Desay API Description

|  |  |
| --- | --- |
| **Manager** | **接口信息** |
| **1.FordCarInfoManager(5)** | 车身信息相关：包含车型，vin码，油电类型，年款，屏幕类型，ESN,360相关，车辆颜色相关信息； |
| **2.FordCarHvacManager(8)** | 空调相关；包含AAR,空调开关，方向盘加热，内外循环，座椅加热，座椅通风等空调接口； |
| **3.FordCarLanDunAARManager(24)** | 空调相关的类：主要是蓝盾相关的接口； |
| **4.FordCarSensorManager(7)** | 车辆信息相关：包含交通信号灯,Lancher,单位的变化，速度，车门，车窗，胎温，胎压，轮胎状态，驾驶模式，挡位，发送机状态，总里程，剩余里程,车辆故障； |
| **5.FordCarVendorExtensionManager(6)** | 车辆其他的；白天黑夜模式，RocketsetUp，运输模式，开机动画，aar图标的状态，syncp，decode加解密，USB mount状态； |
| **6.FordEmergencyManager(14)** | 紧急救援EA相关的接口 |
| **7.FordAmbientLightManager(37)** | 动态氛围灯相关的接口； |
| **8.FordBtManager (15)** | 蓝牙相关的接口； |
| **9.FordCeDTEManager (13)** | cedte相关的接口； |
| **10.FordConfigManager(28)** | 配置位相关的接口； |
| **11.FordElectronicHorizonManager(19)** | EH相关的接口； |
| **12.FordEnhanceMemoryManager(11)** | EM相关的接口； |
| **13.FordEVChargeManager(12)** | EV相关的接口 |
| **14.FordFaceIdManager(27)和FordEVSCameraManager** | 人脸识别相关的接口（2种不同的算法） |
| **15.FordFragranceManager(31)** | 香氛相关的接口；（香氛相关的接口是需要联动的，用EM种香氛的接口） |
| **16.FordHardKeyManager（18）** | 硬按键相关的接口 |
| **17.FordIpptClientManager(26)** | IPPT相关功能的接口； |
| **18.FordLiftgateManager(35)** | 后备箱相关的接口； |
| **19.FordMultiContourSeatManager(30)** | 座椅按摩相关的接口； |
| **20.FordMultiscreenManager(23)** | 多屏相关得接口； |
| **21.FordSeatRelxPosManager(32)** | 座椅位置移动相关的接口； |
| **22.FordSoaManager(36)** | tcuesn相关的接口 |
| **23.FordSunroofManager(34)** | 天窗 遮阳棚相关的接口； |
| **24.FordThemeManager(20)** | 主题相关的接口； |
| **25.FordTSRCoDriverManager(33)** | TSR灯相关的接口； |
| **26.FordV2ILiteManager(29)** | V2I相关的接口； |
| **30** FordCommonUtil(40) | 判断是否是某个车型的接口 |
| **31** FordModeCardManager（42） | 时间 挡位方向，导航信号，副驾坐人，安全带等接口 |
| **32** FordConnectivityManager（44） | 5G流量统计接⼝ |
| **33 FordOffRoadManager** | **OffRoad相关的can信号的接口，OffRoad配置位相关的接口在FordConfigManager中** |

目录

[1 Mic抢占 44](#_Toc141445804)

[2 android.media.AudioManager.java 44](#_Toc141445805)

[2.1 requestAudioFocus(AudioFocusRequest focusRequest) 44](#_Toc141445806)

[2.2 abandonAudioFocusRequest(AudioFocusRequest focusRequest） 44](#_Toc141445807)

[2.3 setStreamVolume (int streamType, int index, int flags) 44](#_Toc141445808)

[2.4 List<AudioFocusInfo> getAudioFocusInfo(int type)获取所有的音源list： 45](#_Toc141445809)

[2.5 String getCarAudioType(AudioFocusInfo info) 45](#_Toc141445810)

[2.6 在线收音机的AudioSourceType： 45](#_Toc141445811)

[2.7 AudioAttributes setAttrShowDisplayId(AudioAttributes attr, int displayId)申请source的接口 45](#_Toc141445812)

[2.8 int getAttrShowDisplayId(AudioAttributes attr)获取source的接口 45](#_Toc141445813)

[3 平台界面跳转 45](#_Toc141445814)

[4 com.ford.base.FordBaseManager.java 45](#_Toc141445815)

[5 com.ford.car.FordCarInfoManager.java 46](#_Toc141445816)

[5.1 public static FordCarInfoManager getInstance(Context context) 46](#_Toc141445817)

[5.2 int getCarModel()获取当前车型。 46](#_Toc141445818)

[5.3 String getVinCode()获取Vin码。 47](#_Toc141445819)

[5.4 String getCarColor()获取车辆颜色 47](#_Toc141445820)

[5.5 String getESN()获取车机设备电子序列号。 47](#_Toc141445821)

