Android 5.1
SIM 卡不识别问题修改指导

发布日期　　2015-09-06

注意

本手册描述的产品及其附件的某些特性和功能，取决于当地网络的设计和性能，以及您安装的软件。某些特性和功能可能由于当地网络运营商或网络服务供应商不支持，或者由于当地网络的设置，或者您安装的软件不支持而无法实现。因此，本手册中的描述可能与您购买的产品或其附件并非完全一一对应。

华为技术有限公司保留随时修改本手册中任何信息的权利，无需提前通知且不承担任何责任。

责任限制

本手册中的内容均“按照现状”提供，除非适用法要求，华为技术有限公司对本手册中的所有内容不提供任何明示或暗示的保证，包括但不限于适销性或者适用于某一特定目的的保证。

在适用法律允许的范围内，华为技术有限公司在任何情况下，都不对因使用本手册相关内容及本手册描述的产品而产生的任何特殊的、附带的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉或预期节约的损失进行赔偿。

在相关法律允许的范围内，在任何情况下，华为技术有限公司对您因为使用本手册描述的产品而遭受的损失的最大责任（除在涉及人身伤害的情况中根据适用的法律规定的损害赔偿外）以您购买本产品所支付的价款为限。

进出口管制

若需将本手册描述的产品（包括但不限于产品中的软件及技术数据等）出口、再出口或者进口，您应遵守适用的进出口管制法律法规。

隐私保护

为了解我们如何保护您的个人信息，请访问 http://consumer.huawei.com/privacy-policy 阅读我们的隐私政策。

产品名称Product name	密级Confidentiality level
Android系列	对外发布
产品版本Product version	Total pages 共 页

Prepared by 拟制	Module Android Team	Date 日期	2015-09-06
Reviewed by 评审人		Date 日期	
Approved by 批准		Date 日期	

Huawei Technologies Co., Ltd.

华为技术有限公司

（DVP05T03 V2.41/ IPD-CMM V3.0）

（DVP05T03 V2.41/ IPD-CMM V3.0）

Android 5.1原生系统存在SIM卡不识别问题，这是由于上层读取ICCID（通过SIM_IO接口）返回值解析错误导致；需要修改framework层逻辑。

一、修改IccFileHandler.java文件

（frameworks\opt\telephony\src\java\com\android\internal\telephony\uicc\IccFileHandler.java），如下：

修改public void handleMessage(Message msg)函数（红色部分）：

➢ 修改EVENT_GET_EF_LINEAR_RECORD_SIZE_DONE事件：

```
case EVENT_GET_EF_LINEAR_RECORD_SIZE_DONE:
    ar = (AsyncResult)msg.obj;
    lc = (LoadLinearFixedContext) ar.userObj;
    result = (IccIoResult) ar.result;
    response = lc.mOnLoaded;

    if (processException(response, (AsyncResult) msg.obj)) {
        break;
    }

    data = result.payload;

    if (UiccTlvData.isUiccTlvData(data)) {

        UiccTlvData tlvData = UiccTlvData.parse(data);

        if (tlvData.isIncomplete()) {
            throw new IccFileTypeMismatch();
        }

        recordSize = new int[3];
        recordSize[0] = tlvData.mRecordSize;
        recordSize[1] = tlvData.mFileSize;
        recordSize[2] = tlvData.mNumRecords;

    } else if (TYPE_EF == data[RESPONSE_DATA_FILE_TYPE] &&
        EF_TYPE_LINEAR_FIXED == data[RESPONSE_DATA_STRUCTURE]) {

        recordSize = new int[3];
        recordSize[0] = data[RESPONSE_DATA_RECORD_LENGTH] & 0xFF;
        recordSize[1] = ((data[RESPONSE_DATA_FILE_SIZE_1] & 0xff) << 8)
                + (data[RESPONSE_DATA_FILE_SIZE_2] & 0xff);
        recordSize[2] = recordSize[1] / recordSize[0];
```

```
        } else {
            throw new IccFileTypeMismatch();
        }

        /*recordSize = new int[3];
        recordSize[0] = data[RESPONSE_DATA_RECORD_LENGTH] & 0xFF;
        recordSize[1] = ((data[RESPONSE_DATA_FILE_SIZE_1] & 0xff) << 8)
                + (data[RESPONSE_DATA_FILE_SIZE_2] & 0xff);
        recordSize[2] = recordSize[1] / recordSize[0];*/

        sendResult(response, recordSize, null);
    break;
```

