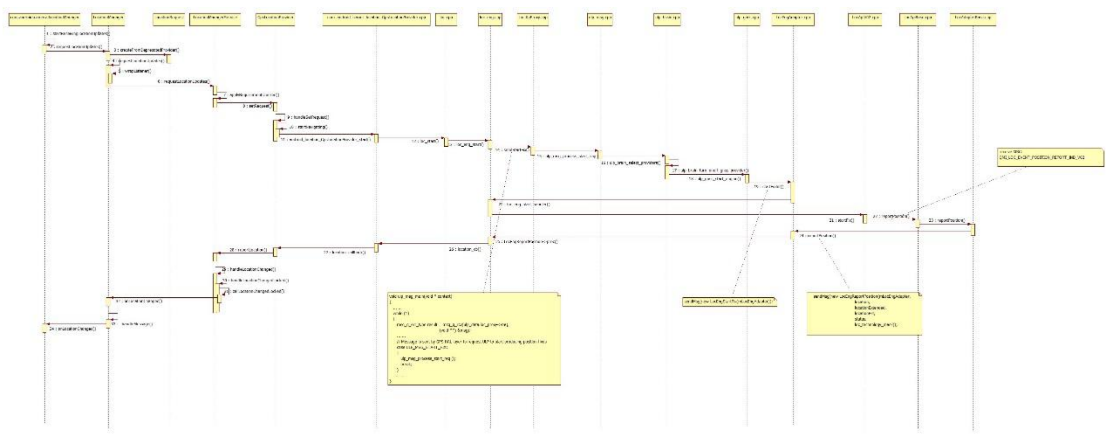
QCOM GPS Start&Report flow

目录



1. 应用层注册 location

以 camera 为例，通过 requestLocationUpdates 注册 provider

packages/apps/Camera/src/com/android/camera/LocationManager.java

```
    private void startReceivingLocationUpdates() {
        if (mLocationManager == null) {
            //获得 LM
            mLocationManager = (android.location.LocationManager)
                    mContext.getSystemService(Context.LOCATION_SERVICE);
        }
        if (mLocationManager != null) {
                try {
                ... ...
                try {
                    mLocationManager.requestLocationUpdates(
                            /*由 GPS 提供 location，室内没 GPS 信号，
                            只能在室外才能获取位置信息*/
                            android.location.LocationManager.GPS_PROVIDER,
                            /*设置两次更新位置的最短时间，
                            每隔 1s 上报一次位置信息*/
                            1000,
                            /*设置两次更新位置的最短距离，单位为米，
                            如果设置为 500，表示位置移动大于 500m 才会更新位置*/
                            0F,
                            /*当有位置信息更新时，就会调用 mLocationListeners 的
                            onLocationChanged(Location)方法*/
                            mLocationListeners[0]);
                    if (mListener != null) mListener.showGpsOnScreenIndicator(false);
                } catch (SecurityException ex) {
                  ... ...
            }
        }
```

2. requestLocationUpdates

App 在调用 requestLocationUpdates 注册了 provider 以后，最终会调用到 LocationManager 里面的 requestLocationUpdates，requestLocationUpdates 主要做了两件事。

frameworks/base/location/java/android/location/LocationManager.java

```
  private void requestLocationUpdates(LocationRequest request, LocationListener listener,
                Looper looper, PendingIntent intent) {
            android.util.SeempLog.record(47);
```

```
        String packageName = mContext.getPackageName();
        /*1. 给 App 注册的 LocationListener 配置一个 ListenerTransport，并放在
        HashMap<LocationListener,ListenerTransport> mListeners 里面*/
        // wrap the listener class
        ListenerTransport transport = wrapListener(listener, looper);

        try {
            //2. 调用 LocationManagerServices 的 requestLocationUpdates
            mService.requestLocationUpdates(request, transport, intent, packageName);
       } catch (RemoteException e) {
           Log.e(TAG, "RemoteException", e);
       }
    }
```

wrapListener 实现如下：

```
    private ListenerTransport wrapListener(LocationListener listener, Looper looper) {
        if (listener == null) return null;
        synchronized (mListeners) {
            //从 mListeners 取出 listener 对应的 transport
            ListenerTransport transport = mListeners.get(listener);
            //当 transport 为 null 时，则 new 一个 ListenerTransport 对象
            if (transport == null) {
                transport = new ListenerTransport(listener, looper);
            }
            //将 listener, transport 放到 mListeners 里面
            mListeners.put(listener, transport);
            return transport;
        }
    }
```