[5.6 String getPCBANumber()获取车机PCBA零件号。 47](#_Toc141445822)

[5.7 String getAssemblyNumber()获取车机设备型号。 47](#_Toc141445823)

[5.8 String getVMCUVersion()获取VMCU版本号。 47](#_Toc141445824)

[5.9 String getSerialNo() 47](#_Toc141445825)

[5.10 String getSystemModel() 47](#_Toc141445826)

[5.11 String getSystemVersion() 47](#_Toc141445827)

[5.12 int getCarPowerType()获取车辆动力类型 47](#_Toc141445828)

[5.13 void setActiveStatus(int status)写激活状态。 47](#_Toc141445829)

[5.14 int getActiveStatus()读取激活状态。 47](#_Toc141445830)

[5.15 String getSystemType() 47](#_Toc141445831)

[5.16 String getCarType() 48](#_Toc141445832)

[5.17 String getCarStyle()获取车辆款式 48](#_Toc141445833)

[5.18 boolean has360Camera() 48](#_Toc141445834)

[5.19 String getCloudDataDir(Context context)返回值：/cert/data/cloud 48](#_Toc141445835)

[5.20 interface IProvisionStateResult { 48](#_Toc141445836)

[5.21 int get542CarModel()返回值； 48](#_Toc141445837)

[5.22 int getDisplayvariants() 48](#_Toc141445838)

[5.23 void setUsbSwitch(String usbSwitch) 48](#_Toc141445839)

[6 com.ford.car.FordCarVendorExtensionManager.java 49](#_Toc141445840)

[6.1 void startServiceAsUser(Intent intent) 49](#_Toc141445841)

[6.2 String getBootAnimIsStopped() 49](#_Toc141445842)

[6.3 void setNeedLoginCheck(String loginCheck) 49](#_Toc141445843)

[6.4 String getNeedLoginCheck() 49](#_Toc141445844)

[6.5 void setAuthResult(int authed) 49](#_Toc141445845)

[6.6 String getVolumeState(String mountPoint) ；USB得挂载情况 49](#_Toc141445846)

[参数 49](#_Toc141445847)

[返回：unknown 没有挂载上 mounted usb已挂载 49](#_Toc141445848)

[6.7 int getDayAndNightMode()获取白天黑夜模式; 49](#_Toc141445849)

[6.8 void registerDayAndNightModeListener(ISvDataBusDispatcher.Stub listener)注册白天黑夜模式的监听 50](#_Toc141445850)

[6.9 void unregisterDayAndNightModeListener(ISvDataBusDispatcher.Stub listener)反注册白天黑夜模式的监听 50](#_Toc141445851)

[6.10 int getPowerState()获取电源的模式 50](#_Toc141445852)

[6.11 void registerPowerStateListener(ISvDataBusDispatcher.Stub listener) *注册电源模式得回调*} 50](#_Toc141445853)

[6.12 void unregisterPowerStateListener(ISvDataBusDispatcher.Stub listener) *取消电源模式得回调*} 50](#_Toc141445854)

[6.13 ActivityManager.RunningTaskInfo getTopTaskOfDisplay(int displayId)获取 50](#_Toc141445855)

[6.14 interface IBootAnimationCompleted {void onBootAnimationCompleted(boolean state);}开机动画结束的回调 51](#_Toc141445856)

[6.15 void setAARState(int state)设置aar的状态 51](#_Toc141445857)

[6.16 int initialize\_syncp()syncp的接口 0:失败，1成功 51](#_Toc141445858)

[6.17 byte[] decode(byte[] decodeBuffer, int size) 51](#_Toc141445859)

[6.18 void connect(ConnectionCallback connection) 连接soa服务 51](#_Toc141445860)

[6.19 int sendProfileValueResponse(String transactionId, int commandStatus, ProfileErrorDetail profileErrorDetail) 51](#_Toc141445861)

[6.20 interface IPortableResponseListener { 51](#_Toc141445862)

[ProfileStructureInfo ：LabelType; //label数据类型 0：是整型，1：String类型  2：Double类型 3：Float 类型 4:  byte 类型，这里设定类型后，下面选择对应的Value值设置，比如 LabelType 设置2，就给DoubleValue设置值 51](#_Toc141445863)

[6.21 List<ProfileValueMap> getPortableData() 51](#_Toc141445864)

[6.22 int getWifiSwitchState()//获取wifi的开关状态 52](#_Toc141445865)

[6.23 int getWifiConnState()//获取wifi的连接状态 52](#_Toc141445866)

[6.24 interface IWifiSwitchState {//wifi开关状态的回调 52](#_Toc141445867)

[6.25 interface IWifiConnState {//wifi连接状态的回调 52](#_Toc141445868)

[6.26 int getDlnaSwitch()//dlna的总开关1：开， 0：关 52](#_Toc141445869)

[6.27 int getPhoneSwitch() 52](#_Toc141445870)