- 修改EVENT_GET_RECORD_SIZE_DONE事件：

```
    case EVENT_GET_RECORD_SIZE_DONE:
        ar = (AsyncResult)msg.obj;
        lc = (LoadLinearFixedContext) ar.userObj;
        result = (IccIoResult) ar.result;
        response = lc.mOnLoaded;

        if (processException(response, (AsyncResult) msg.obj)) {
            break;
        }

        data = result.payload;

        if (UiccTlvData.isUiccTlvData(data)) {

            UiccTlvData tlvData = UiccTlvData.parse(data);

            if (tlvData.isIncomplete()) {
                throw new IccFileTypeMismatch();
            }

            lc.mRecordSize = tlvData.mRecordSize;
            lc.mCountRecords = tlvData.mNumRecords;
            size = tlvData.mFileSize;

        } else if (TYPE_EF == data[RESPONSE_DATA_FILE_TYPE]) {

            if (EF_TYPE_LINEAR_FIXED != data[RESPONSE_DATA_STRUCTURE]) {
                throw new IccFileTypeMismatch();
            }
```

```
        lc.mRecordSize = data[RESPONSE_DATA_RECORD_LENGTH] & 0xFF;

        size = ((data[RESPONSE_DATA_FILE_SIZE_1] & 0xff) << 8)
                + (data[RESPONSE_DATA_FILE_SIZE_2] & 0xff);

        lc.mCountRecords = size / lc.mRecordSize;
    } else {
        throw new IccFileTypeMismatch();
    }

    /*if (EF_TYPE_LINEAR_FIXED != data[RESPONSE_DATA_STRUCTURE]) {
        throw new IccFileTypeMismatch();
    }

    lc.mRecordSize = data[RESPONSE_DATA_RECORD_LENGTH] & 0xFF;

    size = ((data[RESPONSE_DATA_FILE_SIZE_1] & 0xff) << 8)
            + (data[RESPONSE_DATA_FILE_SIZE_2] & 0xff);

    lc.mCountRecords = size / lc.mRecordSize;*/

    if (lc.mLoadAll) {
        lc.results = new ArrayList<byte[]>(lc.mCountRecords);
    }

    mCi.iccIOForApp(COMMAND_READ_RECORD, lc.mEfid, getEFPath(lc.mEfid),
            lc.mRecordNum,
            READ_RECORD_MODE_ABSOLUTE,
            lc.mRecordSize, null, null, mAid,
            obtainMessage(EVENT_READ_RECORD_DONE, lc));
    break;
```

- 修改EVENT_GET_BINARY_SIZE_DONE事件：

```
    case EVENT_GET_BINARY_SIZE_DONE:
        ar = (AsyncResult)msg.obj;
        response = (Message) ar.userObj;
        result = (IccIoResult) ar.result;

        if (processException(response, (AsyncResult) msg.obj)) {
            break;
        }

        data = result.payload;
```

```
fileid = msg.arg1;

if (UiccTlvData.isUiccTlvData(data)) {


    UiccTlvData tlvData = UiccTlvData.parse(data);

    if (tlvData.mFileSize < 0) {
        throw new IccFileTypeMismatch();
    }

    size = tlvData.mFileSize;


} else if (TYPE_EF == data[RESPONSE_DATA_FILE_TYPE]) {

    if (EF_TYPE_TRANSPARENT != data[RESPONSE_DATA_STRUCTURE]) {
        throw new IccFileTypeMismatch();
    }

    size = ((data[RESPONSE_DATA_FILE_SIZE_1] & 0xff) << 8)
            + (data[RESPONSE_DATA_FILE_SIZE_2] & 0xff);
} else {
    throw new IccFileTypeMismatch();
}

/*if (EF_TYPE_TRANSPARENT != data[RESPONSE_DATA_STRUCTURE]) {
    throw new IccFileTypeMismatch();
}

size = ((data[RESPONSE_DATA_FILE_SIZE_1] & 0xff) << 8)
        + (data[RESPONSE_DATA_FILE_SIZE_2] & 0xff);*/

mCi.iccIOForApp(COMMAND_READ_BINARY, fileid, getEFPath(fileid),
                0, 0, size, null, null, mAid,
                obtainMessage(EVENT_READ_BINARY_DONE,
                        fileid, 0, response));
break;
```

二、增加UiccTlvData.java协议解析文件

(frameworks\opt\telephony\src\java\com\android\internal\telephony\uicc\UiccTlvData.java)

