WEA流程和关键字

概述

该流程从底层到上层，逐层分解。

1. ril_unsol_commands.h

文件路径：/v590_s/android/vendor/mediatek/proprietary/hardware/ril/include/telephony/ril_unsol_commands.h

```
{RIL_UNSOL_RESPONSE_NEW_BROADCAST_SMS, radio::newBroadcastSmsInd, WAKE_PARTIAL},
```

2. ril_service.cpp

文件路径：　/v590_s/android/vendor/mediatek/proprietary/hardware/ril/rilproxy/libril/ril_service.cpp
LOG关键字：　**newBroadcastSmsInd**

```
int radio::newBroadcastSmsInd(unsigned int slotId,
```

```
                                int indicationType, int token, RIL_Errno e, v
oid *response,
                                size_t responseLen) {
    if (radioService[slotId] != NULL && radioService[slotId]->mRadioIndicat
ion != NULL) {
        if (response == NULL || responseLen == 0) {
            RLOGE("newBroadcastSmsInd: invalid response");
            return 0;
        }

        hidl_vec<uint8_t> data;
        data.setToExternal((uint8_t *) response, responseLen);
        RLOGD("newBroadcastSmsInd");
        Return<void> retStatus = radioService[slotId]->mRadioIndication->ne
wBroadcastSms(
                convertIntToRadioIndicationType(indicationType), data);
        radioService[slotId]->checkReturnStatus(retStatus);
    } else {
        RLOGE("newBroadcastSmsInd: radioService[%d]->mRadioIndication == NU
LL", slotId);
    }

    return 0;
}
```

3. RadioIndication.java

文件路径： /v590_s/android/frameworks/opt/telephony/src/java/com/android/internal/telephony/RadioIndication.java

LOG关键字： **UNSOL_RESPONSE_NEW_BROADCAST_SMS**

```
public void newBroadcastSms(int indicationType, ArrayList<Byte> data) {
        mRil.processIndication(indicationType);

        byte response[] = RIL.arrayListToPrimitiveArray(data);
        if (RIL.RILJ_LOGD) {
            mRil.unsljLogvRet(RIL_UNSOL_RESPONSE_NEW_BROADCAST_SMS,
```

```
                    IccUtils.bytesToHexString(response));
        }

        if (mRil.mGsmBroadcastSmsRegistrant != null) {
            mRil.mGsmBroadcastSmsRegistrant.notifyRegistrant(new AsyncResul
t(null, response, null));
        }
    }
```

4. RIL.java

文件路径： /v590_s/android/frameworks/opt/telephony/src/java/com/android/internal/telephony/RIL.java

5. CellBroadcastServiceManager.java

文件路径： /v590_s/android/frameworks/opt/telephony/src/java/com/android/internal/telephony/CellBroadcastServiceManager.java

LOG关键字： **GSM SMS CB for phone**

```
                    if (msg.what == EVENT_NEW_GSM_SMS_CB) {
                        mLocalLog.log("GSM SMS CB for phone " + mPhone.getP
honeId());
                        CellBroadcastStatsLog.write(CellBroadcastStatsLog.C
B_MESSAGE_REPORTED,
                                CellBroadcastStatsLog.CELL_BROADCAST_MESSAG
E_REPORTED__TYPE__GSM,
                                CellBroadcastStatsLog.CELL_BROADCAST_MESSAG
E_REPORTED__SOURCE__FRAMEWORK);
                        cellBroadcastService.handleGsmCellBroadcastSms(mPho
ne.getPhoneId(),
                                (byte[]) ((AsyncResult) msg.obj).result);
                    }
```

5. CellBroadcastService.java

文件路径： /v590_s/android/frameworks/base/telephony/java/android/telephony/CellBroadcastService.java

```
        @Override
        public void handleGsmCellBroadcastSms(int slotIndex, byte[] message
) {
            CellBroadcastService.this.onGsmCellBroadcastSms(slotIndex, mess
age);
        }
```

7. DefaultCellBroadcastService.java

文件路径： /v590_s/android/packages/modules/CellBroadcastService/src/com/android/cellbroadcastservice/DefaultCellBroadcastService.java

