2016/6/6，SWD2，Telecom，方胜龙

第一部分：Android运营商名称显示之PLMN的读取

Plmn的全称是Public Land Mobile Network（公共陆地移动网络），而在运营商显示方面主要是指当前SIM所驻留的网络，比如当中国移动的SIM（46000）如果漫游到联通的网络（46001），那么虽然当前的SIM是中国移动，但是他的Plmn就应该是中国联通。也就是说，Plmn的名称与当前驻留的网络相关。那么Plmn的来源是什么呢？

一、AP侧Plmn的读取

在GsmServiceStateTracker中当检测到Radio或者Network状态发生改变时，就会从Modem中读取当前的一些网络状态，其中就包括Plmn的值：

```
1.  //GsmServiceStateTracker.java
2.      /**
3.       * A complete "service state" from our perspective is
4.       * composed of a handful of separate requests to the radio.
5.       *
6.       * We make all of these requests at once, but then abandon them
7.       * and start over again if the radio notifies us that some
8.       * event has changed
9.       */
10.     @Override
11.     public void pollState() {
12.         mPollingContext = new int[1];
13.         mPollingContext[0] = 0;
14.
15.         switch (mCi.getRadioState()) {
16.             case RADIO_UNAVAILABLE:
17.                 mNewSS.setStateOutOfService();
18.                 mNewCellLoc.setStateInvalid();
19.                 setSignalStrengthDefaultValues();
20.                 mGotCountryCode = false;
21.                 mNitzUpdatedTime = false;
22.                 pollStateDone();
23.             break;
24.
25.             case RADIO_OFF:
26.                 mNewSS.setStateOff();
27.                 mNewCellLoc.setStateInvalid();
28.                 setSignalStrengthDefaultValues();
29.                 mGotCountryCode = false;
30.                 mNitzUpdatedTime = false;
31.                 if (ServiceState.RIL_RADIO_TECHNOLOGY_IWLAN
32.                         != mSS.getRilDataRadioTechnology()) {
33.                     pollStateDone();
34.                 }
```

```
35.
36.             default:
37.                 // Issue all poll-related commands at once
38.                 // then count down the responses, which
39.                 // are allowed to arrive out-of-order
40.
41.                 mPollingContext[0]++;
42.                 mCi.getOperator(
43.                     obtainMessage(EVENT_POLL_STATE_OPERATOR, mPollingContext));
44.                 break;
45.             }
46.         }
```

这里的**mCi**就是RILJ对象：

```
1.  //RIL.java
2.      public void getOperator(Message result) {
3.          RILRequest rr = RILRequest.obtain(RIL_REQUEST_OPERATOR, result);
4.          if (RILJ_LOGD) riljLog(rr.serialString() + "> " + requestToString(rr.mRequ
    est));
5.          send(rr);
6.      }
```

当接收到RIL的返回数据后，就会接收到EVENT_POLL_STATE_OPERATOR的消息：

```
1.  //GsmServiceStateTracker.java
2.      public void handleMessage (Message msg) {
3.          AsyncResult ar;
4.          int[] ints;
5.          String[] strings;
6.          Message message;
7.          switch (msg.what) {
8.              case EVENT_POLL_STATE_REGISTRATION:
9.              case EVENT_POLL_STATE_GPRS:
10.             case EVENT_POLL_STATE_OPERATOR:
11.             case EVENT_POLL_STATE_NETWORK_SELECTION_MODE:
12.                  ar = (AsyncResult) msg.obj;
13.                  handlePollStateResult(msg.what, ar);
14.                  break;
15.             ......
16.         }
17.      }
```

然后在handlePollStateResult()中处理该消息：

```
1.  //GsmServiceStateTracker.java
2.      protected void handlePollStateResult (int what, AsyncResult ar) {
3.          int ints[];
4.          String states[];
5.          switch (what) {
6.                  case EVENT_POLL_STATE_OPERATOR: {
7.                      String opNames[] = (String[])ar.result;
8.                      if (opNames != null && opNames.length >= 3) {
```

```
                        // FIXME: Giving brandOverride higher precedence, is this
desired?
                        String brandOverride = mUiccController.getUiccCard(getPhon
eId()) != null ?
                            mUiccController.getUiccCard(getPhoneId()).getOperatorB
randOverride() : null;
                        if (brandOverride != null) {
                            log("EVENT_POLL_STATE_OPERATOR: use brandOverride=" +
brandOverride);
                            mNewSS.setOperatorName(brandOverride, brandOverride, o
pNames[2]);
                        } else {
                            mNewSS.setOperatorName (opNames[0], opNames[1], opName
s[2]);
                        }
                    }
                    break;
                }
```