[6.28 interface IDlnaSwitch{//dlna总开关的回调 52](#_Toc141445871)

[6.29 interface IPhoneHotspotSwitchState{ 52](#_Toc141445872)

[6.30 int getStrPowerState //STR的状态； 52](#_Toc141445873)

[6.31 interface IStrPowerStateListener{ 53](#_Toc141445874)

[6.32 public void registerStrPowerStateListenerWithCompletion(DesayCommonManager. 53](#_Toc141445875)

[6.33 public void unregisterStrPowerStateListenerWithCompletion(DesayCommonManager. 53](#_Toc141445876)

[6.34 int getBtMusicDisplayState()//获取蓝牙音乐的显示/隐藏状态 53](#_Toc141445877)

[6.35 int getRadioDisplayState()//获取radio的显示隐藏状态 53](#_Toc141445878)

[6.36 int getSuixintingDisplayState() //获取随心听显示隐藏状态 54](#_Toc141445879)

[6.37 void setSuixintingDisplayState(int state)//设置随心听的显示隐藏状态 54](#_Toc141445880)

[6.38 interface IBtMusicDisplayStateListener {//蓝牙音乐显示隐藏状态的回调 54](#_Toc141445881)

[6.39 interface IRadioDisplayStateListener {//radio显示隐藏的回调 54](#_Toc141445882)

[6.40 interface ISuixintingDisplayStateListener {//随心听显示隐藏的回调 54](#_Toc141445883)

[7 com.ford.car.FordCarSensorManager.java 55](#_Toc141445884)

[7.1 public static FordCarSensorManager getInstance(Context context) 55](#_Toc141445885)

[7.2 int getCarSpeedUnit()获取车速单位。 55](#_Toc141445886)

[7.3 float getCarSpeed()获取车速。 55](#_Toc141445887)

[7.4 int getCarDriveMode()获取驾驶模式 55](#_Toc141445888)

[0x0 55](#_Toc141445889)

[SelDrvMde01 55](#_Toc141445890)

[0x1 55](#_Toc141445891)

[SelDrvMde02 55](#_Toc141445892)

[0x2 55](#_Toc141445893)

[SelDrvMde03 55](#_Toc141445894)

[0x3 55](#_Toc141445895)

[SelDrvMde04 55](#_Toc141445896)

[0x4 55](#_Toc141445897)

[SelDrvMde05 55](#_Toc141445898)

[0x5 55](#_Toc141445899)

[SelDrvMde06 55](#_Toc141445900)

[0x6 55](#_Toc141445901)

[SelDrvMde07 55](#_Toc141445902)

[0x7 55](#_Toc141445903)

[SelDrvMde08 55](#_Toc141445904)

[0x8 55](#_Toc141445905)

[SelDrvMde09 55](#_Toc141445906)

[0x9 55](#_Toc141445907)

[SelDrvMde10 55](#_Toc141445908)

[0xA 55](#_Toc141445909)

[SelDrvMde11 55](#_Toc141445910)

[0xB 55](#_Toc141445911)

[SelDrvMde12 55](#_Toc141445912)

[0xC 55](#_Toc141445913)

[SelDrvMde13 55](#_Toc141445914)

[0xD 55](#_Toc141445915)

[SelDrvMde14 55](#_Toc141445916)

[0xE 55](#_Toc141445917)

[SelDrvMde15 55](#_Toc141445918)

[0xF 55](#_Toc141445919)

[SelDrvMde16 55](#_Toc141445920)

[0x10 55](#_Toc141445921)

[SelDrvMde17 55](#_Toc141445922)

[0x11 55](#_Toc141445923)

[SelDrvMde18 55](#_Toc141445924)

[0x12 55](#_Toc141445925)

[SelDrvMde19 55](#_Toc141445926)

[0x13 55](#_Toc141445927)

[SelDrvMde20 55](#_Toc141445928)

[0x14 55](#_Toc141445929)

[SelDrvMde21 55](#_Toc141445930)

[0x15 55](#_Toc141445931)

[SelDrvMde22 55](#_Toc141445932)

[0x16 56](#_Toc141445933)

[SelDrvMde23 56](#_Toc141445934)

[0x17 56](#_Toc141445935)

[SelDrvMde24 56](#_Toc141445936)

[0x18 56](#_Toc141445937)

[SelDrvMde25 56](#_Toc141445938)

[0x19 56](#_Toc141445939)

[SelDrvMde26 56](#_Toc141445940)

[0x1A 56](#_Toc141445941)

[SelDrvMde27 56](#_Toc141445942)

[0x1B 56](#_Toc141445943)

[SelDrvMde28 56](#_Toc141445944)

[0x1C 56](#_Toc141445945)

[SelDrvMde29 56](#_Toc141445946)

[0x1D 56](#_Toc141445947)

[SelDrvMde30 56](#_Toc141445948)

[0x1E 56](#_Toc141445949)