UiccTlvData.java 如下（红色部分代码）：

```
package com.android.internal.telephony.uicc;

/**
 * UICC TLV Data Parser according to ETSI TS 102 221 spec.
 */
public class UiccTlvData {

    private static final int TLV_FORMAT_ID = 0x62;

    private static final int TAG_FILE_DESCRIPTOR = 0x82;
    private static final int TAG_FILE_IDENTIFIER = 0x83;
    private static final int TAG_PROPRIETARY_INFO = 0xA5;
    private static final int TAG_LIFECYCLE_STATUS = 0x8A;
    private static final int TAG_SECURITY_ATTR_1 = 0x8B;
    private static final int TAG_SECURITY_ATTR_2 = 0x8C;
    private static final int TAG_SECURITY_ATTR_3 = 0xAB;
    private static final int TAG_FILE_SIZE = 0x80;
    private static final int TAG_TOTAL_FILE_SIZE = 0x81;
    private static final int TAG_SHORT_FILE_IDENTIFIER = 0x88;

    private static final int TYPE_5 = 5;
    private static final int TYPE_2 = 2;

    int mRecordSize;
    int mFileSize;
    int mNumRecords;
    boolean mIsDataEnough;

    private int mFileType = -1;

    private UiccTlvData() {
        mNumRecords = -1;
        mFileSize = -1;
        mRecordSize = -1;
    }

    public boolean isIncomplete() {
        return mNumRecords == -1 || mFileSize == -1 || mRecordSize == -1 || mFileType == -1;
    }

    public static boolean isUiccTlvData(byte[] data) {
        if(data != null && data.length > 0 && TLV_FORMAT_ID == (data[0] & 0xFF)) {
            return true;
        }
```

```
        return false;
    }

    public static UiccTlvData parse(byte[] data) throws IccFileTypeMismatch{

        UiccTlvData parsedData = new UiccTlvData();

        if (data == null || data.length == 0 || TLV_FORMAT_ID != (data[0] & 0xFF)) {
            throw new IccFileTypeMismatch();
        }

        try {

            int currentLocation = 2; //Ignore FCP size
            int currentTag;

            while (currentLocation < data.length) {
                currentTag = data[currentLocation++] & 0xFF;

                switch (currentTag) {
                    case TAG_FILE_DESCRIPTOR:
                        currentLocation = parsedData.parseFileDescriptor(data, currentLocation);
                        break;

                    case TAG_FILE_SIZE:
                        currentLocation = parsedData.parseFileSize(data, currentLocation);
                        break;

                    case TAG_FILE_IDENTIFIER:
                    case TAG_PROPRIETARY_INFO:
                    case TAG_LIFECYCLE_STATUS:
                    case TAG_SECURITY_ATTR_1:
                    case TAG_SECURITY_ATTR_2:
                    case TAG_SECURITY_ATTR_3:
                    case TAG_TOTAL_FILE_SIZE:
                    case TAG_SHORT_FILE_IDENTIFIER:
                        currentLocation = parsedData.parseSomeTag(data, currentLocation);
                        break;

                    default:
                        //Unknown TAG
                        throw new IccFileTypeMismatch();

                }
```

```
            }

        } catch (ArrayIndexOutOfBoundsException e) {

            //We might be looking at incomplete data but we might have what we need.
            //Ignore this    and let caller handle it by checking isIncomplete
        }

        return parsedData;
    }

    private int parseFileSize(byte[] data, int currentLocation) {
        int length = data[currentLocation++] & 0xFF;

        int fileSize = 0;
        for (int i = 0; i < length; i++) {
            fileSize += ((data[currentLocation + i] & 0xFF) << ( 8 * (length - i - 1)));
        }

        mFileSize = fileSize;

        if (mFileType == TYPE_2) {
            mRecordSize = fileSize;
        }

        return currentLocation + length;
    }

    private int parseSomeTag(byte[] data, int currentLocation) {
        //Just skip unwanted tags;
        int length = data[currentLocation++] & 0xFF;
        return currentLocation + length;
    }

    private int parseFileDescriptor(byte[] data, int currentLocation) throws IccFileTypeMismatch {
        int length = data[currentLocation++] & 0xFF;
        if (length == 5) {

            mRecordSize = ((data[currentLocation + 2] & 0xFF) << 8) +
                        (data[currentLocation + 3] & 0xFF); // Length of 1 record
            mNumRecords = data[currentLocation + 4] & 0xFF; // Number of records

            mFileSize = mRecordSize * mNumRecords;
```

```
                mFileType = TYPE_5;

                return currentLocation + 5;
            } else if (length == 2) {

                int descriptorByte = data[currentLocation + 1] & 0xFF;

                //Ignore descriptorByte for now

                mNumRecords = 1;

                mFileType = TYPE_2;

                return currentLocation + 2;

            } else {
                throw new IccFileTypeMismatch();
            }
        }

    }
```