ListenerTransport 继承了 ILocationListener.Stub，里面有只有两个属性，主要是功能是当 LMS 有位置信息更新时，通过 ListenerTransport 发送 message 通知 app 注册的 LocationListener 更新位置信息

```
   // 给 App 注册时的 LocationListener
   private LocationListener mListener;

   // 用来处理位置更新的 Handler
   private final Handler mListenerHandler;

   ListenerTransport(LocationListener listener, Looper looper) {
           mListener = listener;

         if (looper == null) {
              mListenerHandler = new Handler() {
                  @Override
                  public void handleMessage(Message msg) {
                      /*1.  当 GPS 位置信息有更新，状态发生改变等等，
                       就会发信息给 mListenerHandler，  调用_handleMessage
                       处理 message*/
                      _handleMessage(msg);
                  }
              };
           }
           ... ...
       }

    private void _handleMessage(Message msg) {
           switch (msg.what) {
              case TYPE_LOCATION_CHANGED:
```

```
                        Location location = new Location((Location) msg.obj);
                        /* 2.  通知 app 注册的 LocationListener 更新位置信息,
                          即调用 app 层实现的 onLocationChanged 方法*/
                        mListener.onLocationChanged(location);
                        break;
```

3. requestLocationUpdatesLocked

由上面的描述可以知道，LM 不会直接去设置 provider，而是通过 LMS 的 requestLocationUpdates->requestLocationUpdatesLocked->applyRequirementsLocked 去设置一个 provider

frameworks/base/services/core/java/com/android/server/LocationManagerService.java

```
    public void requestLocationUpdates(LocationRequest request, ILocationListener listener,
            PendingIntent intent, String packageName) {
        ... ...
            synchronized (mLock) {
                Receiver recevier = checkListenerOrIntentLocked(listener, intent, pid, uid,
                        packageName, workSource, hideFromAppOps);
                requestLocationUpdatesLocked(sanitizedRequest, recevier, pid, uid,
packageName);
            }
        } finally {
            Binder.restoreCallingIdentity(identity);
        }
    }

    private void requestLocationUpdatesLocked(LocationRequest request, Receiver receiver,
            int pid, int uid, String packageName) {
        // Figure out the provider. Either its explicitly request (legacy use cases), or
        // use the fused provider
        ... ...
```

```
            if (isProviderEnabled) {
                applyRequirementsLocked(name);
            } else {
                // Notify the listener that updates are currently disabled
                receiver.callProviderEnabledLocked(name, false);
            }
            // Update the monitoring here just in case multiple location requests were added to
the
            // same receiver (this request may be high power and the initial might not have been).
            receiver.updateMonitoring(true);
        }

        private void applyRequirementsLocked(String provider) {
            //  通过 loadProvidersLocked 将支持的 provider 都放到 mProvidersByName 里面
            LocationProviderInterface p = mProvidersByName.get(provider);
            if (p == null) return;

            ... ...
            if (D) Log.d(TAG, "provider request: " + provider + " " + providerRequest);
            //调用注册的 provider 的 setRequest 方法
            p.setRequest(providerRequest, worksource);
        }
```

4. setRequest

frameworks/base/services/core/java/com/android/server/location/GpsLocationProvider.java

```
        @Override
        public void setRequest(ProviderRequest request, WorkSource source) {
            //发送 SET_REQUEST 的 message 给 ProviderHandler
            sendMessage(SET_REQUEST, 0, new GpsRequest(request, source));
```

```
}

private final class ProviderHandler extends Handler {
    public ProviderHandler(Looper looper) {
        super(looper, null, true /*async*/);
    }

    @Override
    public void handleMessage(Message msg) {
        int message = msg.what;
        switch (message) {
            ... ...
            case SET_REQUEST:
                GpsRequest gpsRequest = (GpsRequest) msg.obj;
                //处理 Request GPS 的请求
                handleSetRequest(gpsRequest.request, gpsRequest.source);
                break;
            ... ...
    }
private void handleSetRequest(ProviderRequest request, WorkSource source) {
    mProviderRequest = request;
    mWorkSource = source;
    updateRequirements();
}
private void updateRequirements() {
    if (mProviderRequest == null || mWorkSource == null) {
        return;
    }

        ... ...
```

```
                } else if (!mStarted) {
                        // start GPS
                        //如果 GPS 没 start，先启动导航
                        startNavigating(singleShot);
                }
        } else {
              … …
        }
}
private void startNavigating(boolean singleShot) {
      … …
                //调用 JNI 层方法
                if (!native_start()) {
                        mStarted = false;
                        Log.e(TAG, "native_start failed in startNavigating()");
                        return;
                }

                // reset SV count to zero
                updateStatus(LocationProvider.TEMPORARILY_UNAVAILABLE, 0);
                … …
        }
}
```

