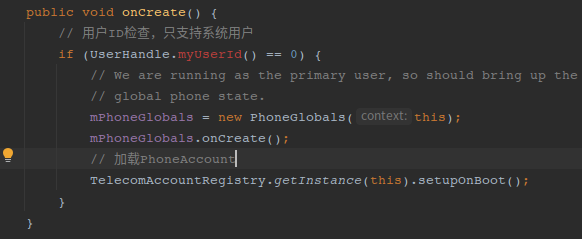
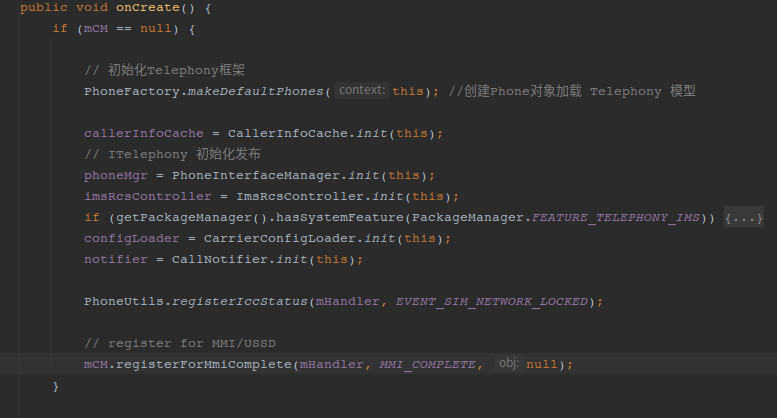
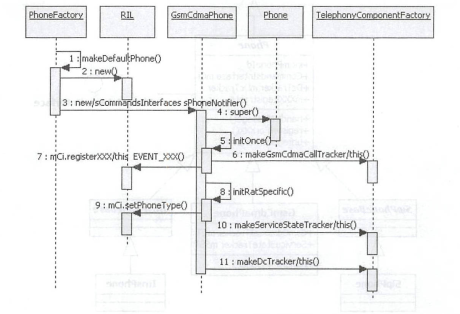
# TeleService

#### 加载过程

Phone进程启动：

1. com.android.server.SystemServer#main
2. com.android.server.SystemServer#run
3. com.android.server.SystemServer#startCoreServices
4. com.android.server.am.ActivityManagerService#systemReady
5. com.android.server.am.ActivityManagerService#startPersistentApps
6. com.android.server.am.ActivityManagerService#addAppLocked
7. com.android.server.am.ProcessList#newProcessRecordLocked
8. com.android.server.am.ProcessList#startProcessLocked
9. com.android.server.am.ProcessList#startProcess
10. android.os.ZygoteProcess#start
11. 启动完成，这个用户进程会被attach到系统中，调用ActivityThread中的main方法。
12. com.android.phone.PhoneApp#onCreate
13. com.android.phone.PhoneGlobals#onCreate
    * 1. **创建 Phone对象：**调用 PhoneFactory 类的静态方法 makeDefaultPhones 创建GsmCdmaPhone对象,将同步创建和初始化Telephony 业务模型的核心对象,比如 RILJ 和各种 Tracker 对象。
      2. **CallManager的消息注册：**向所有创建的GsmCdmaPhone对象注册相关消息(双卡或多卡)。
      3. **初始化ITelephony服务：**通过 PhoneInterfaceManager的init 法创建ITelephony .Stub 类型的Binder对象,再通过ServiceManager.add()的调用增加系统服务ITelephony。
      4. **其他操作：**包括 TelephonyDebugService 、NotificationMgr等初始化操作,并注册了两个广播接收器 mReceiver和 mSipReceiver。

