Sim 卡设置

SIM 卡设置相关类:

packages/apps/settings/src/com/android/settings/SimManagerActivity.java

onResume()方法:

```
initSim() – Log.d(TAG, " length " + mSimCounts)
prepareForDataAdapter()
prepareForAdapter() – Log.d(TAG, sims[i].toString())
  onClickListener()
     – Log.w(TAG, "simAdapter default-- position  = " +  v.getId())
     – Log.w(TAG, "simMmsAdapter default-- position  = " + v.getId())
   准备 adapter 并设置监听器响应用户操作
Log.d(TAG, "onResume: airplane="+airplane)
updateState()
```

updateState(): 更新 sim 卡设置的界面状态，并将数据写入偏好设置文件和数据库

```
updateConfigPrefrence()
   主要是获取 sim 卡数量和飞行模式，判断是否可操作设置界面
updateDataSummary()
   initSim()
   prepareForDataAdapter() – onCLickListener – preferenceChanged()
   getDefaultDataPhoneId() – Log.d(TAG, "updateDataSummary:defaultPhoneId=" +
Data_val)
updateVoiceSummary()
updateVedioSummary()(先判断是否支持视频通话)
updateMmsSummary()
   setSummary()
        – Log.d(TAG, "setSummary:mode=" + mode + " phoneId=" + phoneId)
        – Log.d(TAG, "setSummary: active Counts  = " + mSimCounts)
```

setSummary()方法主要调用 **Preference** 的 **setValue** 方法将用户设置存入 **SharedPreferences** 中，并调用 **Preference** 的 **setSummary() – notifyChanged** 来通知相关监听者 **sim** 卡设置的变化。

```
    private void setSummary(ListPreference pref, int mode) {
        int phoneId = 0;
        if (airplane) {
            pref.setEnabled(false);
            pref.setSummary(null);
            return;
        }
        if (mMms.getKey().equals(pref.getKey())) {
            pref.getBuilder().setAdapter(simMmsAdapter, null);
        } else {
            pref.getBuilder().setAdapter(simAdapter, null);
        }
        if (mVoice.getKey().equals(pref.getKey())) {
            phoneId = TelephonyManager.getDefaultSim(this, TelephonyManager.MODE_VOICE);
        } else if (mVideo.getKey().equals(pref.getKey())) {
            phoneId = TelephonyManager.getDefaultSim(this, TelephonyManager.MODE_VEDIO);
        } else if (mMms.getKey().equals(pref.getKey())) {
            phoneId = TelephonyManager.getDefaultSim(this, TelephonyManager.MODE_MMS);
        }

        if (DEBUG) Log.d(TAG, "setSummary:mode=" + mode + " phoneId=" + phoneId);
        if (mSimCounts >= 2) {
            if (DEBUG) Log.d(TAG, "setSummary: active Counts  = " + mSimCounts);
            if ((TelephonyManager.MODE_VEDIO == mode) && (supportMulticard(isVTCall) < 2)) {
                pref.setEnabled(false);
                if (supportMulticard(isVTCall) == 0) {
                    pref.setSummary(null);
                    return;
                }
            } else
                pref.setEnabled(true);
        } else {
            pref.setEnabled(false);
            if (mSimCounts == 0) {
                pref.setSummary(null);
```

```
                return;
            }
        }
        pref.setValue(Integer.toString(phoneId));
        if (mMms.getKey().equals(pref.getKey())) {
            for (int i = 0; i < simMms.length; i++) {
                if (simMms[i].getPhoneId() == phoneId) {
                    pref.setValue(Integer.toString(i));
                    pref.setSummary(simMms[i].getName());
                }
            }
        } else {
            for (int i = 0; i < sim.length; i++) {
                if (sim[i].getPhoneId() == phoneId) {
                    pref.setValue(Integer.toString(i));
                    pref.setSummary(sim[i].getName());
                }
            }
        }
    }
```

