android o automotive简介

Last edited by **caoquanli** 1 month ago

Android O Automotive

层次结构

Car Api

Car API 中包含与Car 相关的各种 manager API，通过这些manager 对app提供 car service 所能提供的服务，主要实现在

```
/packages/services/Car/car-lib/
```

Car Service

Car Service 是android Automotive 的主要实现，在这一层会处理部分车辆业务逻辑， 并向下调用Vehicle Hal暴露的接口。主要实现在

```
/packages/services/Car/service/
```

Vehicle Hal

Vehicle Hal 主要用来定义车辆属性，并向上提供读写属性的接口，以及对属性变化注册监听的接口。 厂家需实现各自自定义的Vehicle Hal.主要实现在

```
/hardware/interfaces/automotive
```

实现

Android O 对 Vehicle 的支持除HAL层，基本都在/packages/service/Car 目录下，android 的car 架构可以理解为在android framework之上，利用android framework的android api 实现的更高级的业务框架。

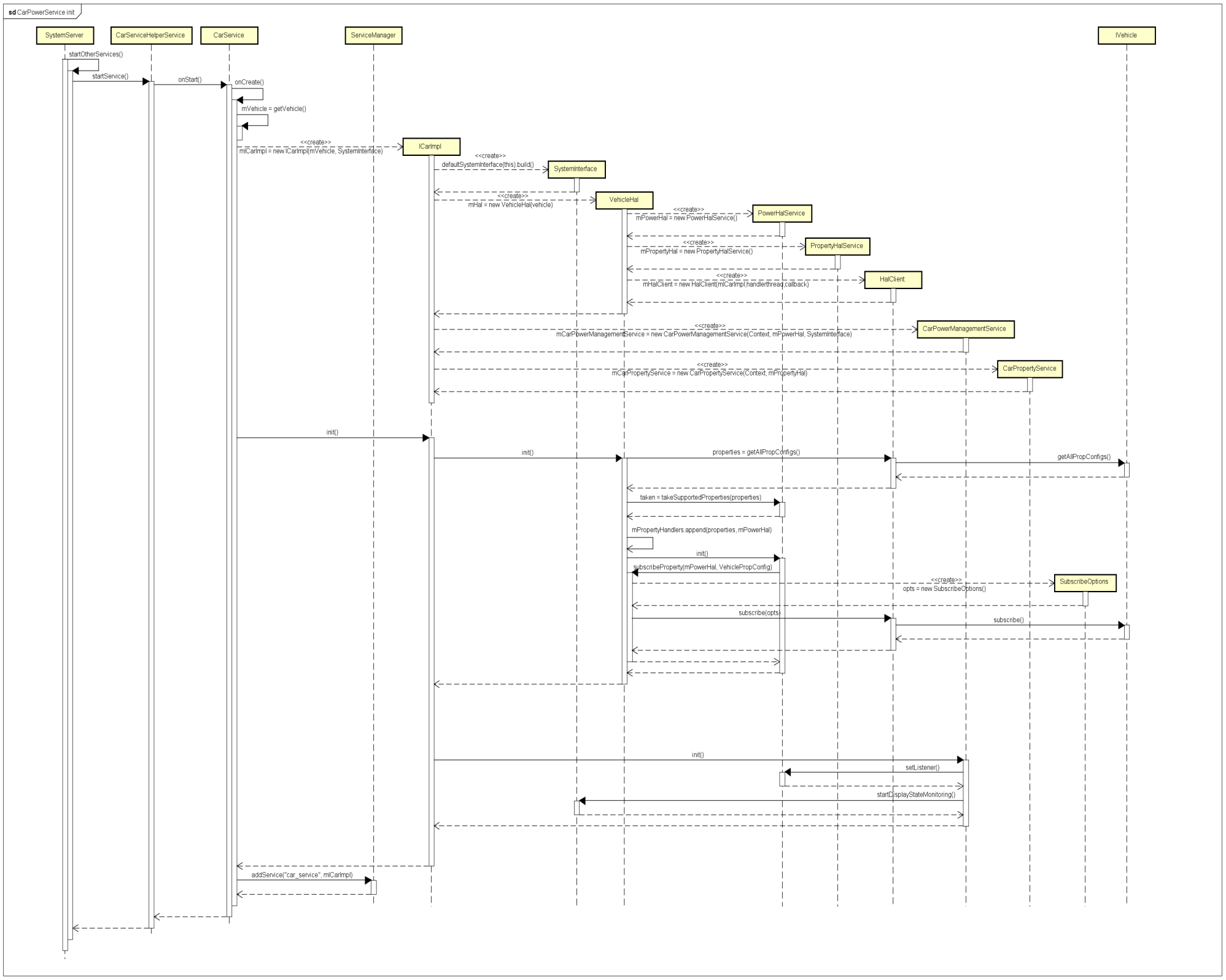
CarApi

Car类是Vehicle 的 API层的具体实现。Car 通过aidl最终与CarService进行进程间通信。 用户可使用如下代码使用Car的API:

```
 Car car = Car.createCar(getContext(), mConnectionListener);
// 2.连接Car service.
car.connect();
waitForConnection(DEFAULT_WAIT_TIMEOUT_MS);
// 3.通过名字获取对应的manager对象。
CarSensorManager carSensorManager =
(CarSensorManager) car.getCarManager(Car.SENSOR_SERVICE);
// 4.调用具体manager对象的方法
carSensorManager.isSensorSupported(CarSensorManager.SENSOR_TYPE_CAR_SPEED);
```

CarService init

时序图将以CarPowerService为例。



Step1

启动的入口 在SystemServer中startOtherServices。根据是否有AUTOMOTIVE的FEATURE来判断是否启动Carservice

```
if (mPackageManager.hasSystemFeature(PackageManager.FEATURE_AUTOMOTIVE)) {
                traceBeginAndSlog("StartCarServiceHelperService");
                mSystemServiceManager.startService(CAR_SERVICE_HELPER_SERVICE_CLASS);
                traceEnd();
        }
```