#### TelephonyPhone



com.android.phone.PhoneApp#onCreate

com.android.phone.PhoneGlobals#onCreate

com.android.internal.telephony.PhoneFactory#makeDefaultPhones

com.android.internal.telephony.PhoneFactory#makeDefaultPhone

com.android.internal.telephony.TelephonyComponentFactory#makePhone

com.android.internal.telephony.GsmCdmaPhone#GsmCdmaPhone

com.android.internal.telephony.GsmCdmaPhone#initOnce

com.android.internal.telephony.TelephonyComponentFactory#makeGsmCdmaCallTracker

com.android.internal.telephony.TelephonyComponentFactory#makeServiceStateTracker

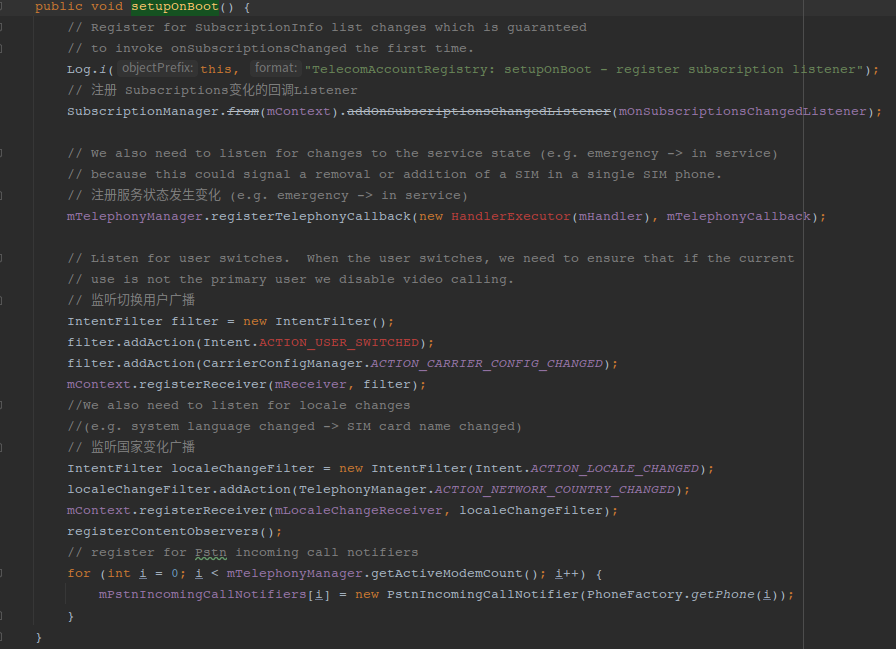
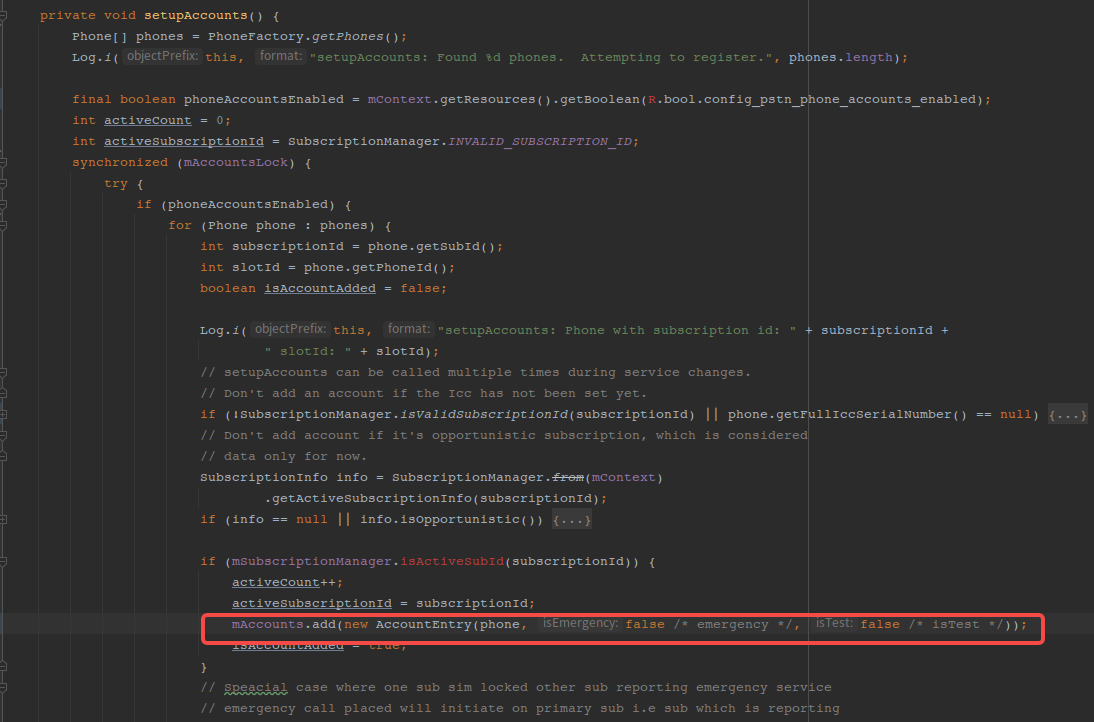
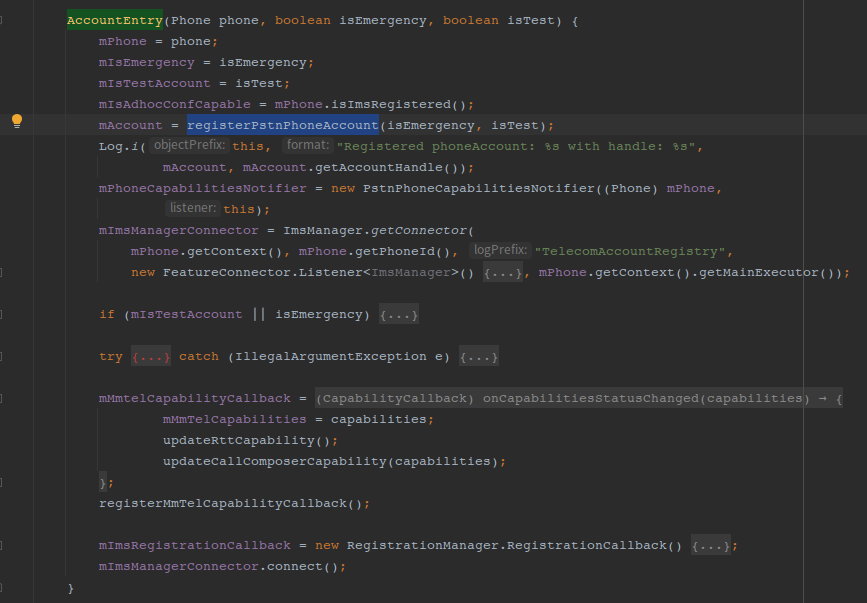
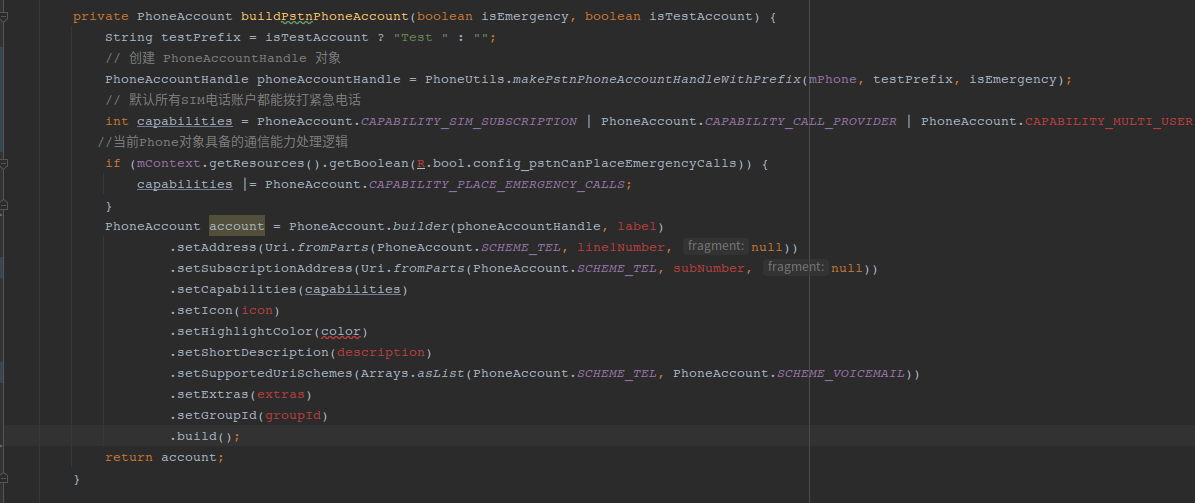
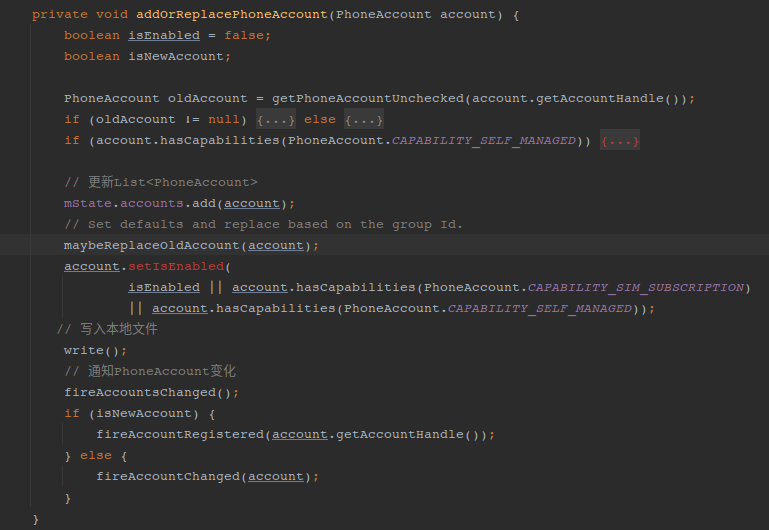
com.android.internal.telephony.TelephonyComponentFactory#makeDcTracker