这里看到，mNewSS(也就是ServiceState)可以从两个途径获取Plmn：

1、经过setOperatorBrandOverride设置过的Plmn，该方法目前尚未使用。

2、从Modem获取的Plmn，我们主要介绍这种常规的方法。

上面是通过setOperatorName()方法来将Plmn传输给mNewSS，也就是ServiceState对象：

```
//ServiceState.java
    public void setOperatorName(String longName, String shortName, String numeric)
 {
        mVoiceOperatorAlphaLong = longName;
        mVoiceOperatorAlphaShort = shortName;
        mVoiceOperatorNumeric = numeric;
        mDataOperatorAlphaLong = longName;
        mDataOperatorAlphaShort = shortName;
        mDataOperatorNumeric = numeric;
    }
```

由此，便将当前的Plmn Name保存在ServiceState对象中，其他对象就可以通过ServiceState的getOperatorAlphaLong()方法得到当前的Plmn值：

```
//ServiceState.java
    /**
     * Get current registered operator name in long alphanumeric format.
     *
     * In GSM/UMTS, long format can be up to 16 characters long.
     * In CDMA, returns the ERI text, if set. Otherwise, returns the ONS.
     *
     * @return long name of operator, null if unregistered or unknown
     */
    public String getOperatorAlphaLong() {
        return mVoiceOperatorAlphaLong;
    }
```

二、Modem侧Plmn的读取

刚才我们看到AP侧获取的Plmn实际上是通过向RIL发送了RIL_REQUEST_OPERATOR的请求，那么Modem侧如何获取Plmn的名称呢？

由于不同厂商使用的Modem不同，因此实现的方法也不同，但是根据3Gpp协议，该Plmn的取值可能来自以下几个地方(根据优先级排序)：

1、**Eons(Enhanced Operator Name String)**，也就是从SIM的EF_OPL和EF_PNN分区来读取Plmn Name

EF_OPL中存放的是LAC和EF_PNN中的Record Identifier

EF_PNN中存放的是Network Name，也就是具体的Plmn Name

如果注册上的网络是HPLMN，那么EF_OPL返回的Record Identifier就是1。

如果不是HPLMN的话，就根据LAC在EF_OPL中寻找对应的Record Identifier，然后根据OPL的Record Identifier，在PNN中找对应的Network Name。

需要注意的是，Record Identifier是基于1的，而EF_PNN的记录是基于0的。也就是说，Record Identifier是1，那匹配的是EF_PNN中的第0条记录。

2、**CPHS ONS(Common PCN Handset Specification Operator Name String)**，该字串也是保存在SIM文件系统中

该取值要求当前手机注册到HPLMN网络，此时Modem将会先读取SIM中的CPHS ONS的长格式文件(6F14)，如果存在，则将其作为Plmn上报，否则的话读取短格式文件(6F18)，如果存在，则将其作为Plmn上报。

3、**NITZ Operator Name**

该名称是由当前注册的网络下发给手机的，如果该值存在，就会将该值作为Plmn 那么上报给AP。

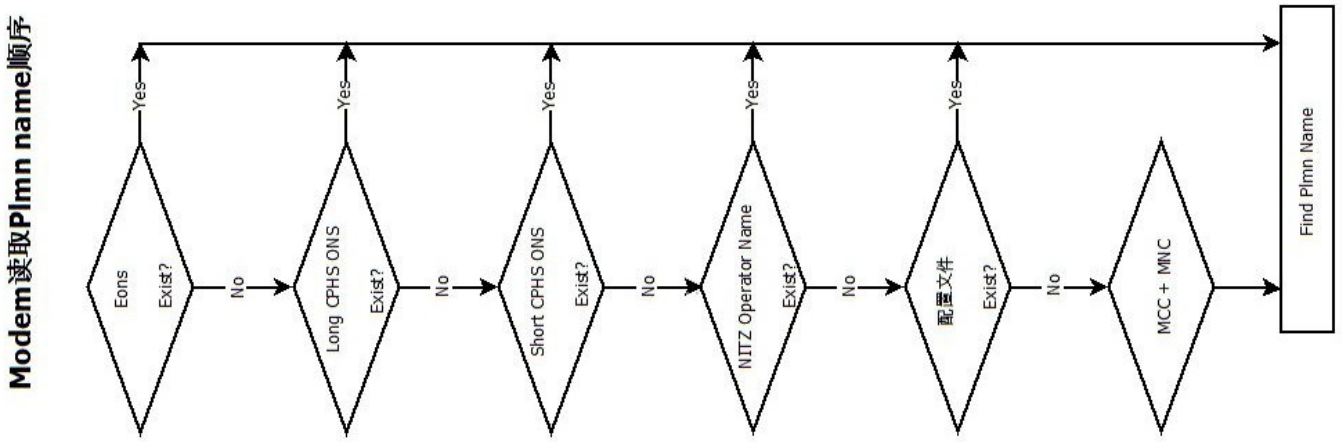
4、**配置文件读取**

如果以上的几种途径都没有获取到当前的Plmn Name，那么平台自身会提供从手机内存中根据当前注册的MCC MNC读取相应的Plmn Name，一般都是一个类似于apns-conf.xml的文件，在开机的时候被加载，这个方法在每个平台中也会不同。