Handler处理接收到的消息：EVENT_SET_SUBSCRIPTION_TIMEOUT：表明用户设置结束，调用finishSettingsWait-closeTimer来结束TimerTask线程并关闭对话框，调用updateState更新数据和界面信息；*EVENT_SET_DATA_SUBSCRIPTION_DONE*：主卡设置完成，如果没有异常就会调用startTimer()方法。当时间超过DIALOG_WAIT_MAX_TIME = 5000ms时，会发出EVENT_SET_SUBSCRIPTION_TIMEOUT，则提示“设置成功”，并且使progressDialog消失。

```
    private Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            AsyncResult ar;
            switch (msg.what) {
                case EVENT_SET_SUBSCRIPTION_TIMEOUT:
                    Log.i(TAG, "EVENT_SET_SUBSCRIPTION_TIMEOUT");
                    finishSettingsWait();
                    updateState();
                    if (isLeave == 0) {
                        Toast toast = Toast.makeText(getApplicationContext(), getResources()
                                .getString(R.string.set_dds_success), Toast.LENGTH_LONG);
                        toast.show();
                    }
                    break;
                case EVENT_SET_DATA_SUBSCRIPTION_DONE:
                    Log.i(TAG, "EVENT_SET_DATA_SUBSCRIPTION_DONE");
                    if (setPhoneId >= 0) {
                        PhoneFactory.getPhone(setPhoneId).unregisterForGprsDetached(mHandler);
                    }
                    ar = (AsyncResult) msg.obj;
                    if (ar.exception == null) {
                        startTimer(DIALOG_WAIT_MAX_TIME);
                    }
                    break;
             }
        }
    };

    private void startTimer(int time) {
        closeTimer();
        timer = new ScheduledThreadPoolExecutor(1);
        timerTask = new TimerTask() {
            public void run() {
                // force execute busy act
                mHandler.sendEmptyMessage(EVENT_SET_SUBSCRIPTION_TIMEOUT);
            }
        };
        if (DEBUG) Log.d(TAG, "startTimer,timer start");
        timer.schedule(timerTask, time , TimeUnit.MILLISECONDS);
    }
```

```
    private void closeTimer() {
        if (DEBUG) Log.d(TAG, "closeTimer,timer end");
        if (timerTask != null) {
            timerTask.cancel();
            timerTask = null;
        }
        if (timer != null) {
            timer.shutdownNow();
            timer = null;
        }
    }
```

SimManagerActivity.java实现了**Preference.OnPreferenceClickListener**，通过**onPreferenceClick**方法来响应用户点击操作，而在不同**adapter**中的监听器的**onClick**方法中调用**preferenceChanged()**方法来响应用户操作：

```
    public boolean onPreferenceClick(Preference preference) {
        Log.w(TAG, " preference = " +preference.getKey());
        if(mData.getKey().equals(((ListPreference) preference).getKey())) {
            preferencekey = mData.getKey();
            simsAdapter.setMode(-1);
        }else {
            if ((ListPreference) preference == mVoice) {
                preferencekey = mVoice.getKey();
                simAdapter.setMode(TelephonyManager.MODE_VOICE);
            }else if ((ListPreference) preference == mMms) {
                preferencekey = mMms.getKey();
                simMmsAdapter.setMode(TelephonyManager.MODE_MMS);
            }else if ((ListPreference) preference == mVideo) {
                preferencekey = mVideo.getKey();
                simAdapter.setMode(TelephonyManager.MODE_VEDIO);
            }
        }
        return false;
    }

    public void preferenceChanged(String key,int phoneId) {
        int mode =-1;
        Log.w(TAG, "key = " + key);
        if (KEY_DATA.equals(key)) {
            startUpdateDataSettings(phoneId);
        } else if (KEY_VOICE.equals(key)) {
            mode = TelephonyManager.MODE_VOICE;
            TelephonyManager.setDefaultSim(this, TelephonyManager.MODE_VOICE, phoneId);
            setSummary(mVoice, mode);
        } else if (KEY_VIDEO.equals(key)) {
            mode = TelephonyManager.MODE_VEDIO;
            TelephonyManager.setDefaultSim(this, TelephonyManager.MODE_VEDIO, phoneId);
            setSummary(mVideo, mode);
        } else if (KEY_MMS.equals(key)) {
            mode = TelephonyManager.MODE_MMS;
            TelephonyManager.setDefaultSim(this, TelephonyManager.MODE_MMS, phoneId);
            setSummary(mMms, mode);
        }
    }
```