5. native_start

frameworks/base/services/core/jni/com_android_server_location_GpsLocationProvider.cpp

```
{"native_start", "()Z", (void*)android_location_GpsLocationProvider_start},
static jboolean android_location_GpsLocationProvider_start(JNIEnv* /* env */, jobject /* obj */)
```

```
{
        if (sGpsInterface) {
                // sGpsInterface 在 GPS init 部分已经说过了，这里不再重复了
                if (sGpsInterface->start() == 0) {
                        return JNI_TRUE;
                } else {
                        return JNI_FALSE;
                }
        }
        else
                return JNI_FALSE;
}
```

6. loc_start

hardware/qcom/gps/loc_api/libloc_api_50001/loc.cpp

```
static int loc_start()
{
        ENTRY_LOG();
        int ret_val = loc_eng_start(loc_afw_data);

        EXIT_LOG(%d, ret_val);
        return ret_val;
}
```

hardware/qcom/gps/loc_api/libloc_api_50001/loc_eng.cpp

```
int loc_eng_start(loc_eng_data_s_type &loc_eng_data)
{
      ENTRY_LOG_CALLFLOW();
      INIT_CHECK(loc_eng_data.adapter, return -1);
      /* 调用 LocUlpProxy 的 sendStartFix，为什么是调的 LocUlpProxy 的 sendStartFix
      在 GPS init 里面有说明*/
```

```
    if(! loc_eng_data.adapter->getUlpProxy()->sendStartFix())
    {
        loc_eng_data.adapter->sendMsg(new LocEngStartFix(loc_eng_data.adapter));
    }

    EXIT_LOG(%d, 0);
    return 0;
}
```

vendor/qcom/proprietary/gps/ulp2/src/LocUlpProxy.cpp

```
bool LocUlpProxy::sendStartFix()
{
    //发送消息 ULP_MSG_START_FIX
    ulp_msg *msg(new ulp_msg(this, ULP_MSG_START_FIX));
    msg_q_snd(mQ, msg, ulp_msg_free);
    return true;
}
```

vendor/qcom/proprietary/gps/ulp2/src/ulp_msg.cpp

```
void ulp_msg_main(void * context)
{
    ... ...
    // Message is sent by GPS HAL layer to request ULP to start producing position fixes
        case ULP_MSG_START_FIX:
        {
            //处理 ULP_MSG_START_FIX 消息
            ulp_msg_process_start_req ();
            break;
        }
    ... ...
}
int ulp_msg_process_start_req (void)
```

```
{
    int ret_val = -1;
    ENTRY_LOG();

    LOC_LOGI ("%s, at ulp state = %d\n", __func__, ulp_data.ulp_started);

    do
    {
        ulp_data.ulp_started = true;

        ... ...
        if (ulp_data.run_provider_selection_logic == true)
        {
            ret_val = ulp_brain_select_providers ();
        }
    } while (0);

    EXIT_LOG(%d, ret_val);
    return ret_val;
}
```