5、**利用MCCMNC作为Plmn Name**

如果连ROM都没有找到当前Plmn对应的Name，那么就会把当前注册的Plmn所对应的MCC、MNC数字作为当前的Plmn Name

以上的读取流程可以用下图来表示



第二部分：Android运营商名称显示之SPN的读取

SPN(Service Provider Name)就是当前发行SIM卡的运营商的名称，可以从以下两个路径获取：

1、从SIM文件系统读取

2、从配置文件读取

一、从SIM读取SPN过程

一般来说，SIM上保存有当前SIM的发行运营商名称，也就是SPN，该字串可以存储在SIM的**EF_SPN(0x6F46)、EF_SPN_CPHS(0x6f14)、EF_SPN_SHORT_CPHS(0x6f18)**三个地址上，在SIMRecords初始化时通过getSpnFsm()从SIM中读取出来并保存。下面来看读取SPN的过程：

```
1.  //SIMRecords
2.  protected void fetchSimRecords() {
3.      getSpnFsm(true, null);
4.  }
```

请注意，此时的getSpnFsm()的start参数为true，而且mSpnState为初始化值：GetSpnFsmState.IDLE

```
1.  //SIMRecords
2.  private void getSpnFsm(boolean start, AsyncResult ar) {
3.      byte[] data;
4.      if (start) {
5.          if(mSpnState == GetSpnFsmState.READ_SPN_3GPP ||
6.                  mSpnState == GetSpnFsmState.READ_SPN_CPHS ||
7.                  mSpnState == GetSpnFsmState.READ_SPN_SHORT_CPHS ||
8.                  mSpnState == GetSpnFsmState.INIT) {
9.              mSpnState = GetSpnFsmState.INIT;
```

```
            return;
        } else {
            //mSpnState默认为IDLE，然后修改为INIT
            mSpnState = GetSpnFsmState.INIT;
        }
    }
    switch(mSpnState){
        case INIT:
            //初始化SPN
            setServiceProviderName(null);
            //从SIM的EF_SPN读取SPN
            mFh.loadEFTransparent(EF_SPN, obtainMessage(EVENT_GET_SPN_DONE));
            mRecordsToLoad++;
            //mSpnState修改为READ_SPN_3GPP
            mSpnState = GetSpnFsmState.READ_SPN_3GPP;
            break;
        case READ_SPN_3GPP:
        case READ_SPN_CPHS:
        case READ_SPN_SHORT_CPHS:
        default:
            mSpnState = GetSpnFsmState.IDLE;
    }
}
```

在上面的过程中，将会从EF_SPN中读取当前SPN，并且将mSpnState置为READ_SPN_3GPP。当读取完毕后，在handleMessage()中读取反馈：

```
public void handleMessage(Message msg) {
    try {
        switch (msg.what) {
            case EVENT_GET_SPN_DONE:
                isRecordLoadResponse = true;
                ar = (AsyncResult) msg.obj;
                getSpnFsm(false, ar);
                break;
        }
    } catch (RuntimeException exc) {
    } finally {
        if (isRecordLoadResponse) {
            onRecordLoaded();
        }
    }
}
```

然后再次进入getSpnFsm()中处理，此时的mSpnState状态为READ_SPN_3GPP，而start为false，所以直接进入switch语句判断：

```
private void getSpnFsm(boolean start, AsyncResult ar) {
    byte[] data;
    if (start) {
    }

```

```
6.
7.      switch(mSpnState){
8.          case INIT:
9.              break;
10.         case READ_SPN_3GPP:
11.             if (ar != null && ar.exception == null) {
12.                 data = (byte[]) ar.result;
13.                 //设置mSpnDisplayCondition显示标志位，如果通过EF_SPN没有取到，则认为mSpnDisplayCondition=-1
14.                 mSpnDisplayCondition = 0xff & data[0];
15.                 setServiceProviderName(IccUtils.adnStringFieldToString( data, 1, data.length - 1));
16.                 //将当前的SPN写入系统属性
17.                 setSystemProperty(PROPERTY_ICC_OPERATOR_ALPHA, getServiceProviderName());
18.                 mSpnState = GetSpnFsmState.IDLE;
19.             } else {
20.                 mFh.loadEFTransparent( EF_SPN_CPHS, obtainMessage(EVENT_GET_SPN_DONE));
21.                 mRecordsToLoad++;
22.                 mSpnState = GetSpnFsmState.READ_SPN_CPHS;
23.                 mSpnDisplayCondition = -1;
24.             }
25.             break;
26.         case READ_SPN_CPHS:
27.         case READ_SPN_SHORT_CPHS:
28.         default:
29.             mSpnState = GetSpnFsmState.IDLE;
30.     }
31. }
```