[SelDrvMde31 56](#_Toc141445950)

[0x1F 56](#_Toc141445951)

[Faulty 56](#_Toc141445952)

[0x20-0xFF 56](#_Toc141445953)

[not update 56](#_Toc141445954)

[7.5 int getCarDoorOpenStatus(int doorPos)获取各位置的车门打开状态。 56](#_Toc141445955)

[0x0 56](#_Toc141445956)

[Closed 56](#_Toc141445957)

[0x1 56](#_Toc141445958)

[Ajar 56](#_Toc141445959)

[7.6 float getCarEnduranceMileage()获取车辆续航里程（电车） 56](#_Toc141445960)

[7.7 float getCarEndurancePercent()获取车辆续航百分比 56](#_Toc141445961)

[7.8 int getCarEnduranceWarnStatus()获取续航警告状态 56](#_Toc141445962)

[0x0 56](#_Toc141445963)

[Null 56](#_Toc141445964)

[0x1 56](#_Toc141445965)

[Thres20mi\_32km 56](#_Toc141445966)

[0x2 57](#_Toc141445967)

[Thres30mi\_48km 57](#_Toc141445968)

[0x3 57](#_Toc141445969)

[Thres50mi\_80km 57](#_Toc141445970)

[0x4 57](#_Toc141445971)

[Thres30km\_18mi 57](#_Toc141445972)

[0x5 57](#_Toc141445973)

[Thres50km\_31mi 57](#_Toc141445974)

[0x6 57](#_Toc141445975)

[Thres80km\_50mi 57](#_Toc141445976)

[0x7 57](#_Toc141445977)

[NotUsed 57](#_Toc141445978)

[7.9 float getCarFuelPercent()获取剩余油量百分比。 57](#_Toc141445979)

[7.10 int getCarFuelWarnStatus()获取油量告警状态 57](#_Toc141445980)

[0x0 57](#_Toc141445981)

[OK 57](#_Toc141445982)

[0x1 57](#_Toc141445983)

[Low 57](#_Toc141445984)

[0x2 57](#_Toc141445985)

[VeryLow 57](#_Toc141445986)

[0x3 57](#_Toc141445987)

[DteLevel1MyKey 57](#_Toc141445988)

[0x4 57](#_Toc141445989)

[DteLevel2NonMyKey 57](#_Toc141445990)

[0x5 57](#_Toc141445991)

[DteLevel3 57](#_Toc141445992)

[0x6 57](#_Toc141445993)

[DteLevel4 57](#_Toc141445994)

[0x7 57](#_Toc141445995)

[DteLevel5Lowest 57](#_Toc141445996)

[7.11 int getCarGear()获取车辆档位 57](#_Toc141445997)

[0 57](#_Toc141445998)

[Park 57](#_Toc141445999)

[1 57](#_Toc141446000)

[Reverse 57](#_Toc141446001)

[2 57](#_Toc141446002)

[Neutral 57](#_Toc141446003)

[3 57](#_Toc141446004)

[Drive 57](#_Toc141446005)

[4 57](#_Toc141446006)

[Sport\_DriveSport 57](#_Toc141446007)

[5 57](#_Toc141446008)

[Low 57](#_Toc141446009)

[6 57](#_Toc141446010)

[first 57](#_Toc141446011)

[7 58](#_Toc141446012)

[second 58](#_Toc141446013)

[8 58](#_Toc141446014)

[third 58](#_Toc141446015)

[9 58](#_Toc141446016)

[fourth 58](#_Toc141446017)

[0xA 58](#_Toc141446018)

[fifth 58](#_Toc141446019)

[0xB 58](#_Toc141446020)

[sixth 58](#_Toc141446021)

[0xC 58](#_Toc141446022)

[Undefined\_Treat\_as\_Fault 58](#_Toc141446023)

[0xD 58](#_Toc141446024)

[Undefined\_Treat\_as\_\_Fault1 58](#_Toc141446025)

[0xE 58](#_Toc141446026)

[Unknown\_Position 58](#_Toc141446027)

[0xF 58](#_Toc141446028)

[Fault 58](#_Toc141446029)

[0xFF 58](#_Toc141446030)

[not update 58](#_Toc141446031)

[7.12 int getCarMileageUnit()获取里程单位 58](#_Toc141446032)

[0 58](#_Toc141446033)

[TripComputer\_metric 公里 58](#_Toc141446034)

[1 58](#_Toc141446035)

[TripComputer\_imperial 英里 58](#_Toc141446036)

[7.13 float getCarTirePressure(int tirePos)获取胎压 58](#_Toc141446037)

[7.14 int getCarTirePressureUnit()获取胎压单位 58](#_Toc141446038)

[7.15 float getCarTireTemperature(int tirePos)获取胎温 58](#_Toc141446039)

[7.16 int getCarTireTemperatureUnit()获取胎温单位 59](#_Toc141446040)