com.android.internal.telephony.GsmCdmaPhone#initRatSpecific

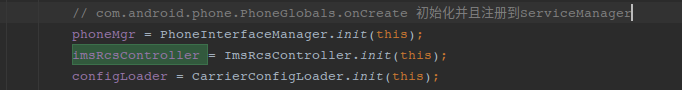
Telephony业务加载过程，以GsmCdmaPhone对象为中心,同步创建GsmCdmaCallTracler、ServiceStateTracker、DcTracker和RILJ 等关键业务对象的过程,同时向RILJ对象注册HandlerMessage回调消息。

* **加载入口：**PhoneFactory#makeDefaultPhones是Telephony业务模型的加载入口 。
* **GsmCdmaPhone对象创建：** 传入RILJ、PhoneNotifier、Phone类型（GSM或CDMA）、TelephonyComponentFactory。
* **phoneId的隐藏逻辑：**步骤 2 和步骤 3 是在for循环中创建RILJ和Phone对象并传入循环下标值作为phoneId的参数。比如双卡双待numberPhones取值 2 ,将创建两个RILJ对象和两个Phone对象。
* **GsmCdmaPhone对象的消息注册：**步骤 7 中调用registerXXX方法向RILJ注册 Registrantlist 消息, 关注this对象和 EVENT\_XXX消息类别,也就是向RILJ注册 GsmCdmaPhone对象的HandlerMessage回调消息 。
* **创建Tracker对象：** 步骤6、步骤10和步骤11在GsmCdmaPhone构造方法中, 通过TelephonyComponentFactory类的makeXXXTracker方法，创建非常关键的三个对象: GsmCdamCallTracker、ServiceStateTracker、Dctracker，它们将分别承载Telephony业务模型种非常重要的三个业务能力：VoiceCall语音通话、ServiceState网络服务和DataCall移动数据业务。
* **GsmCdmaPhone：** Phone抽象类实现PhoneInternalInterface接口并继承Handler类,通过重写handleMessage实现消息的异步处理。Phone抽象类三个子类：GsmCdmaPhone（CS域业务）、ImsPhone（PS域业务）、SipPhone（Sip网络业务） 。
* GsmCdmaPhone类的构造方法可体现它与mCi 、mCT 、mSST 、 mDcTracker等对象之间的强组合关系。
* GsmCdmaPhone为Facade (门面),而GsmCdmaCallTracker、ServiceStateTracker、 DcTracker 、 RILJ都作为内部的子系统。
* GsmCdmaPhone消息处理机制： ①基本Handler消息注册相应机制，②Registrantlist 封装的Handler消息运行机制，③创建的Message对象作为 RILJ 对象回调。