如果此时从SIM读取的SPN不为空，则会通过adnStringFieldToString()将数据转换为字串后，**通过setServiceProviderName()保存**，同时也要存储在PROPERTY_ICC_OPERATOR_ALPHA的系统属性中，并且重置mSpnState为IDLE；

另外，这里的mSpnDisplayCondition是SPN的第一位数据，在显示SPN时用来判定显示规则。

如果SIM中的SPN为空，则再去读取SIM中的EF_SPN_CPHS分区，和上面流程相同，该请求会通过handleMessage()再次发送给getSpnFsm()内部，只不过这次进入READ_SPN_CPHS分支处理：

```
1.  private void getSpnFsm(boolean start, AsyncResult ar) {
2.      byte[] data;
3.      if (start) {
4.      }
5.
6.
7.      switch(mSpnState){
8.          case INIT:
9.              break;
10.         case READ_SPN_3GPP:
11.             break;
```

```
        case READ_SPN_CPHS:
            if (ar != null && ar.exception == null) {
                data = (byte[]) ar.result;
                setServiceProviderName(IccUtils.adnStringFieldToString(data, 0, da
ta.length));
                setSystemProperty(PROPERTY_ICC_OPERATOR_ALPHA, getServiceProviderN
ame());
                mSpnState = GetSpnFsmState.IDLE;
            } else {
                mFh.loadEFTransparent( EF_SPN_SHORT_CPHS, obtainMessage(EVENT_GET_
SPN_DONE));
                mRecordsToLoad++;
                mSpnState = GetSpnFsmState.READ_SPN_SHORT_CPHS;
            }
            break;
        case READ_SPN_SHORT_CPHS:
        default:
            mSpnState = GetSpnFsmState.IDLE;
    }
}
```

与上面读取EF_SPN类似，如果读取成功就保存，否则再去读取EF_SPN_SHORT_CPHS，而读取后的结果一样在getSpnFsm()中处理：

```
private void getSpnFsm(boolean start, AsyncResult ar) {
    byte[] data;
    if (start) {
    }


    switch(mSpnState){
        case INIT:
            break;
        case READ_SPN_3GPP:
            break;
        case READ_SPN_CPHS:
            break;
        case READ_SPN_SHORT_CPHS:
            if (ar != null && ar.exception == null) {
                data = (byte[]) ar.result;
                setServiceProviderName(IccUtils.adnStringFieldToString(data, 0, da
ta.length));
                setSystemProperty(PROPERTY_ICC_OPERATOR_ALPHA, getServiceProviderN
ame());
            }else {
                if (DBG) log("No SPN loaded in either CHPS or 3GPP");
            }
            mSpnState = GetSpnFsmState.IDLE;
        default:
            mSpnState = GetSpnFsmState.IDLE;
    }
}
```

遇上面流程类似，读取成功就保存，不成功就不再处理。经过上面的过程，就将SIM中的SPN读取并保存起来了。

二、从配置文件读取SPN过程

Android原始代码中，无论在SIM的三个文件分区有没有查询到SPN，系统都会继续尝试从配置文件中读取SPN，如果读取成功，则**覆盖**刚才SIM中读取的值，如果配置文件读取失败，就使用上面的SIM中的SPN。

开发者可以将所有预置的SPN存入spn-conf.xml这个文件中（不同平台该文件的存储位置不同），在编译时候就会将其拷贝到out的system\etc\目录中，以供系统读取。我们来挑选几条该文件中的项来看一下：

```
//spn-conf.xml
<spnOverride numeric="46000" spn="CHINA MOBILE"/>
<spnOverride numeric="46001" spn="CHN-UNICOM"/>
<spnOverride numeric="46002" spn="CHINA MOBILE"/>
<spnOverride numeric="46003" spn="CHINA TELECOM"/>
<spnOverride numeric="46007" spn="CHINA MOBILE"/>
<spnOverride numeric="46008" spn="CHINA MOBILE"/>
<spnOverride numeric="46009" spn="CHN-UNICOM"/>
```