[7.17 int getCarTireStatus(int tirePos)获取轮胎状态 59](#_Toc141446041)

[0 59](#_Toc141446042)

[Unknown 59](#_Toc141446043)

[1 59](#_Toc141446044)

[Normal 59](#_Toc141446045)

[2 59](#_Toc141446046)

[Low 59](#_Toc141446047)

[3 59](#_Toc141446048)

[Fault 59](#_Toc141446049)

[4 59](#_Toc141446050)

[Alert 59](#_Toc141446051)

[0xF 59](#_Toc141446052)

[Not\_Supported 59](#_Toc141446053)

[7.18 int getCarTireSupportResult()车辆是否支持获取胎温、胎压 59](#_Toc141446054)

[7.19 float getCarTotalMileage()获取车辆总里程 59](#_Toc141446055)

[7.20 int getCarWindowOpenStatus(int windowPos)获取车窗打开状态 60](#_Toc141446056)

[7.21 float getChargePercentage()电量百分比 60](#_Toc141446057)

[7.22 addListener(SensorChangeListener listener) 60](#_Toc141446058)

[7.23 removeListener(SensorChangeListener listener) 60](#_Toc141446059)

[7.24 clearListeners() 60](#_Toc141446060)

[7.25 float getOutCarTemperature()获取车外温度 返回值：温度的值 60](#_Toc141446061)

[7.26 回调接口 60](#_Toc141446062)

[7.27 sendNaviToTCU(Bundle bundle) 61](#_Toc141446063)

[7.28 void connect(ConnectionCallback connection) 连接服务 62](#_Toc141446064)

[7.29 void disConnect() 断开服务 62](#_Toc141446065)

[7.30 int getTemperatureUnit()获取温度单位can信号返回的对应的值如下： 62](#_Toc141446066)

[0 62](#_Toc141446067)

[Temperature\_deg\_c 62](#_Toc141446068)

[1 62](#_Toc141446069)

[Temperature\_deg\_f 62](#_Toc141446070)

[7.31 int getIgnition\_Status() 获取点火状态 can信号返回的值如下： 62](#_Toc141446071)

[7.32 float getBearing() 获取车的朝向 62](#_Toc141446072)

[7.33 void setDriveMode(int mode) 设置驾驶模式 62](#_Toc141446073)

[7.34 int getStabCtlBrkActv\_B\_Actl() 62](#_Toc141446074)

[7.35 int getTurnLghtRightOn\_B\_Stat() 62](#_Toc141446075)

[7.36 int getApedPos\_Pc\_ActlArb() 63](#_Toc141446076)

[7.37 int getStePinCompAnEst\_D\_Qf() 63](#_Toc141446077)

[7.38 int getStePinComp\_An\_Est() 63](#_Toc141446078)

[7.39 int getStopStrtDrvMde\_D\_Indic() 63](#_Toc141446079)

[7.40 int getWhlRotatRr\_No\_Cnt() 63](#_Toc141446080)

[7.41 int getWhlRotatRl\_No\_Cnt() 63](#_Toc141446081)

[7.42 int getBpedDrvAppl\_D\_Actl() 63](#_Toc141446082)

[7.43 int getGearLvrPos\_D\_Actl() 63](#_Toc141446083)

[7.44 int getTurnLghtRight\_D\_Rq() 63](#_Toc141446084)

[7.45 int getTurnLghtLeft\_D\_Rq() 63](#_Toc141446085)

[7.46 int[] getVehicleGGCCDataArr() 63](#_Toc141446086)

[7.47 float getFuelCarEnduranceMileage()油量剩余里程（油车） 64](#_Toc141446087)

[7.48 int getTeltalWarnData\_No\_Actl()防抱死系统故障; 64](#_Toc141446088)

[7.49 int getThe\_tire\_pressure\_detection\_system\_detects\_faults()胎压监测系统警告 64](#_Toc141446089)

[7.50 int getThe\_engine\_maintenance\_lamp\_is\_out\_of\_order() 发动机故障 64](#_Toc141446090)

[7.51 int getEngine\_overheat\_condition() 冷却液温度过高 64](#_Toc141446091)

[7.52 int getEngine\_oil\_pressure\_test() 机油压力低 64](#_Toc141446092)

[7.53 int getPower\_system\_failure()动力系统故障 64](#_Toc141446093)

[7.54 int getThe\_electric\_power\_steering\_system\_detects\_faults() 64](#_Toc141446094)

[7.55 int getSteep\_descent\_control\_system() 陡坡缓降系统故障 64](#_Toc141446095)

[7.56 int getFailure\_detection\_of\_ramp\_starting\_auxiliary\_system() 64](#_Toc141446096)

[7.57 int getFault\_detection\_by\_external\_light\_system() 照明系统故障 64](#_Toc141446097)

[7.58 int getWindshield\_washing\_liquid\_level()挡风玻璃洗涤液液位状态 64](#_Toc141446098)