#### PhoneAccount

1. PhoneAccount初始化过程：
   1. com.android.phone.PhoneApp#onCreate
   2. com.android.services.telephony.TelecomAccountRegistry#setupOnBoot回调相应，调用tearDownAccounts()和setupAccounts() 
   3. com.android.services.telephony.TelecomAccountRegistry#setupAccounts
   4. com.android.services.telephony.TelecomAccountRegistry.AccountEntry#AccountEntry
   5. com.android.services.telephony.TelecomAccountRegistry.AccountEntry#registerPstnPhoneAccount
   6. com.android.services.telephony.TelecomAccountRegistry.AccountEntry#buildPstnPhoneAccountPhoneAccount对象的mAccountHandle属性为PhoneAccountHandle类型, mid 取值是GsmCdmaPhone 对象读取SIM卡的 ICCID ,与 GsmCdmaPhone 对象建立了对应关系 。
   7. android.telecom.TelecomManager#registerPhoneAccount向Telecom应用注册PhoneAccount。
2. PhoneAccount注册响应
   1. com.android.server.telecom.TelecomServiceImpl#mBinderImpl#registerPhoneAccount
   2. com.android.server.telecom.PhoneAccountRegistrar#registerPhoneAccount
   3. com.android.server.telecom.PhoneAccountRegistrar#addOrReplacePhoneAccount
3. PhoneAccount在拨号中的作用：Telecom 系统应用中晌应的拨号请求处理逻辑,最关键的是CallsManager对象的startOutgoingCall和placeOutgoingCall 方法调用,它们分别创建Call对象和发起Connection请求,这两个方法都涉及PhoneAccount对象的使用和传递 。
   1. startOutgoingCall： startOutgoningCall中的逻辑根据拨号请求匹配到 TeleService 注册的PhoneAccount对象 ,然后通过call.setTargetPhoneAccount调用,将 PhoneAccount与Call 产生了关联 。
   2. placeOutgoingCall：
   3. createConnection：
   4. IConnectionService：PhoneAccount 对象转了一圈之后又 回到TeleService 系统应用,在 IConnectionService的createConnection如onCreateOutgoingConnection 方法中,经过 getPhoneForAccount 调用最终获取了 GsmCdmaPhone 对象 。GsmCdmaPhone 与 PhoneAccountHandle 对象通过 PhoneAccountHand le的 mid 即 ICCID 与 GsmCdmaPhone 对象进行的关联.
   5. 小结：PhoneAccount 是在 TeleService 系统应用中创建的, 通过 PhoneAccountHandle 的 Id 来标识或关联 GsmCdmaPhone 对象, ComponentName 对象则用来标识 IConnectionService 服务。 Telecom系统应用接收到 PhoneAccount 注册后,将保存在 PhoneAccountRegistrar 对象的 List<PhoneAccount>列表中。产生通话业务时, Telecom 系统应用根据 List<PhoneAccount>列表判断是否匹配对应的通话能力,最终保存 PhoneAccountHandle 信息到 ConnectionRequest 对象,发起创建 Connection 请求。TeleService 系统应用接收到请求后,解析出 Connection Request 对象中的 PhoneAccountHandle 信息,通过 Id 即 SIM 卡的 ICCID 获取对应的 GsmCdmaPhone 对象,最后使用 GsmCdmaPhone 对象发起对应的通话管理或控制请求。

* PhoneAccount 是在TeleService系统应用中创建的，注册到 Telecom 系统应用。
* PhoneAccount 的唯一标识为PhoneAccountHandle。
* PhoneAccountHandle 通过mid即 ICCID与GsmCdmaPhone对象产生了唯一关联。
* PhoneAcount的主要功能是描述GsmCdmaPhone对象的通话能力,其中最关键的是 capabilities和 supported\_uri\_schemes。
* Telecom 系统应用通过拨号请求信息匹配对应的 PhoneAcount ( 能力匹配和IConnectionService服务匹配),过滤非法或不支持的拨号请求 。
* TeleService 系统应用接收 createConnection 请求, 通过 PhoneAccount 找出关联的GsmCdmaPhone 对象,支持对应的通话管理和控制请求 。
* PhoneAcount 在 Telecom 和 TeleService 两个系统应用间流转,保障了通话相关请求的正常晌应,过滤掉非法或不支持的通话请求消息 。

1. TeleService服务：TeleService 系统应用中提供了系统级服务和应用级服务
   1. 系统级服务：TeleService系统应用加载过程中将创建和发布的系统服务,如TelephonyManager、CarrierConfigManager、ImsRcsManager、SubscriptionManager等系统服务。 Android系统平台中的所有应用,均可通过ServiceManager.getService的方式获取服务的Binder代理对象,从而访问系统服务接口。
   2. 应用级服务：即 IConnectionService应用服务,通话业务中 Telecom 系统应用将通过绑定服务的方式访问TeleService系统应用提供的服务。

#### 小结

* **TeleService加载过程**：TeleService系统应用常驻内存, PhoneApp.onCreate 是该应用的加载入口, com.android.phone进程中则运行着以GsmCdmaPhone对象为中心的 Telephony业务模型；TeleService加载过程概括如下：加载Telephony 业务模型 (创 建以 GsmCdmaPhone对象为中心的核心业务对象)， 完成PhoneAccount初始化注册操作,发布TelephonyManager、CarrierConfigManager、ImsRcsManager、SubscriptionManager等系统服务。
* **Telephony业务模型**：GsmCdmaPhone与三大 Tracker： GsmCdmaCallTracker、 ServiceStateTracker、 DcTracker的组合关系以及门面设计模式)应用 。 GsmCdmaPhone 作为门面提供了 VoiceCall 语音通话、 ServiceState 网络服务和 DataCall 移动数据业务三大 Telephony能力。
* **PhoneAccount**： 作为Parcelable 类型,在 Telecom和 TeleService两个系统应用间跨进程传递，保障了通话相关请求的正常响应，过滤掉非法 、异常或不支持的通话请求消息。
* TeleService服务：包括系统服务PhoneInterFaceManager、SubscriptionController等和应用服务TelephonyConnectionService。