这些项是针对中国区的SPN，我们看到，每一项都包含两个元素，PLMN和SPN，**我们可以用当前SIM所驻留的网络的PLMN号码来匹配查找当前的SPN字串。**

下面我们来看如何将该文件读取到SPN中。SIMRecords对象在初始化时，在构造方法里面创建了一个SpnOverride对象：

```
//SIMRecords.java
public SIMRecords(UiccCardApplication app, Context c, CommandsInterface ci) {
    super(app, c, ci);
    mSpnOverride = new SpnOverride();
}
```

```
    public SpnOverride () {
        mCarrierSpnMap = new HashMap<String, String>();
        loadSpnOverrides();
    }
```

```
private void loadSpnOverrides() {
        FileReader spnReader;
        //PARTNER_SPN_OVERRIDE_PATH ="etc/spn-conf.xml";
        //OEM_SPN_OVERRIDE_PATH = "telephony/spn-conf.xml";
        File spnFile = new File(Environment.getRootDirectory(),PARTNER_SPN_OVERRID
E_PATH);
        File oemSpnFile = new File(Environment.getOemDirectory(),OEM_SPN_OVERRIDE_
PATH);
```

```
7.
8.          if (oemSpnFile.exists()) {
9.              // OEM image exist SPN xml, get the timestamp from OEM & System image
    for comparison.
10.             long oemSpnTime = oemSpnFile.lastModified();
11.             long sysSpnTime = spnFile.lastModified();
12.             Rlog.d(LOG_TAG, "SPN Timestamp: oemTime = " + oemSpnTime + " sysTime =
     " + sysSpnTime);
13.
14.             // To get the newer version of SPN from OEM image
15.             if (oemSpnTime > sysSpnTime) {
16.                 Rlog.d(LOG_TAG, "SPN in OEM image is newer than System image");
17.                 spnFile = oemSpnFile;
18.             }
19.         } else {
20.             // No SPN in OEM image, so load it from system image.
21.             Rlog.d(LOG_TAG, "No SPN in OEM image = " + oemSpnFile.getPath() +
22.                 " Load SPN from system image");
23.         }
24.
25.         try {
26.             spnReader = new FileReader(spnFile);
27.         } catch (FileNotFoundException e) {
28.             Rlog.w(LOG_TAG, "Can not open " + spnFile.getAbsolutePath());
29.             return;
30.         }
31.         //解析spn-conf.xml文件
32.         try {
33.             XmlPullParser parser = Xml.newPullParser();
34.             parser.setInput(spnReader);
35.
36.             XmlUtils.beginDocument(parser, "spnOverrides");
37.
38.             while (true) {
39.                 XmlUtils.nextElement(parser);
40.
41.                 String name = parser.getName();
42.                 if (!"spnOverride".equals(name)) {
43.                     break;
44.                 }
45.
46.                 String numeric = parser.getAttributeValue(null, "numeric");
47.                 String data    = parser.getAttributeValue(null, "spn");
48.
49.                 mCarrierSpnMap.put(numeric, data);
50.             }
51.             spnReader.close();
52.         } catch (XmlPullParserException e) {
53.             Rlog.w(LOG_TAG, "Exception in spn-conf parser " + e);
54.         } catch (IOException e) {
55.             Rlog.w(LOG_TAG, "Exception in spn-conf parser " + e);
56.         }
57.     }
```

这个对象在初始化时就将"etc/spn-conf.xml"文件加载进来，并进行XML解析，把每一项存入mCarrierSpnMap的HashMap中。
然后该对象提供了两个查询SPN的方法：

```
public boolean containsCarrier(String carrier) {
    //查询是否包含某个运营商的SPN
    return mCarrierSpnMap.containsKey(carrier);
}
public String getSpn(String carrier) {
    //获取某个运营商的SPN
    return mCarrierSpnMap.get(carrier);
}
```

然后我们接着上一节的介绍，当SIM中的SPN被读取之后，就会在SIMRecords中的handleMessage()消息中收到EVENT_GET_SPN_DONE的消息：

```
public void handleMessage(Message msg) {
    try {
        switch (msg.what) {
            case EVENT_GET_SPN_DONE:
                isRecordLoadResponse = true;
                ar = (AsyncResult) msg.obj;
                getSpnFsm(false, ar);
                break;
        }
    } catch (RuntimeException exc) {
    } finally {
        if (isRecordLoadResponse) {
            onRecordLoaded();
        }
    }
}
```

前面我们分析过，在getSpnFsm()中将会对Modem的返回结果进行解析，如果读取成功，就会将SPN字串保存起来，现在我们继续来看如果保存之后，将会进入finally的处理当中，也就是onRecordLoaded()方法：

```
protected void onRecordLoaded() {
    mRecordsToLoad -= 1;
    if (mRecordsToLoad == 0 && mRecordsRequested == true) {
        onAllRecordsLoaded();
    } else if (mRecordsToLoad < 0) {
        mRecordsToLoad = 0;
    }
}
```