ulp_msg.cpp 里面所有的函数开头前的注释里面，都有描述这些函数是干什么用的，可以帮助对代码的理解。ulp_brain_select_providers 会根据设置来打开或者关闭 3 个 GPS provider，这 3 个 provider 分别指 GNSS GPS，GNP GPS，ZPP GPS

```
/*===========================================================================
FUNCTION    ulp_brain_select_providers
DESCRIPTION
    This function will evaulate all three providers based on recent position
    requests, phone context settings, provider state update.
      DEPENDENCIES
    None
```

```
RETURN VALUE
    0: success
    -1: failure
SIDE EFFECTS
    N/A
=============================================================================*/
int ulp_brain_select_providers ()
{
    int ret_val = -1;

    ENTRY_LOG();
        ... ...
        // This function is called when to start/stop GNSS provider based
        ulp_brain_turn_onoff_gnss_provider    ();
        ... ...
    return ret_val;
}
/*=============================================================================
FUNCTION       ulp_brain_turn_onoff_gnss_provider
DESCRIPTION
    This function is called when to start/stop GNSS provider based
    on its recent state change.
    The following will trigger QUIPC state change:
    (1) GPS enabled via UI
    (2) GPS Provider based or high accuracy fix request addition or
        removal
    (3) Recurrence type, fix interval and position mode changes of
        GPS provider based or high accuracy fix requests
    (4) GNSS provider state change, QUIPC provider state change,
        and GNP provider state change
```

```
DEPENDENCIES
    None
RETURN VALUE
    0: success
    -1: failure
SIDE EFFECTS
    N/A
===========================================================================*/
static int ulp_brain_turn_onoff_gnss_provider ()
{
    int            ret_val = -1;
    ... ...
        /*This function is called to configure and start GNSS. libulp module posts
        messages to GPS HAL layer via message queue for the request.*/
        ulp_gnss_start_engine ();
    ... ..
    return ret_val;
}
int ulp_gnss_start_engine ()
{
    int                              ret_val = -1;
    ... ...
        if (ulp_gnss_engine_running () == false || gnss_need_configuration == true)
        {
          /*LocInternalAdapter 是 LocAdapterBase  的子类，
          最终调用到 LocInternalAdapter::startFixInt*/
            adapter->startFixInt();
        }
        ... ...
    EXIT_LOG(%d, ret_val);
    return ret_val;
}
void LocInternalAdapter::startFixInt() {
        /*LocInternalAdapter 是 LocAdapterBase  的子类,
        最终调用的是 LocAdapterBase 的 sendMsg*/
        sendMsg(new LocEngStartFix(mLocEngAdapter));
}
```

hardware/qcom/gps/core/LocAdapterBase.h

```
class LocAdapterBase {
        … …
        inline void sendMsg(const LocMsg* msg) const {
                //发送 message 给 MsgTask
                mMsgTask->sendMsg(msg);
        }
```

hardware/qcom/gps/utils/MsgTask.cpp

在 MsgTask::run 函数中处理由 mMsgTask->sendMsg 发送的消息

```
bool MsgTask::run() {
        LOC_LOGV("MsgTask::loop() listening ...\n");
        LocMsg* msg;
        /*这里的 msg， 就是通过 sendMsg(new LocEngStartFix(mLocEngAdapter));中
        传入的参数 LocEngStartFix 对象*/
        msq_q_err_type result = msg_q_rcv((void*)mQ, (void **)&msg);
        … …
        msg->log();
        // there is where each individual msg handling is invoked
        //  其实调用的是 LocEngStartFix 的 proc 函数
        msg->proc();

        delete msg;
        return true;
}
```

hardware/qcom/gps/loc_api/libloc_api_50001/loc_eng.cpp

```
LocEngStartFix::LocEngStartFix(LocEngAdapter* adapter) :
        LocMsg(), mAdapter(adapter)
{
        locallog();
}
inline void LocEngStartFix::proc() const
{
        loc_eng_data_s_type* locEng = (loc_eng_data_s_type*)mAdapter->getOwner();
        loc_eng_start_handler(*locEng);
}
static int loc_eng_start_handler(loc_eng_data_s_type &loc_eng_data)
{
      ENTRY_LOG();
      int ret_val = LOC_API_ADAPTER_ERR_SUCCESS;

      if (!loc_eng_data.adapter->isInSession()) {
              /*最终会调到 LocApiV02.cpp 的 startFix，为什么是调用的 LocApiV02.cpp 的 startFix，
              在 GPS init 中有描述*/
              ret_val = loc_eng_data.adapter->startFix();

            ... ...
      }

      EXIT_LOG(%d, ret_val);
      return ret_val;
}
```

LocApiV02 已经是调到 HAL 层了，GPS start 部分结束

vendor/qcom/opensource/location/loc_api/loc_api_v02/LocApiV02.cpp

```
/* start positioning session */

enum loc_api_adapter_err LocApiV02 :: startFix(const LocPosMode& fixCriteria)

{

    locClientStatusEnumType status;

    locClientReqUnionType req_union;


    qmiLocStartReqMsgT_v02 start_msg;


    … …


    return convertErr(status);

}
```

7. reportPosition(JNI 层)