[7.59 int getAll\_wheel\_drive\_system\_detects\_faults() 64](#_Toc141446099)

[7.60 int getDetection\_of\_stolen\_goods\_in\_air\_filter\_element() 65](#_Toc141446100)

[7.61 void skipToRoadRescueActivity() 65](#_Toc141446101)

[7.62 boolean initAudioOFFState AudioOFF初始状态 65](#_Toc141446102)

[7.63 void skipToVHA() 65](#_Toc141446103)

[7.64 驾驶模式 65](#_Toc141446104)

[7.65 车速 65](#_Toc141446105)

[7.66 总里程; 65](#_Toc141446106)

[7.67 里程单位 66](#_Toc141446107)

[7.68 油量 66](#_Toc141446108)

[7.69 档位 66](#_Toc141446109)

[7.70 车窗 66](#_Toc141446110)

[7.71 车门 67](#_Toc141446111)

[7.72 胎压单位 67](#_Toc141446112)

[7.73 胎压 67](#_Toc141446113)

[7.74 车外温度 68](#_Toc141446114)

[7.75 轮胎状态 68](#_Toc141446115)

[7.76 剩余里程（电池续航里程） 69](#_Toc141446116)

[7.77 温度单位 69](#_Toc141446117)

[7.78 胎温 69](#_Toc141446118)

[7.79 速度单位 70](#_Toc141446119)

[7.80 续航警告 70](#_Toc141446120)

[7.81 续航百分比 70](#_Toc141446121)

[7.82 点火状态 70](#_Toc141446122)

[7.83 油量百分比 70](#_Toc141446123)

[7.84 总里程 71](#_Toc141446124)

[7.85 VehicleGGCCData 71](#_Toc141446125)

[7.86 TurnLghtLeft\_D\_Rq 71](#_Toc141446126)

[7.87 TurnLghtRight\_D\_Rq 71](#_Toc141446127)

[7.88 GearLvrPos\_D\_Actl 71](#_Toc141446128)

[7.89 BpedDrvAppl\_D\_Actl 71](#_Toc141446129)

[7.90 WhlRotatRl\_No\_Cnt 72](#_Toc141446130)

[7.91 WhlRotatRr\_No\_Cnt 72](#_Toc141446131)

[7.92 StopStrtDrvMde\_D\_Indic 72](#_Toc141446132)

[7.93 StePinComp\_An\_Est 72](#_Toc141446133)

[7.94 StePinCompAnEst\_D\_Qf 72](#_Toc141446134)

[7.95 ApedPos\_Pc\_ActlArb 72](#_Toc141446135)

[7.96 TurnLghtRightOn\_B\_Stat 73](#_Toc141446136)

[7.97 StabCtlBrkActv\_B\_Actl 73](#_Toc141446137)

[7.98 油量剩余里程 73](#_Toc141446138)

[7.99 胎温单位 73](#_Toc141446139)

[7.100 防抱死系统故障; 73](#_Toc141446140)

[7.101 胎压监测系统警告; 74](#_Toc141446141)

[7.102 发动机故障 74](#_Toc141446142)

[7.103 冷却液温度过高; 74](#_Toc141446143)

[7.104 机油压力低; 74](#_Toc141446144)

[7.105 动力系统故障; 74](#_Toc141446145)

[7.106 电动转向故障; 75](#_Toc141446146)

[7.107 陡坡缓降系统故障; 75](#_Toc141446147)

[7.108 坡道起步辅助故障; 75](#_Toc141446148)

[7.109 照明系统故障; 75](#_Toc141446149)

[7.110 胎温单位(同温度单位) 75](#_Toc141446150)

[7.111 电量百分比 76](#_Toc141446151)

[7.112 挡风玻璃洗涤液液位状态 76](#_Toc141446152)

[0 76](#_Toc141446153)

[正常 76](#_Toc141446154)

[1 76](#_Toc141446155)

[低 76](#_Toc141446156)

[2 76](#_Toc141446157)

[正在检测中 76](#_Toc141446158)

[7.113 全轮驱动或四轮驱动系统检测故障 76](#_Toc141446159)

[0 76](#_Toc141446160)

[正常 76](#_Toc141446161)

[1 76](#_Toc141446162)

[故障告警 76](#_Toc141446163)

[2 76](#_Toc141446164)

[正在检测中 76](#_Toc141446165)

[7.114 空气滤清器滤芯赃物检测 76](#_Toc141446166)

[0 76](#_Toc141446167)

[正常 76](#_Toc141446168)

[1 77](#_Toc141446169)

[故障告警 77](#_Toc141446170)

[2 77](#_Toc141446171)

[正在检测中 77](#_Toc141446172)

[7.115 定义的宏： 77](#_Toc141446173)

[8 com.ford.car.FordCarHvacManager.java 79](#_Toc141446174)