这里的mRecordsToLoad表明当前需要读取的SIM信息条数(SIMRecords初始化过程中需要读取大量的SIM数据)，每向Modem发送一条读取的指令，该计数就会加1，当一条记录读取完毕后该计数就会减1，当所有记录全部读取完毕，就会进入onAllRecordsLoaded()的处理：

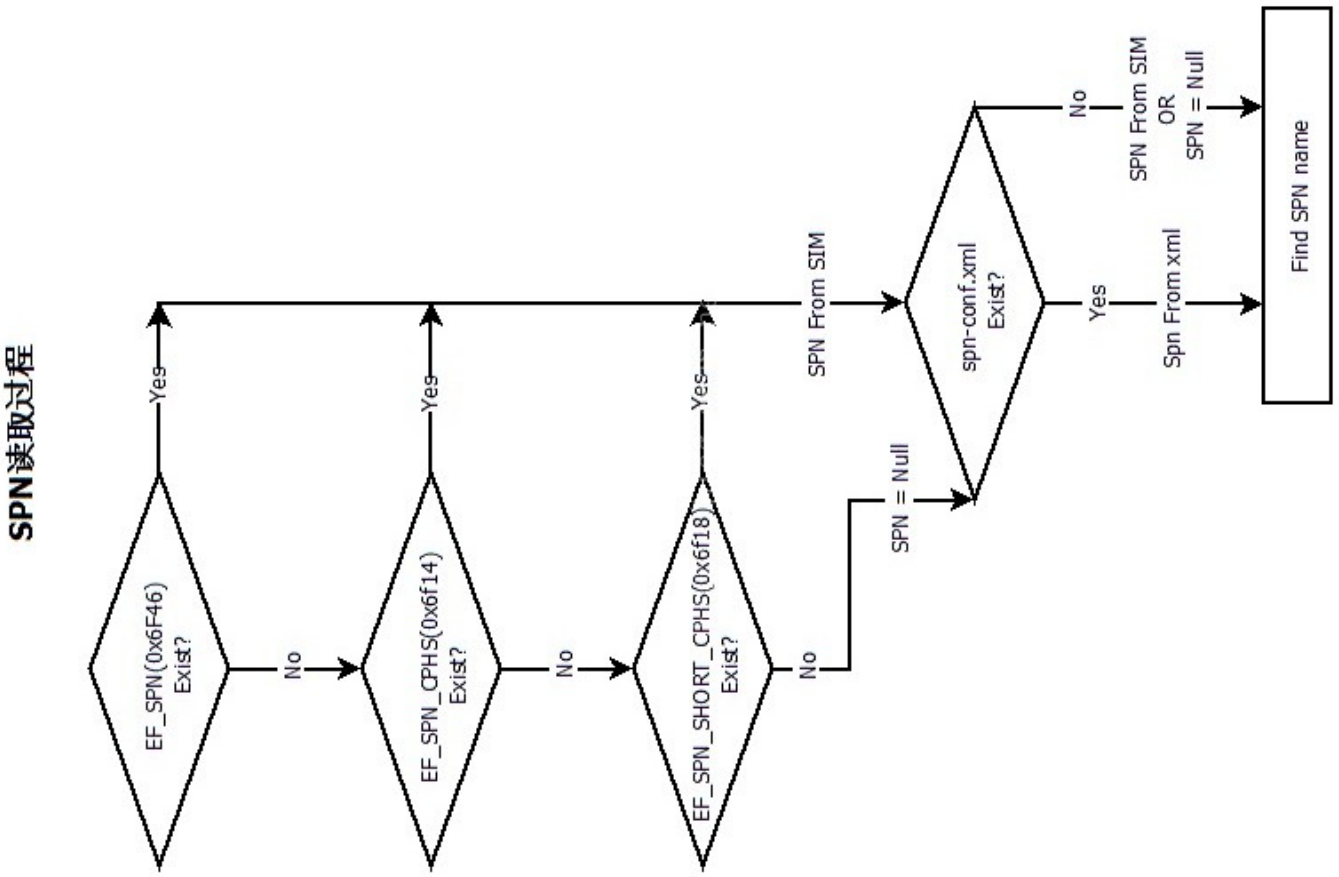
```
protected void onAllRecordsLoaded() {
    setLocaleFromUsim();
    if (mParentApp.getState() == AppState.APPSTATE_PIN || mParentApp.getState() ==
 AppState.APPSTATE_PUK) {
        mRecordsRequested = false;
        return ;
    }


    String operator = getOperatorNumeric();
    if (!TextUtils.isEmpty(operator)) {
        //保存当前的PLMN
        setSystemProperty(PROPERTY_ICC_OPERATOR_NUMERIC, operator);
        final SubscriptionController subController = SubscriptionController.getIns
tance();
        subController.setMccMnc(operator, subController.getDefaultSmsSubId());
    } else {
    }


    if (!TextUtils.isEmpty(mImsi)) {
        setSystemProperty(PROPERTY_ICC_OPERATOR_ISO_COUNTRY, MccTable.countryCodeF
orMcc(Integer.parseInt(mImsi.substring(0,3))));
    } else {
        log("onAllRecordsLoaded empty imsi skipping setting mcc");
    }
    //设置当前的语音信箱
    setVoiceMailByCountry(operator);
    //读取配置文件中的SPN
    setSpnFromConfig(operator);
    //将通知发送出来
    mRecordsLoadedRegistrants.notifyRegistrants( new AsyncResult(null, null, null)
);
}
```