LOG关键字： **onGsmCellBroadcastSms received message on slotId**

```
    @Override
    public void onGsmCellBroadcastSms(int slotIndex, byte[] message) {
        Log.d(TAG, "onGsmCellBroadcastSms received message on slotId=" + sl
otIndex);
        CellBroadcastStatsLog.write(CellBroadcastStatsLog.CB_MESSAGE_REPORT
ED,
                CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_REPORTED__TYPE
__GSM,
                CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_REPORTED__SOUR
CE__CB_SERVICE);
        mGsmCellBroadcastHandler.onGsmCellBroadcastSms(slotIndex, message);
    }
```

8. GsmCellBroadcastHandler.java

文件路径： /v590_s/android/packages/modules/CellBroadcastService/src/com/android/cellbroadcastservice/

GsmCellBroadcastHandler.java

LOG关键字： **onGsmCellBroadcastSms received message on slotId**

```
    /**
     * Handle a GSM cell broadcast message passed from the telephony framew
ork.
     * @param message
     */
    public void onGsmCellBroadcastSms(int slotIndex, byte[] message) {
        sendMessage(EVENT_NEW_SMS_MESSAGE, slotIndex, -1, message);
    }
```

9. StateMachine.java

文件路径： /v590_s/android/packages/modules/CellBroadcastService/src/com/android/cellbroadcastservice/GsmCellBroadcastHandler.java

```
    public void sendMessage(int what, int arg1, int arg2, Object obj) {
        // mSmHandler can be null if the state machine has quit.
        SmHandler smh = mSmHandler;
        if (smh == null) return;

        smh.sendMessage(obtainMessage(what, arg1, arg2, obj));
    }
```

```
        /**
         * Handle messages sent to the state machine by calling
         * the current state's processMessage. It also handles
         * the enter/exit calls and placing any deferred messages
         * back onto the queue when transitioning to a new state.
         */
        @Override
        public final void handleMessage(Message msg) {
            if (!mHasQuit) {
                if (mSm != null && msg.what != SM_INIT_CMD && msg.what != S
M_QUIT_CMD) {
                    mSm.onPreHandleMessage(msg);
```

```
                }

                if (mDbg) mSm.log("handleMessage: E msg.what=" + msg.what);

                /** Save the current message */
                mMsg = msg;

                /** State that processed the message */
                State msgProcessedState = null;
                if (mIsConstructionCompleted || (mMsg.what == SM_QUIT_CMD))
 {
                    /** Normal path */
                    msgProcessedState = processMsg(msg);
                } else if (!mIsConstructionCompleted && (mMsg.what == SM_IN
IT_CMD)
                        && (mMsg.obj == mSmHandlerObj)) {
                    /** Initial one time path. */
                    mIsConstructionCompleted = true;
                    invokeEnterMethods(0);
                } else {
                    throw new RuntimeException("StateMachine.handleMessage:
 "
                            + "The start method not called, received msg: "
 + msg);
                }
                //消息处理完毕状态切换 更新mStateStack
                performTransitions(msgProcessedState, msg);

                // We need to check if mSm == null here as we could be quit
ting.
                if (mDbg && mSm != null) mSm.log("handleMessage: X");

                if (mSm != null && msg.what != SM_INIT_CMD && msg.what != S
M_QUIT_CMD) {
                    mSm.onPostHandleMessage(msg);
                }
            }
        }
```

```
        //处理当前消息到state中去处理
        /**
         * Process the message. If the current state doesn't handle
         * it, call the states parent and so on. If it is never handled the
n
         * call the state machines unhandledMessage method.
         * @return the state that processed the message
         */
        private final State processMsg(Message msg) {
            StateInfo curStateInfo = mStateStack[mStateStackTopIndex];
            if (mDbg) {
                mSm.log("processMsg: " + curStateInfo.state.getName());
            }

            if (isQuit(msg)) {
                transitionTo(mQuittingState);
            } else {
                while (!curStateInfo.state.processMessage(msg)) {
                    /**
                     * Not processed
                     */
                    curStateInfo = curStateInfo.parentStateInfo;
                    if (curStateInfo == null) {
                        /**
                         * No parents left so it's not handled
                         */
                        mSm.unhandledMessage(msg);
                        break;
                    }
                    if (mDbg) {
                        mSm.log("processMsg: " + curStateInfo.state.getName
());

                    }
                }
            }
            return (curStateInfo != null) ? curStateInfo.state : null;
        }
```