当收到 QMI_LOC_EVENT_POSITION_REPORT_IND_V02 消息时，就会逐级上报位置信息了，这里不再描述，调用过程直接看代码比较快。

vendor/qcom/opensource/location/loc_api/loc_api_v02/LocApiV02.cpp

```
/* event callback registered with the loc_api v02 interface */

void LocApiV02 :: eventCb(locClientHandleType clientHandle,

    uint32_t eventId, locClientEventIndUnionType eventPayload)

{

    LOC_LOGD("%s:%d]: event id = %d\n", __func__, __LINE__,

                        eventId);

    switch(eventId)

    {

        //Position Report

        case QMI_LOC_EVENT_POSITION_REPORT_IND_V02:

            reportPosition(eventPayload.pPositionReportEvent);
            break;
            ... ...
        }
        ... ...
    }
    void LocApiV02 :: reportPosition (
        const qmiLocEventPositionReportIndMsgT_v02 *location_report_ptr)
    {
            ... ...
            LocApiBase::reportPosition(location,
                                        locationExtended,
                                        NULL,
                                        LOC_SESS_FAILURE);

            ... ...
    }
```

hardware/qcom/gps/core/LocApiBase.cpp

```
void LocApiBase::reportPosition(UlpLocation &location,
                                GpsLocationExtended &locationExtended,
                                void* locationExt,
                                enum loc_sess_status status,
                                LocPosTechMask loc_technology_mask)
{
    ... ...
      TO_ALL_LOCADAPTERS(
          mLocAdapters[i]->reportPosition(location,
                                          locationExtended,
                                          locationExt,
                                          status,
                                          loc_technology_mask)
    );
}
```

hardware/qcom/gps/core/LocAdapterBase.cpp

```
void LocAdapterBase::
    reportPosition(UlpLocation &location,
                   GpsLocationExtended &locationExtended,
                   void* locationExt,
                   enum loc_sess_status status,
                   LocPosTechMask loc_technology_mask) {
    if (mLocAdapterProxyBase == NULL ||
        !mLocAdapterProxyBase->reportPosition(location,
                                              locationExtended,
                                              status,
                                              loc_technology_mask)) {
        DEFAULT_IMPL()
    }
}
```

hardware/qcom/gps/loc_api/libloc_api_50001/LocEngAdapter.cpp

```
void LocEngAdapter::reportPosition(UlpLocation &location,
                                   GpsLocationExtended &locationExtended,
                                   void* locationExt,
                                   enum loc_sess_status status,
                                   LocPosTechMask loc_technology_mask)
{
    if (! mUlp->reportPosition(location,
                               locationExtended,
                               locationExt,
                               status,
                               loc_technology_mask )) {
        mInternalAdapter->reportPosition(location,
                                              locationExtended,
                                              locationExt,
                                              status,
                                              loc_technology_mask);
    }
}



void LocInternalAdapter::reportPosition(UlpLocation &location,
                                        GpsLocationExtended &locationExtended,
                                        void* locationExt,
                                        enum loc_sess_status status,
                                        LocPosTechMask loc_technology_mask)
{
    //sendMsg 调用和前面的类似，不清楚的可以倒回去看看调用流程
    sendMsg(new LocEngReportPosition(mLocEngAdapter,
                                     location,
                                     locationExtended,
                                     locationExt,
                                     status,
                                     loc_technology_mask));
}
```

hardware/qcom/gps/loc_api/libloc_api_50001/loc_eng.cpp

```
void LocEngReportPosition::proc() const {
   … …
                locEng->location_cb((UlpLocation*)&(mLocation),
                                    (void*)mLocationExt);
   … …
}
```

frameworks/base/services/core/jni/com_android_server_location_GpsLocationProvider.cpp

```
static void location_callback(GpsLocation* location)
{
    JNIEnv* env = AndroidRuntime::getJNIEnv();
    //已经调到JNI 层了
    env->CallVoidMethod(mCallbacksObj, method_reportLocation, location->flags,
            (jdouble)location->latitude, (jdouble)location->longitude,
            (jdouble)location->altitude,
            (jfloat)location->speed, (jfloat)location->bearing,
            (jfloat)location->accuracy, (jlong)location->timestamp);
    checkAndClearExceptionFromCallback(env, __FUNCTION__);
}
```