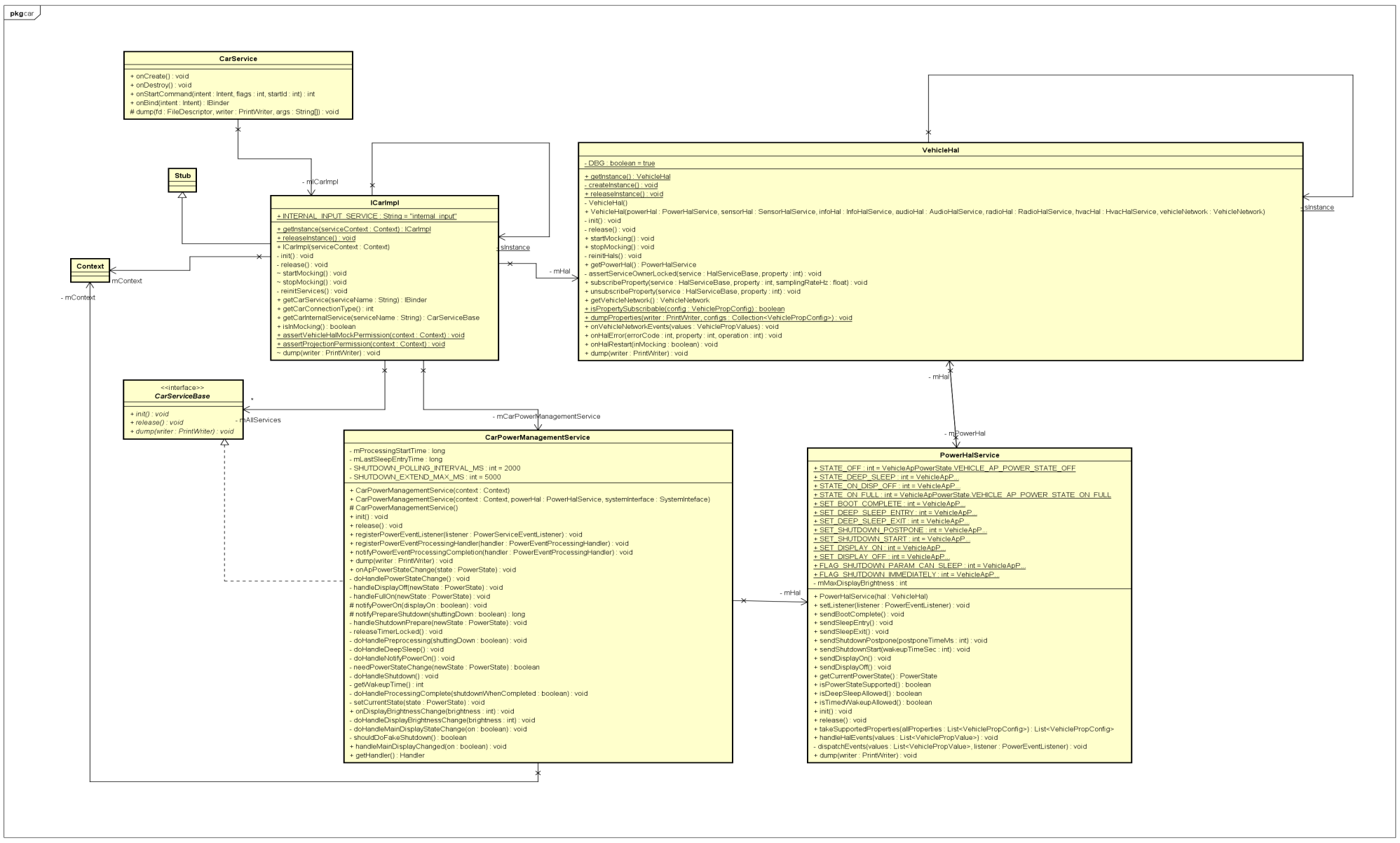
Step2

CarSerivceHelperService实则就是启动CarService

```
    public void onStart() {
        Intent intent = new Intent();
        intent.setPackage("com.android.car");
        intent.setAction(CAR_SERVICE_INTERFACE);
        if (!getContext().bindServiceAsUser(intent, mCarServiceConnection, Context.BIND_AUT
                UserHandle.SYSTEM)) {
            Slog.wtf(TAG, "cannot start car service");
        }
        System.loadLibrary("car-framework-service-jni");
    }
```