10. WakeLockStateMachine.java

处理EVENT_NEW_SMS_MESSAGE消息
文件路
径： /v590_s/android/frameworks/opt/telephony/src/java/com/android/internal/telephony/WakeLockStateMachine.java
LOG关键字： **Idle: new cell broadcast message**

```
    class IdleState extends State {
        @Override
        public void enter() {
            sendMessageDelayed(EVENT_RELEASE_WAKE_LOCK, WAKE_LOCK_TIMEOUT);
        }

        @Override
        public void exit() {
            mWakeLock.acquire();
            if (DBG) log("Idle: acquired wakelock, leaving Idle state");
        }

        @Override
        public boolean processMessage(Message msg) {
            switch (msg.what) {
                case EVENT_NEW_SMS_MESSAGE:
                    log("Idle: new cell broadcast message");
                    // transition to waiting state if we sent a broadcast
                    if (handleSmsMessage(msg)) {
                        transitionTo(mWaitingState);
                    }
                    return HANDLED;

                case EVENT_RELEASE_WAKE_LOCK:
                    log("Idle: release wakelock");
                    releaseWakeLock();
                    return HANDLED;

                case EVENT_BROADCAST_NOT_REQUIRED:
                    log("Idle: broadcast not required");
                    return HANDLED;
```

```
                default:
                    return NOT_HANDLED;
            }
        }
    }
```

11. CellBroadcastHandler.java

文件路径： /v590_s/android/packages/modules/CellBroadcastService/src/com/android/cellbroadcastservice/CellBroadcastHandler.java