我们来看setSpnFromConfig()的方法：

```
private void setSpnFromConfig(String carrier) {
    if (mSpnOverride.containsCarrier(carrier)) {
        //用配置文件中的SPN来作为最终的SPN
        setServiceProviderName(mSpnOverride.getSpn(carrier));
        SystemProperties.set(PROPERTY_ICC_OPERATOR_ALPHA, getServiceProviderName()
);
    }
}
```

这里就用上mSpnOverride这个对象了，前面我们分析过，他的作用就是把spn-conf.xml文件中的SPN信息解析出来，保存到HashMap中，现在我们需要根据当前的MCC/MNC去该HashMap中寻找匹配的SPN值，并把其作为最终的SPN保存起来。

以上就是整个SPN的读取流程，下面用一张逻辑图来展示上述过程：



第三部分：Android运营商名称显示之PLMN与SPN显示规则

上面的两节分别介绍了PLMN和SPN的读取方法，那么在**锁屏、状态栏、通知栏**这些地方的运营商名称究竟是来自于PLMN呢？还是来自于SPN呢？

在3GPP中规定的运营商名称显示规则如下：

1、名称可以为SPN或PLMN

2、如果没有SPN文件，那么就显示PLMN

3、若有SPN，并且注册的PLMN是HPLMN或者注册的PLMN在SIM卡文件EF_SPDI中，那么：

(1)如果有SPN就要显示SPN

(2)如果SPN的bit1 = 1, 则需要同时显示PLMN，如果SPN的bit1=0，则不需要同时显示PLMN

4、若有SPN，注册的PLMN是Roaming PLMN且注册的PLMN也不在SIM卡文件EF_SPDI中，那么

(1)显示PLMN

(2)如果SPN的bit2=0，则需要同时显示SPN，如果SPN的bit2=1，则不需要同时显示SPN

下面我们用代码来梳理上面的规则。

在GsmServiceStateTracker中，接收到EVENT_SIM_RECORDS_LOADED消息或者ACTION_LOCALE_CHANGED广播后，就会触发SPN的更新显示机制。其入口为updateSpnDisplay():

```
1.  //GsmServiceStateTracker.java
2.  protected void updateSpnDisplay() {
3.      IccRecords iccRecords = mIccRecords;
4.      String plmn = null;
5.      boolean showPlmn = false;
6.      int rule = (iccRecords != null) ? iccRecords.getDisplayRule(mSS.getOperatorNum
    eric()) : 0;
7.      if (mSS.getVoiceRegState() == ServiceState.STATE_OUT_OF_SERVICE || mSS.getVoic
    eRegState() == ServiceState.STATE_EMERGENCY_ONLY) {
8.          //当前无网络
9.          showPlmn = true;
10.         if (mEmergencyOnly) {
11.             // No service but emergency call allowed
12.             plmn = Resources.getSystem().  getText(com.android.internal.R.string.e
    mergency_calls_only).toString();
13.         } else {
14.             // No service at all
15.             plmn = Resources.getSystem().  getText(com.android.internal.R.string.l
    ockscreen_carrier_default).toString();
16.         }
17.     } else if (mSS.getVoiceRegState() == ServiceState.STATE_IN_SERVICE) {
18.         //当前注册网络OK
19.         plmn = mSS.getOperatorAlphaLong();
20.         showPlmn = !TextUtils.isEmpty(plmn) && ((rule & SIMRecords.SPN_RULE_SHOW_P
    LMN) == SIMRecords.SPN_RULE_SHOW_PLMN);
21.     } else {
22.         if (DBG) log("updateSpnDisplay: radio is off w/ showPlmn=" + showPlmn + "
    plmn=" + plmn);
23.     }
24.
25.     String spn = (iccRecords != null) ? iccRecords.getServiceProviderName() : "";

26.     boolean showSpn = !TextUtils.isEmpty(spn)
27.         && ((rule & SIMRecords.SPN_RULE_SHOW_SPN)
28.                 == SIMRecords.SPN_RULE_SHOW_SPN);
29.
30.     //发送Intent通知
31.     if (showPlmn != mCurShowPlmn
32.             || showSpn != mCurShowSpn
33.             || !TextUtils.equals(spn, mCurSpn)
34.             || !TextUtils.equals(plmn, mCurPlmn)) {
35.         Intent intent = new Intent(TelephonyIntents.SPN_STRINGS_UPDATED_ACTION);
36.         intent.addFlags(Intent.FLAG_RECEIVER_REPLACE_PENDING);
37.         intent.putExtra(TelephonyIntents.EXTRA_SHOW_SPN, showSpn);
38.         intent.putExtra(TelephonyIntents.EXTRA_SPN, spn);
39.         intent.putExtra(TelephonyIntents.EXTRA_SHOW_PLMN, showPlmn);
40.         intent.putExtra(TelephonyIntents.EXTRA_PLMN, plmn);
41.         mPhone.getContext().sendStickyBroadcastAsUser(intent, UserHandle.ALL);
42.     }
```

```
43.         mCurShowSpn = showSpn;
44.         mCurShowPlmn = showPlmn;
45.         mCurSpn = spn;
46.         mCurPlmn = plmn;
47.     }
```