Step3

初始化mVehicle和mICarImpl，随后将car_service加入到ServiceManager中。



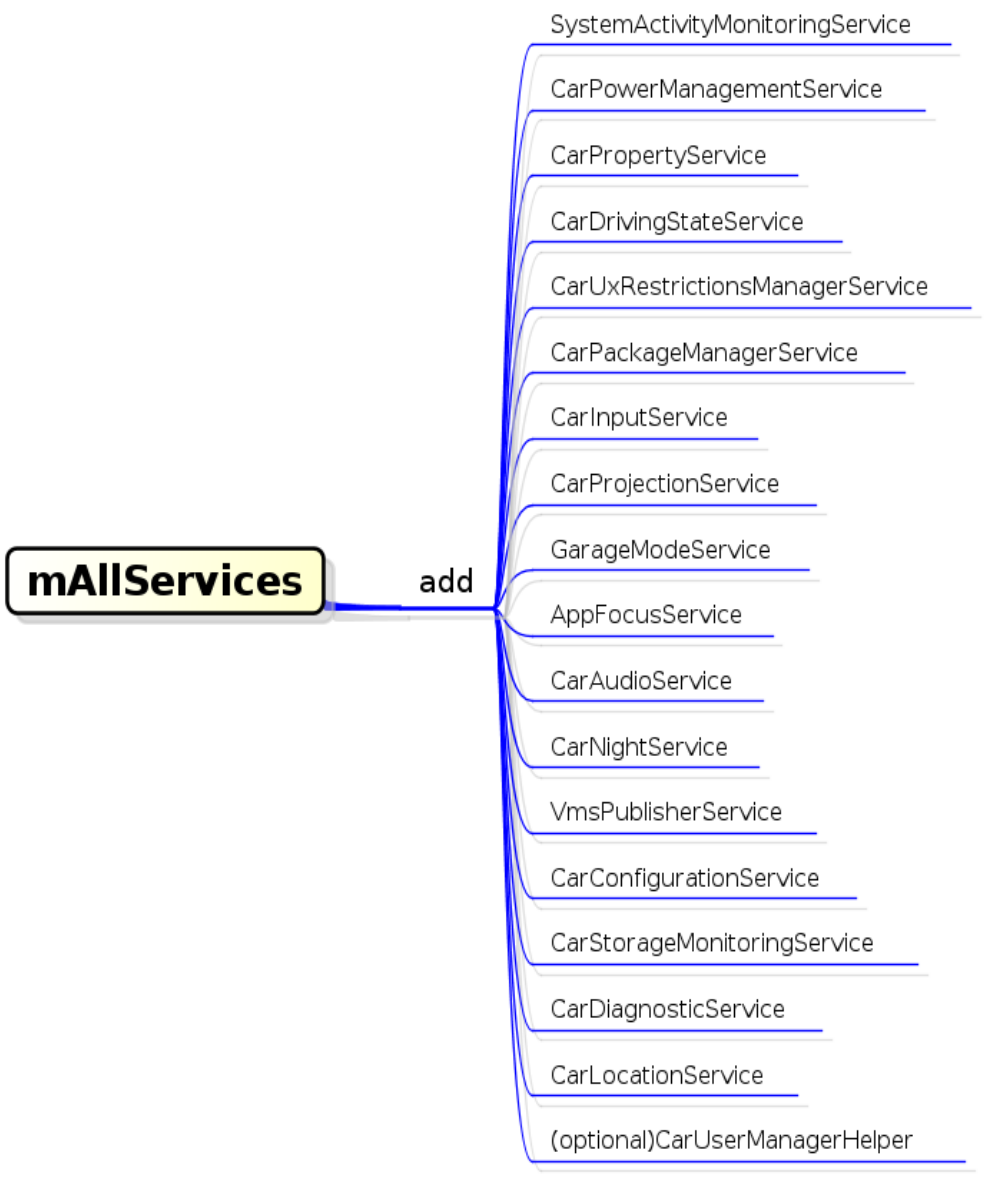
Step3.1 mVehicle

mVehicle是VehicleHal在客户端（针对Hal层来说的客户端）的代理对象，他的类型为IVehicle，其实质是一个IVehicle.Poxy对象。之后会通过mVehicle与VehicleHal进行跨进程通讯。

Step3.2 mIcarImpl

mIcarImpl是ICarImpl的实例，ICarImpl才是CarService中众多manager的真正创建者和管理者。

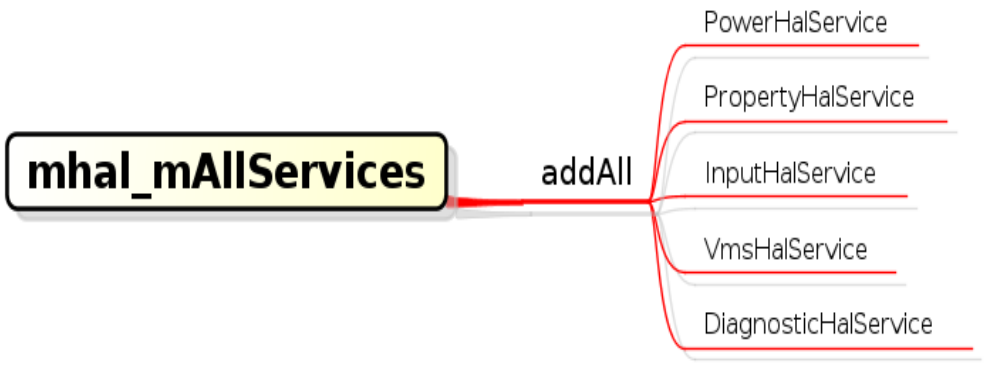
在ICarImpl中管理了众多的车辆服务，这些服务均实现了CarServiceBase接口，为方便管理和遍历所有service，将所有service的引用构建为一个数组。



Step3.3 mhal

对应的在VehicleHAL中管理着众多的HalService 这些HALService 通过VehicleHAL中的HALClient 与 HAL层通信，处理相应的属性变化，这些HALService 均继承自HALServiceBase。

mhal对象管理着各Service的HalService



例如：CarPowerManagementService 对应于 PowerHalService。

Step3.4

在初始化后将car_service加入到ServiceManager中。

整个init完成。