LOG关键字：

Broadcast the message directly because no geo-fencing required,

Dispatching emergency SMS CB, SmsCbMessage is:

```
    @Override
    protected boolean handleSmsMessage(Message message) {
        if (message.obj instanceof SmsCbMessage) {
            if (!isDuplicate((SmsCbMessage) message.obj)) {
                handleBroadcastSms((SmsCbMessage) message.obj);
                return true;
            } else {
                CellBroadcastStatsLog.write(CellBroadcastStatsLog.CB_MESSAG
E_FILTERED,
                        CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_FILTER
ED__TYPE__CDMA,
                        CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_FILTER
ED__FILTER__DUPLICATE_MESSAGE);
            }
            return false;
        } else {
            final String errorMessage =
                    "handleSmsMessage got object of type: " + message.obj.g
etClass().getName();
            loge(errorMessage);
            CellBroadcastStatsLog.write(CellBroadcastStatsLog.CB_MESSAGE_ER
```

```
ROR,
                    CELL_BROADCAST_MESSAGE_ERROR__TYPE__UNEXPECTED_CDMA_MES
SAGE_TYPE_FROM_FWK,
                    errorMessage);
            return false;
        }
    }
```

```
    /**
     * Dispatch a Cell Broadcast message to listeners.
     * @param message the Cell Broadcast to broadcast
     */
    protected void handleBroadcastSms(SmsCbMessage message) {
        int slotIndex = message.getSlotIndex();

        // TODO: Database inserting can be time consuming, therefore this s
hould be changed to
        // asynchronous.
        ContentValues cv = message.getContentValues();
        Uri uri = mContext.getContentResolver().insert(CellBroadcasts.CONTE
NT_URI, cv);

        if (message.needGeoFencingCheck()) {
            int maximumWaitingTime = getMaxLocationWaitingTime(message);
            if (DBG) {
                log("Requesting location for geo-fencing. serialNumber = "
                        + message.getSerialNumber() + ", maximumWaitingTime
 = "
                        + maximumWaitingTime);
            }

            CbSendMessageCalculator calculator =
                    mCbSendMessageCalculatorFactory.createNew(mContext, mes
sage.getGeometries());
            requestLocationUpdate(new LocationUpdateCallback() {
                @Override
                public void onLocationUpdate(@NonNull LatLng location,
                        float accuracy) {
                    if (VDBG) {
```

```
                        logd("onLocationUpdate: location=" + location
                                + ", acc=" + accuracy + ". "  + getMessageS
tring(message));
                    }
                    performGeoFencing(message, uri, calculator, location, s
lotIndex,
                            accuracy);
                    if (!isMessageInAmbiguousState(calculator)) {
                        cancelLocationRequest();
                    }
                }

                @Override
                public void onLocationUnavailable() {
                    CellBroadcastHandler.this.onLocationUnavailable(
                            calculator, message, uri, slotIndex);
                }
            }, maximumWaitingTime);
        } else {
            if (DBG) {
                log("Broadcast the message directly because no geo-fencing
required, "
                        + " needGeoFencing = " + message.needGeoFencingChec
k() + ". "
                        + getMessageString(message));
            }
            broadcastMessage(message, uri, slotIndex);
        }
    }
```

```
    /**
     * Broadcast the {@code message} to the applications.
     * @param message a message need to broadcast
     * @param messageUri message's uri
     */
    protected void broadcastMessage(@NonNull SmsCbMessage message, @Nullabl
e Uri messageUri,
            int slotIndex) {
        String msg;
```

```
        Intent intent;
        if (VDBG) {
            logd("broadcastMessage: " + getMessageString(message));
        }

        if (message.isEmergencyMessage()) {
            msg = "Dispatching emergency SMS CB, SmsCbMessage is: " + messa
ge;
            log(msg);
            mLocalLog.log(msg);
            intent = new Intent(Telephony.Sms.Intents.ACTION_SMS_EMERGENCY_
CB_RECEIVED);
            //Emergency alerts need to be delivered with high priority
            intent.addFlags(Intent.FLAG_RECEIVER_FOREGROUND);

            intent.putExtra(EXTRA_MESSAGE, message);
            putPhoneIdAndSubIdExtra(mContext, intent, slotIndex);

            if (IS_DEBUGGABLE) {
                // Send additional broadcast intent to the specified packag
e. This is only for sl4a
                // automation tests.
                String[] testPkgs = mContext.getResources().getStringArray(
                        R.array.test_cell_broadcast_receiver_packages);
                if (testPkgs != null) {
                    Intent additionalIntent = new Intent(intent);
                    for (String pkg : testPkgs) {
                        additionalIntent.setPackage(pkg);
                        mContext.createContextAsUser(UserHandle.ALL, 0).sen
dOrderedBroadcast(
                                intent, null, (Bundle) null, null, getHandl
er(),
                                Activity.RESULT_OK, null, null);

                    }
                }
            }

            List<String> pkgs = new ArrayList<>();
            pkgs.add(getDefaultCBRPackageName(mContext, intent));
```

```
            pkgs.addAll(Arrays.asList(mContext.getResources().getStringArra
y(
                    R.array.additional_cell_broadcast_receiver_packages)));
            if (pkgs != null) {
                mReceiverCount.addAndGet(pkgs.size());
                for (String pkg : pkgs) {
                    // Explicitly send the intent to all the configured cel
l broadcast receivers.
                    intent.setPackage(pkg);
                    mContext.createContextAsUser(UserHandle.ALL, 0).sendOrd
eredBroadcast(
                            intent, null, (Bundle) null, mOrderedBroadcastR
eceiver, getHandler(),
                            Activity.RESULT_OK, null, null);
                }
            }
        } else {
            msg = "Dispatching SMS CB, SmsCbMessage is: " + message;
            log(msg);
            mLocalLog.log(msg);
            // Send implicit intent since there are various 3rd party carri
er apps listen to
            // this intent.

            mReceiverCount.incrementAndGet();
            CellBroadcastIntents.sendSmsCbReceivedBroadcast(
                    mContext, UserHandle.ALL, message, mOrderedBroadcastRec
eiver, getHandler(),
                    Activity.RESULT_OK, slotIndex);
        }

        if (messageUri != null) {
            ContentValues cv = new ContentValues();
            cv.put(CellBroadcasts.MESSAGE_BROADCASTED, 1);
            mContext.getContentResolver().update(CellBroadcasts.CONTENT_URI
, cv,
                    CellBroadcasts._ID + "=?", new String[] {messageUri.get
LastPathSegment()});
        }
    }
```