preferenceChanged方法根据传入的**key**执行不同的**if**分支，当用户操作的是主卡设置时，调用**startUpdateDataSettings**方法，而其他三项操作都是先调用**TelephonyManager**的**setDefaultSim**方法将数据写入数据库，然后再调用**setSummary**方法将用户设置写入**SharedPreferences**中并通知相关监听者。

```
    private void startUpdateDataSettings(int phoneId) {
        if (setPhoneId >= 0 && setPhoneId < TelephonyManager.getPhoneCount()) {
            oldSetPhoneId = setPhoneId;
        } else {
            oldSetPhoneId = TelephonyManager.getDefaultDataPhoneId(this
                    .getApplicationContext());
        }
        if (DEBUG) Log.d(TAG, "startUpdateDataSettings: " + phoneId + " setPhoneId=" +
setPhoneId
                + " old=" + oldSetPhoneId);
```

```
        if (setPhoneId == phoneId) {
            Toast toast = Toast.makeText(getApplicationContext(),
                    R.string.no_need_set_data_subscription,
                    Toast.LENGTH_LONG);
            toast.show();
            return;
        }
        setPhoneId = phoneId;
        if (MsmsGsmDataConnectionTrackerProxy.isActivePhoneId(setPhoneId)) {
            Log.w(TAG, "[" + setPhoneId + "]" + "already active phone, just start timer");
            startTimer(DIALOG_WAIT_MAX_TIME);
        } else {
            int activePhoneId = MsmsGsmDataConnectionTrackerProxy.getActivePhoneId();
            Log.w(TAG, "[" + activePhoneId + "]" + "is active phone, register for
GprsDetached");
            oldSetPhoneId = (activePhoneId < 0 ? oldSetPhoneId : activePhoneId);
            PhoneFactory.getPhone(oldSetPhoneId).registerForGprsDetached(mHandler,
                    EVENT_SET_DATA_SUBSCRIPTION_DONE, null);
        }
        restoreDataSettings(phoneId);
        getPreferenceScreen().setEnabled(false);
        this.showDialog(DIALOG_SET_DATA_SUBSCRIPTION_IN_PROGRESS);
        if(oldSetPhoneId >= 0 && (Settings.System.getInt(mContext.getContentResolver(),
PhoneFactory.getSetting(Settings.System.SIM_STANDBY, oldSetPhoneId), 1) == 0)){
            startTimer(BAN_CARD_DIALOG_WAIT_MAX_TIME);
        }
        isLeave = 0;
    }

    private void restoreDataSettings(int index) {
        boolean updateResult;
        if (DEBUG) Log.d(TAG, "restoreDataSettings: " + index);
        TelephonyManager.setAutoDefaultPhoneId(this, index);
        updateResult = PhoneFactory.updateDefaultPhoneId(index);
        if (DEBUG) Log.d(TAG, "SimmanagerA updateResult: " + updateResult);
        if (!updateResult) {
            startTimer(DIALOG_WAIT_MAX_TIME);
        }
    }
```

调用PhoneFactory的updateDefaultPhoneId方法会发送广播*ACTION_DEFAULT_PHONE_CHANGE*，SimManagerActivity在接收到该广播后调用updateDataSummary方法更新数据。