制方式

Android 10.0 提供了控制默认联系人是否显示的开关，通过关闭开关，去除掉默认联系人，实现客户需求。去除默认联系人的开关路径：

/vendor/sprd/platform/packages/providers/SprdContactsProvider/res/values/config.xml

```
<bool name="is_support_default_contacts">true</bool>
```

只需将该值设置为 false，默认联系人即不再生效。默认为 true，内置展锐联系人。

1.3 验证方式

编译 SprdContactsProvider 模块，并 push 到路径：system_ext/priv-app/SprdContactsProvider 以后，删除联系人数据库并重启，查看 Contacts 是否还有默认联系人。

2 支持联系人电话号码片段搜索

2.1 功能说明

支持联系人电话号码片段搜索。

2.2 定制方式

Android 10.0 提供了控制是否支持片段搜索的开关，通过打开开关，可以实现片段搜索。开关路径：

/vendor/sprd/platform/packages/providers/SprdContactsProvider/res/values/config.xml

```
<bool name="is_support_segment_search">false</bool>
```

只需将该值设置为 true，即可支持联系人电话号码片段搜索。

2.3 验证方式

编译 SprdContactsProvider 模块，并 push 到路径：system_ext/priv-app/SprdContactsProvider 以后，查看是否可以支持片段搜索。

3

SIM 卡联系人支持 SNE 和 AAS

3.1 功能说明

Android 原生不支持 SIM 卡文件 SNE 和 AAS 读取，此功能开启之后支持读写上述两个 SIM 卡文件。

3.2 定制方式

Android 10.0 提供了控制是否支持 SNE 和 AAS 的开关，通过打开开关，可以实现对上述文件的读写。开关路径：

frameworks/base/core/res/res/values/config_ex.xml

```
<bool name="config_supportorangeCapable">false</bool>
```

只需将该值设置为 true，即可实现对 SNE 和 AAS 文件的读写。

3.3 验证方式

编译 framework-res 模块，并 push 到路径：system/framework/以后，插入支持 SNE 和 AAS 的 SIM 卡，确认是否可以对这两个文件进行读写。

4 禁止添加相同的号码

4.1 功能说明

默认情况下不同的联系人可以保存两个或者多个相同的电话号码，此功能要求禁止添加相同的号码。

4.2 定制方式

代码仓库修改路径：

/vendor/sprd/platform/packages/apps/SprdContacts

修改文件 1

```
diff --git a/res/values-zh-rCN/strings.xml b/res/values-zh-rCN/strings.xml
```

```
index 08a5f23..b53f96d 100644
```

```
--- a/res/values-zh-rCN/strings.xml
```

```
+++ b/res/values-zh-rCN/strings.xml
```

```
@@ -740,6 +740,7 @@
```

```
<!--SPRD: Bug 784388 add warn location express when enter into contacts-->
```

```
<string name="warn_location_title">警示</string>
```

```
<string name="warn_location_message">是否允许开启定位服务?</string>
```

```
+ <string name="warn_dup_number_message">该电话号码已经存在</string>
```

```
<!--SPRD: Bug 869541 add import note-->
```

```
<string name="importing_vcard_not_valid"><xliff:g id="filename" example="import.vcf">%s</xliff:g> 文件无效，导入失败。</string>
```

修改文件 2

```
diff --git a/res/values/strings.xml b/res/values/strings.xml
```

```
index de1305f..4e7f2a5 100644
```

```
--- a/res/values/strings.xml
```

```
+++ b/res/values/strings.xml
```

```
@@ -1811,6 +1811,7 @@
```

```
<!--SPRD: Bug 784388 add warn location express when enter into contacts-->
```

```
<string name="warn_location_title">Warning</string>
```

```
<string name="warn_location_message">Whether to allow to turn on the location service?</string>
```

```
+ <string name="warn_dup_number_message">This phone number already exists</string>
```

```
<!--SPRD: Bug 869541 add import note-->
```

```
<string name="importing_vcard_not_valid"><xliff:g id="filename" example="import.vcf">%s</xliff:g> is not valid,importing  
has failed.</string>
```