12. CellBroadcastReceiver.java

文件路
径： /v590_s/android/packages/apps/CellBroadcastReceiver/src/com/android/cellbroadcastreceiver/CellBroadcastReceiver.java

```
        } else if (Telephony.Sms.Intents.ACTION_SMS_EMERGENCY_CB_RECEIVED.e
quals(action) ||
                Telephony.Sms.Intents.SMS_CB_RECEIVED_ACTION.equals(action)
) {
            intent.setClass(mContext, CellBroadcastAlertService.class);
            mContext.startService(intent);
        }
```

13. CellBroadcastAlertService.java

文件路
径： /v590_s/android/packages/apps/CellBroadcastReceiver/src/com/android/cellbroadcastreceiver/CellBroadcastAlertService.java
LOG关键字：

onStartCommand: android.provider.action.SMS_EMERGENCY_CB_RECEIVED

onStartCommand: cellbroadcastreceiver.SHOW_NEW_ALERT

```
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        mContext = getApplicationContext();
        String action = intent.getAction();
        Log.d(TAG, "onStartCommand: " + action);
        if (Telephony.Sms.Intents.ACTION_SMS_EMERGENCY_CB_RECEIVED.equals(a
ction) ||
                Telephony.Sms.Intents.SMS_CB_RECEIVED_ACTION.equals(action)
) {
            handleCellBroadcastIntent(intent);
        } else if (SHOW_NEW_ALERT_ACTION.equals(action)) {
            if (UserHandle.myUserId() == ((ActivityManager) getSystemServic
```

```
e(
                    Context.ACTIVITY_SERVICE)).getCurrentUser()) {
                showNewAlert(intent);
            } else {
                Log.d(TAG, "Not active user, ignore the alert display");
            }
        } else {
            Log.e(TAG, "Unrecognized intent action: " + action);
        }
        return START_NOT_STICKY;
    }
```

```
    private void handleCellBroadcastIntent(Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras == null) {
            Log.e(TAG, "received SMS_CB_RECEIVED_ACTION with no extras!");
            return;
        }

        SmsCbMessage message = (SmsCbMessage) extras.get(EXTRA_MESSAGE);

        if (message == null) {
            Log.e(TAG, "received SMS_CB_RECEIVED_ACTION with no message ext
ra");
            return;
        }

        if (message.getMessageFormat() == MESSAGE_FORMAT_3GPP) {
            CellBroadcastStatsLog.write(CellBroadcastStatsLog.CB_MESSAGE_RE
PORTED,
                    CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_REPORTED__
TYPE__GSM,
                    CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_REPORTED__
SOURCE__CB_RECEIVER_APP);
        } else if (message.getMessageFormat() == MESSAGE_FORMAT_3GPP2) {
            CellBroadcastStatsLog.write(CellBroadcastStatsLog.CB_MESSAGE_RE
PORTED,
                    CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_REPORTED__
TYPE__CDMA,
```

```
                    CellBroadcastStatsLog.CELL_BROADCAST_MESSAGE_REPORTED__
SOURCE__CB_RECEIVER_APP);
        }

        if (!shouldDisplayMessage(message)) {
            return;
        }

        final Intent alertIntent = new Intent(SHOW_NEW_ALERT_ACTION);
        alertIntent.setClass(this, CellBroadcastAlertService.class);
        alertIntent.putExtra(EXTRA_MESSAGE, message);

        // write to database on a background thread
        new CellBroadcastContentProvider.AsyncCellBroadcastTask(getContentR
esolver())
                .execute((CellBroadcastContentProvider.CellBroadcastOperati
on) provider -> {
                    CellBroadcastChannelManager channelManager =
                            new CellBroadcastChannelManager(mContext, messa
ge.getSubscriptionId());
                    CellBroadcastChannelRange range = channelManager
                            .getCellBroadcastChannelRangeFromMessage(messag
e);
                    // Check if the message was marked as do not display. S
ome channels
                    // are reserved for biz purpose where the msg should be
 routed as a data SMS
                    // rather than being displayed as pop-up or notificatio
n. However,
                    // per requirements those messages might also need to w
rite to sms inbox...
                    boolean ret = false;
                    if (range != null && range.mDisplay == true) {
                        if (provider.insertNewBroadcast(message)) {
                            // new message, show the alert or notification
on UI thread
                            // if not display..
                            startService(alertIntent);
                            // mark the message as displayed to the user.
                            markMessageDisplayed(message);
```

```
                            ret = true;
                        }
                    } else {