[8.1 public static FordCarHvacManager getInstance(Context context)获取Manager单例。 79](#_Toc141446175)

[8.2 boolean isCarFrontClimateOpen()前排空调是否打开。 79](#_Toc141446176)

[8.3 boolean isCarRearClimateOpen()后排空调是否打开。 79](#_Toc141446177)

[8.4 boolean isSupportAAR()是否有AAR功能。 79](#_Toc141446178)

[8.5 int getCarClimateCycleMode()获取空调循环模式。 79](#_Toc141446179)

[Enabled\_Inactive 80](#_Toc141446180)

[0x0 80](#_Toc141446181)

[Active 80](#_Toc141446182)

[0x1 80](#_Toc141446183)

[Disabled 80](#_Toc141446184)

[0x2 80](#_Toc141446185)

[Unused 80](#_Toc141446186)

[0x3 80](#_Toc141446187)

[8.6 void setCarClimateCycleMode(int mode)设置空调循环模式。 80](#_Toc141446188)

[8.7 void setCarFrontClimateOnOff(boolean isOpen) 设置前排空调开关。 80](#_Toc141446189)

[8.8 void setCarRearClimateOnOff(boolean isOpen)设置后排空调开关。 80](#_Toc141446190)

[8.9 addListener(ClimateChangeListener listener)设置空调状态变化的监听器。 80](#_Toc141446191)

[8.10 removeListener(ClimateChangeListener listener)移除添加的listener。 81](#_Toc141446192)

[8.11 clearListeners()清空listener集合。 81](#_Toc141446193)

[8.12 int getNewAirRefresh()获取aar得开关状态 81](#_Toc141446194)

[8.13 int getGlobalProperty(int type)当前时刻的PM2.5值 81](#_Toc141446195)

[8.14 interface AARChangeListener 81](#_Toc141446196)

[8.15 int getGlobalProperty1(int type) X分钟前的PM2.5值 82](#_Toc141446197)

[8.16 void setGlobalProperty(int type, int value) 发送当前空气过滤状态发送当前空气过滤状态 82](#_Toc141446198)

[0x0 83](#_Toc141446199)

[NotKnown 83](#_Toc141446200)

[PmCabn\_D\_Stat (Msg:0x227 APIM\_Request\_Signals\_1) 83](#_Toc141446201)

[0x1 83](#_Toc141446202)

[FilteringOff 83](#_Toc141446203)

[0x2 83](#_Toc141446204)

[FilteringOn 83](#_Toc141446205)

[0x3 83](#_Toc141446206)

[FilteringComplete 83](#_Toc141446207)

[0xFF 83](#_Toc141446208)

[Not Update 83](#_Toc141446209)

[0x0 83](#_Toc141446210)

[NotKnown 83](#_Toc141446211)

[PmCabnLvl\_D\_Stat (Msg:0x227 APIM\_Request\_Signals\_1) 83](#_Toc141446212)

[0x1 83](#_Toc141446213)

[Level\_1\_Best 83](#_Toc141446214)

[0x2 83](#_Toc141446215)

[Level\_2 83](#_Toc141446216)

[0x3 83](#_Toc141446217)

[Level\_3 83](#_Toc141446218)

[0x4 83](#_Toc141446219)

[Level\_4 83](#_Toc141446220)

[0x5 83](#_Toc141446221)

[Level\_5 83](#_Toc141446222)

[0x6 83](#_Toc141446223)

[Level\_6\_Worst 83](#_Toc141446224)

[0x7 83](#_Toc141446225)

[NotUsed\_1 83](#_Toc141446226)

[0xFF 83](#_Toc141446227)

[Not Update 83](#_Toc141446228)

[8.17 void setNewAirRefresh()开启/关闭座舱新风按钮 83](#_Toc141446229)

[8.18 int getAcConfig()空调是否可用 83](#_Toc141446230)

[8.19 void onConnect(FordBaseManager.ConnectionCallback connection) 同步数据返回，需要先连接服务 84](#_Toc141446231)

[8.20 void onDisConnect()断开服务的连接 84](#_Toc141446232)

[8.21 interface INewAirRefresh{ 84](#_Toc141446233)

[8.22 void setAutoPress()设置前auto 84](#_Toc141446234)

[8.23 int getAutoButtState(int front\_AUTO) int front\_AUTO: 84](#_Toc141446235)

[8.24 interface IAutoButtState { void onAutoButtStateChange(int btn, int state);}auto的回调 84](#_Toc141446236)

[8.25 void setWindSpeed(int speed) 设置风量 85](#_Toc141446237)

[8.26 int getWindSpeed() 获取风量的值 85](#_Toc141446238)

[8.27 interface IWindSpeedChange { 85](#_Toc141446239)

[8.28 void setPreSeatHeatingGearSwitchStatus(int status) 设置前座椅的挡位关闭： 85](#_Toc141446240)