修改文件 3

```
diff --git a/src/com/android/contacts/editor/ContactEditorFragment.java  
b/src/com/android/contacts/editor/ContactEditorFragment.java  
index 4a1ff99..4157dda 100644  
--- a/src/com/android/contacts/editor/ContactEditorFragment.java  
+++ b/src/com/android/contacts/editor/ContactEditorFragment.java  
@@ -874,7 +874,16 @@ public class ContactEditorFragment extends Fragment implements  
    Toast.makeText(getActivity(), R.string.toast_batchoperation_is_running,  
        Toast.LENGTH_LONG).show();  
    } else {  
+        if (null != mAction && !"".equals(mAction)) {  
+            if (mAction.equals(Intent.ACTION_EDIT)) {  
+                return save(SaveMode.EDITOR);  
+            } else if (mAction.equals(Intent.ACTION_INSERT)) {  
+                return save(SaveMode.CLOSE);  
+            }  
+        }  
+  
        return save(SaveMode.CLOSE);  
+  
    }  
+  
    /**  
    * @}  
@@ -985,6 +994,15 @@ public class ContactEditorFragment extends Fragment implements  
    }  
    mStatus = Status.SAVING;  
  
+    if (saveMode == SaveMode.CLOSE || saveMode == SaveMode.EDITOR) {  
+        if (AccountRestrictionUtils.get(mContext).violateDupNumber(mState, saveMode)) {  
+            int toastId = R.string.warn_dup_number_message;  
+            Toast.makeText(mContext, toastId, Toast.LENGTH_LONG).show();  
+            mStatus = Status.EDITING;  
+            return true;  
+        }  
+    }  
+  
    if (!hasPendingChanges()) {  
        if (mLookupUri == null && saveMode == SaveMode.RELOAD) {  
            // We don't have anything to save and there isn't even an existing contact yet.
```

修改文件 4

```
diff --git a/src/com/sprd/contacts/util/AccountRestrictionUtils.java b/src/com/sprd/contacts/util/AccountRestrictionUtils.java
index e667cae..b4aaace 100644
```

```
--- a/src/com/sprd/contacts/util/AccountRestrictionUtils.java
+++ b/src/com/sprd/contacts/util/AccountRestrictionUtils.java
@@ -18,6 +18,7 @@ package com.sprd.contacts.util;
```

```
import android.text.TextUtils;
import com.android.contacts.R;
+import com.android.contacts.activities.ContactEditorActivity.ContactEditor.SaveMode;
import com.android.contacts.model.AccountTypeManager;
import com.android.contacts.model.RawContactDelta;
import com.android.contacts.model.RawContactDeltaList;
@@ -76,6 +77,11 @@ import android.provider.Telephony.Mms;
```

```
import java.util.HashMap;
```

```
+import android.net.Uri;
+import android.provider.ContactsContract.PhoneLookup;
+import android.provider.ContactsContract;
+import android.telephony.PhoneNumberUtils;
```

```
+
public class AccountRestrictionUtils {
    private static AccountRestrictionUtils sInstance;
    private AccountManager mAm;
@@ -350,6 +356,72 @@ public class AccountRestrictionUtils {
    return ret;
}
```

```
+ public boolean violateDupNumber(RawContactDeltaList set, int saveMode) {
+     if (set == null) {
+         return false;
+     }
+     String mimeType = null;
+
+     for (RawContactDelta state : set) {
+         ArrayList<ContentValues> ret = state.getContentValues();
+         if (ret == null) {
+             continue;
+         }
+     }
+ }
```

```
+         for (ContentValues cv : ret) {
+             mimeType = cv.getAsString(Data.MIMETYPE);
+             if (Phone.CONTENT_ITEM_TYPE.equals(mimeType)
+                 && cv.getAsString(Data.DATA1) != null
+                 && !cv.getAsString(Data.DATA1).equals("")) {
+                 return exitsContactsNumber(cv.getAsString(Data.DATA1), state.getRawContactId());
+             }
+         }
+     }
+     return false;
+ }
+
+ private boolean existsNormalizeNumber(String number, long contactID) {
+     Uri uri = Uri.withAppendedPath(PhoneLookup.CONTENT_FILTER_URI, Uri.encode(number));
+     ContentResolver cr = mContext.getContentResolver();
+     Cursor cursor = cr.query(uri, new String[] {
+         "number", ContactsContract.CommonDataKinds.Phone.CONTACT_ID,
+         null, null, null);
+     long id = -1;
+     if (cursor.moveToNext()) {
+         id = cursor.getLong(1);
+         if (id != contactID) {
+             return true;
+         }
+     }
+
+     if (cursor != null) {
+         cursor.close();
+     }
+
+     return false;
+ }
+
+ private boolean exitsContactsNumber(String number, long rawContactID) {
+     Uri uri = ContactsContract.Data.CONTENT_URI;
```

```
+    if (null == number || "".equals(number)) {  
+        return false;  
+    }  
+    number = number.trim();  
+    number = PhoneNumberUtils.normalizeNumber(number);  
+    if (existsNormalizeNumber(number, rawContactID)) {  
+        return true;  
+    }  
+  
+    return false;  
+  
+ }  
+  
+  
+ public static byte[] getGsmAlphabetBytes(String record) {  
+     byte[] bytes = new byte[0];  
+     if (record == null) {
```

4.3 验证方式

编译 SprdContacts 模块，并 push 到路径：system_ext/priv-app/SprdContacts 以后，保存或者编辑两个一样的号码，查看是否能保存成功。

Unisoc Confidential For hiar