8. reportLocation(应用层)

frameworks/base/services/core/java/com/android/server/location/GpsLocationProvider.java

```
    private void reportLocation(int flags, double latitude, double longitude, double altitude,
            float speed, float bearing, float accuracy, long timestamp) {
        … …

            try {
                //调用 LMS 的 reportLocation 方法
                mILocationManager.reportLocation(mLocation, false);
            } catch (RemoteException e) {
                Log.e(TAG, "RemoteException calling reportLocation");
            }
       … …
   }
```

frameworks/base/services/core/java/com/android/server/LocationManagerService.java

```
    public void reportLocation(Location location, boolean passive) {
        checkCallerIsProvider();

        if (!location.isComplete()) {
            Log.w(TAG, "Dropping incomplete location: " + location);
            return;
        }

        mLocationHandler.removeMessages(MSG_LOCATION_CHANGED, location);
        //发送消息给 mLocationHandler，通知有位置信息更新
        Message m = Message.obtain(mLocationHandler, MSG_LOCATION_CHANGED,
location);
        m.arg1 = (passive ? 1 : 0);
        mLocationHandler.sendMessageAtFrontOfQueue(m);
    }
    private class LocationWorkerHandler extends Handler {
        public LocationWorkerHandler(Looper looper) {
            super(looper, null, true);
        }

        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case MSG_LOCATION_CHANGED:
                    //调用 handleLocationChanged 方法，处理位置更新的消息
                    handleLocationChanged((Location) msg.obj, msg.arg1 == 1);
                    break;
            }
        }
    }
    private void handleLocationChanged(Location location, boolean passive) {
        ... ...

        synchronized (mLock) {
            if (isAllowedByCurrentUserSettingsLocked(provider)) {
                if (!passive) {
                    location = screenLocationLocked(location, provider);
                    if (location == null) {
                        return;
                    }
                    // notify passive provider of the new location
                    mPassiveProvider.updateLocation(myLocation);
                }
                handleLocationChangedLocked(myLocation, passive);
            }
        }
    }
    private void handleLocationChangedLocked(Location location, boolean passive) {
        if (D) Log.d(TAG, "incoming location: " + location);

        ... ...
        // Skip if the provider is unknown.
        LocationProviderInterface p = mProvidersByName.get(provider);
        if (p == null) return;

       ... ...
        // Skip if there are no UpdateRecords for this provider.
        ArrayList<UpdateRecord> records = mRecordsByProvider.get(provider);
       ... ...

        // Broadcast location or status to all listeners
```

```
            for (UpdateRecord r : records) {
                Receiver receiver = r.mReceiver;
                … …
                        if (!receiver.callLocationChangedLocked(notifyLocation)) {
                            Slog.w(TAG, "RemoteException calling onLocationChanged on " +
receiver);
                            receiverDead = true;
                        }
                        r.mRequest.decrementNumUpdates();
                    }
                }
               … …
        }
        public boolean callLocationChangedLocked(Location location) {
            … …
            /* 调到 LocationManager 中 ListenerTransport -> onLocationChanged */
                        mListener.onLocationChanged(new Location(location));
                        // call this after broadcasting so we do not increment
                        // if we throw an exeption.
            … …
        }
```

frameworks/base/location/java/android/location/LocationManager.java

```
        private class ListenerTransport extends ILocationListener.Stub {
          … …
            @Override
            public void onLocationChanged(Location location) {
                Message msg = Message.obtain();
                msg.what = TYPE_LOCATION_CHANGED;
                msg.obj = location;
                //发送消息 TYPE_LOCATION_CHANGED 给 mListenerHandler
                mListenerHandler.sendMessage(msg);
            }
            ListenerTransport(LocationListener listener, Looper looper) {
                mListener = listener;

                if (looper == null) {
                    mListenerHandler = new Handler() {
                        @Override
                        public void handleMessage(Message msg) {
                            _handleMessage(msg);
                        }
                    };
                }
                ... ...
            }
            ... ...
          }
      private void _handleMessage(Message msg) {
                switch (msg.what) {
                    case TYPE_LOCATION_CHANGED:
                        Location location = new Location((Location) msg.obj);
                        //通知所有 APP 层注册的 listener 更新位置信息
                        mListener.onLocationChanged(location);
                        break;
                    ... ...
       }
```

packages/apps/Camera/src/com/android/camera/LocationManager.java

```
        //调用 APP 层的 onLocationChanged
        public void onLocationChanged(Location newLocation) {
```

```
        if (newLocation.getLatitude() == 0.0

                    && newLocation.getLongitude() == 0.0) {

            // Hack to filter out 0.0,0.0 locations

            return;

        }

        // If GPS is available before start camera, we won't get status

        // update so update GPS indicator when we receive data.

        if (mListener != null && mRecordLocation &&

                    android.location.LocationManager.GPS_PROVIDER.equals(mProvider)) {

            mListener.showGpsOnScreenIndicator(true);

        }

        if (!mValid) {

            Log.d(TAG, "Got first location.");

        }

        //设置上报的位置信息

        mLastLocation.set(newLocation);

        mValid = true;

    }
```