上面的更新过程分为两步，分别完成SPN的读取和广播的发送，我们主要来看读取SPN的过程。

在读取过程中，先对当前的网络状态进行分类:

1、如果当前网络处于STATE_OUT_OF_SERVICE或者STATE_EMERGENCY_ONLY状态，则分别显示“No service”和“Emergency calls only”字串。

2、对于当前网络注册成功的情况(STATE_IN_SERVICE)，则根据3GPP协议来确定当前显示的PLMN显示规则。

我们主要关注第二种情况下显示规则的确认。

我们先来看一下显示的rule，他是通过以下调用来定义的：

```
1.  int rule = (iccRecords != null) ? iccRecords.getDisplayRule(mSS.getOperatorNumeric
    ()) : 0;
```

也就时说，这里的rule是通过SIMRecords的getDisplayRule()方法得到的：

```
1.  //SIMRecords.java
2.  public int getDisplayRule(String plmn) {
3.      int rule;
4.      if (TextUtils.isEmpty(mSpn) || mSpnDisplayCondition == -1) {
5.          //如果SPN为空，则显示PLMN(Rule 2)
6.          rule = SPN_RULE_SHOW_PLMN;
7.      } else if (isOnMatchingPlmn(plmn)) {
8.          //如果当前注册的PLMN为HPLMN或者注册的PLMN存在于SIM中的EF_SPDI字段内，则显
    示SPN(Rule 3.1)
9.          rule = SPN_RULE_SHOW_SPN;
10.         if ((mSpnDisplayCondition & 0x01) == 0x01) {
11.             //如果SPN的bit1=1，则需要同时显示SPN和PLMN(Rule 3.2)
12.             rule |= SPN_RULE_SHOW_PLMN;
13.         }
14.     } else {
15.         //如果注册的PLMN为Roaming PLMN，并且注册的PLMN不在EF_SPDI中，则显示PLMN(Ru
    le 4.1)
16.         rule = SPN_RULE_SHOW_PLMN;
17.         if ((mSpnDisplayCondition & 0x02) == 0x00) {
18.             //如果注册的PLMN为Roaming PLMN，并且注册的PLMN不在EF_SPDI中，并且SPN的
    bit2=0，则要同时显示PLMN和SPN(Rule 4.2)
19.             rule |= SPN_RULE_SHOW_SPN;
20.         }
21.     }
22.     return rule;
23. }
```

上面的过程刚好匹配3GPP对PLMN的显示规则，其中的mSpnDisplayCondition就是在SIMRecords获取到SPN时解析出来的data的bit1数据。当该方法结束时，返回出来了int类型的rule变量，该变量只有前两位bit有效，其中bit1=1代表显示SPN，bit2=1代表显示PLMN。

然后回到updateSpnDisplay()方法中，此时将会通过mSS.getOperatorAlphaLong()得到当前的PLMN值，以及通过rule得到当前是否需要显示PLMN(showPlmn)，然后通过iccRecords.getServiceProviderName()得到当前的SPN以及通过rule得到当前是否需要显示SPN。

拿到上面的数据之后，就通过广播(SPN_STRINGS_UPDATED_ACTION)的形式通知其他模块SPN的变化。

这就是SPN的显示机制，以下是该机制的逻辑图：