                        Log.d(TAG, "ignoring the alert due to configured ch
annels was marked "
                                + "as do not display");
                    }
                    if (CellBroadcastSettings.getResources(mContext, messag
e.getSubscriptionId())
                            .getBoolean(R.bool.enable_write_alerts_to_sms_i
nbox)) {
                        if (CellBroadcastReceiver.isTestingMode(getApplicat
ionContext())
                                || (range != null && range.mWriteToSmsInbox
)) {
                            provider.writeMessageToSmsInbox(message, mConte
xt);
                        }
                    }
                    return ret;
                });
    }
```

```
   private void showNewAlert(Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras == null) {
            Log.e(TAG, "received SHOW_NEW_ALERT_ACTION with no extras!");
            return;
        }

        SmsCbMessage cbm = intent.getParcelableExtra(EXTRA_MESSAGE);

        if (cbm == null) {
            Log.e(TAG, "received SHOW_NEW_ALERT_ACTION with no message extr
a");
            return;
        }

        if (mTelephonyManager.getCallState() != TelephonyManager.CALL_STATE
```

```
_IDLE
                && CellBroadcastSettings.getResources(mContext, cbm.getSubs
criptionId())
                .getBoolean(R.bool.enable_alert_handling_during_call)) {
            Log.d(TAG, "CMAS received in dialing/during voicecall.");
            sRemindAfterCallFinish = true;
        }

        // Either shown the dialog, adding it to notification (non emergenc
y, or delayed emergency),
        CellBroadcastChannelManager channelManager = new CellBroadcastChann
elManager(
                mContext, cbm.getSubscriptionId());
        if (channelManager.isEmergencyMessage(cbm) && !sRemindAfterCallFini
sh) {
            // start alert sound / vibration / TTS and display full-screen
alert
            openEmergencyAlertNotification(cbm);
            Resources res = CellBroadcastSettings.getResources(mContext, cb
m.getSubscriptionId());
            // KR carriers mandate to always show notifications along with
alert dialog.
            if (res.getBoolean(R.bool.show_alert_dialog_with_notification)
||
                    // to support emergency alert on companion devices use
flag
                    // show_notification_if_connected_to_companion_devices
instead.
                    (res.getBoolean(R.bool.show_notification_if_connected_t
o_companion_devices)
                            && isConnectedToCompanionDevices())) {
                // add notification to the bar by passing the list of unrea
d non-emergency
                // cell broadcast messages. The notification should be of L
OW_IMPORTANCE if the
                // notification is shown together with full-screen dialog.
                addToNotificationBar(cbm, CellBroadcastReceiverApp.addNewMe
ssageToList(cbm),
                        this, false, true, shouldDisplayFullScreenMessage(c
bm));
```

```
            }
        } else {
            // add notification to the bar by passing the list of unread no
n-emergency
            // cell broadcast messages
            ArrayList<SmsCbMessage> messageList = CellBroadcastReceiverApp
                    .addNewMessageToList(cbm);
            addToNotificationBar(cbm, messageList, this, false, true, false
);
        }
    }

    static void addToNotificationBar(SmsCbMessage message,
            ArrayList<SmsCbMessage> messageList, Context context,
            boolean fromSaveState, boolean shouldAlert, boolean fromDialog)
 {
    ...
        // Create intent to show the new messages when user selects the not
ification.
        Intent intent;
        if (isWatch) {
            // For FEATURE_WATCH we want to mark as read
            intent = createMarkAsReadIntent(context, message.getReceivedTim
e());
        } else {
            // For anything else we handle it normally
            intent = createDisplayMessageIntent(context, CellBroadcastAlert
Dialog.class,
                    messageList);
        }
    ...
    }

    private static Intent createDisplayMessageIntent(Context context, Class
 intentClass,
            ArrayList<SmsCbMessage> messageList) {
        // Trigger the list activity to fire up a dialog that shows the rec
eived messages
        Intent intent = new Intent(context, intentClass);
        intent.putParcelableArrayListExtra(CellBroadcastAlertService.SMS_CB
```

```
_MESSAGE_EXTRA,
                messageList);
        intent.addFlags(Intent.FLAG_ACTIVITY_NO_USER_ACTION);
        return intent;
    }
```