[8.29 void setRearSeatHeatingGearSwitchStatus(int status) 设置后座椅的挡位关闭： 85](#_Toc141446241)

[8.30 void setPreSeatHeatingSwitchStatus() 设置前座椅加热开关状态 86](#_Toc141446242)

[8.31 int getPreSeatHeatingSwitchStatus() 获取前座椅加热开关的状态 86](#_Toc141446243)

[8.32 interface IPreSeatHeatingSwitchStatusChange { 86](#_Toc141446244)

[8.33 void setRearSeatHeatingSwitchStatus() 设置后座椅加热的开关状态 86](#_Toc141446245)

[8.34 int getRearSeatHeatingSwitchStatus() 86](#_Toc141446246)

[8.35 interface IRearSeatHeatingSwitchStatusListener { 86](#_Toc141446247)

[8.36 void setSteeringWheelHeating() 设置方向盘加热的开关状态 86](#_Toc141446248)

[8.37 int getSteeringWheelHeating() 获取加热方向盘开关的状态 86](#_Toc141446249)

[8.38 elHeatingChange { 86](#_Toc141446250)

[8.39 void setDefrost() 设置除雾的状态 87](#_Toc141446251)

[8.40 int getDefrost() 获取除雾的状态 87](#_Toc141446252)

[8.41 interface IDefrostChange { 87](#_Toc141446253)

[8.42 void setLHS\_Cld\_Seat\_Pressed()主驾制冷 87](#_Toc141446254)

[8.43 int getLHS\_Cond\_Seat\_Status() 主驾制冷状态 87](#_Toc141446255)

[8.44 interface ILHS\_Cond\_Seat\_StatusChange { 87](#_Toc141446256)

[8.45 void setRHS\_Cld\_Seat\_Pressed()副驾制冷 87](#_Toc141446257)

[8.46 int getRHS\_Cond\_Seat\_Status() 副驾制冷的状态 87](#_Toc141446258)

[8.47 interface IRHS\_Cond\_Seat\_StatusChange { 87](#_Toc141446259)

[8.48 int getDualButtonStatus() 双开模式 87](#_Toc141446260)

[Enabled\_Inactive 87](#_Toc141446261)

[0x0 87](#_Toc141446262)

[Active 87](#_Toc141446263)

[0x1 87](#_Toc141446264)

[Disabled 87](#_Toc141446265)

[0x2 87](#_Toc141446266)

[Unused 87](#_Toc141446267)

[0x3 87](#_Toc141446268)

[8.49 void setNone\_Pressed() None\_Pressed开关 88](#_Toc141446269)

[8.50 void setPanel\_Pressed() 吹脸 88](#_Toc141446270)

[8.51 int getPanel\_Btn\_Stt() 吹脸的状态 88](#_Toc141446271)

[Enabled\_Inactive 88](#_Toc141446272)

[0x0 88](#_Toc141446273)

[Active 88](#_Toc141446274)

[0x1 88](#_Toc141446275)

[Disabled 88](#_Toc141446276)

[0x2 88](#_Toc141446277)

[Unused 88](#_Toc141446278)

[0x3 88](#_Toc141446279)

[8.52 interface IPanel\_Btn\_SttListener { 吹脸状态的回调 88](#_Toc141446280)

[8.53 void setPanel\_Floor\_Pressed() 吹脸+地板 88](#_Toc141446281)

[8.54 void setFloor\_Pressed() 地板 88](#_Toc141446282)

[8.55 int getFloor\_Pressed() 吹地的状态 88](#_Toc141446283)

[Enabled\_Inactive 88](#_Toc141446284)

[0x0 88](#_Toc141446285)

[Active 88](#_Toc141446286)

[0x1 88](#_Toc141446287)

[Disabled 88](#_Toc141446288)

[0x2 88](#_Toc141446289)

[Unused 88](#_Toc141446290)

[0x3 88](#_Toc141446291)

[8.56 interface IFloor\_PressedListener { 吹地的状态的回调 88](#_Toc141446292)

[8.57 setWindscreen\_Pressed（）前挡 88](#_Toc141446293)

[8.58 int getWindscreen\_Btn\_Stt() 挡风返回的状态 88](#_Toc141446294)

[Enabled\_Inactive 88](#_Toc141446295)

[0x0 88](#_Toc141446296)

[Active 89](#_Toc141446297)

[0x1 89](#_Toc141446298)

[Disabled 89](#_Toc141446299)

[0x2 89](#_Toc141446300)

[Unused 89](#_Toc141446301)

[0x3 89](#_Toc141446302)

[8.59 interface IWindscreen\_Btn\_SttListener { 挡风返回状态的回调 89](#_Toc141446303)

[8.60 void setFloor\_Ws\_Pressed() 地板+前挡 89](#_Toc141446304)

[8.61 void setAC\_Pressed() 制冷 89](#_Toc141446305)

[8.62 int getAC\_Btn\_Stt() 制冷返回的状态 89](#_Toc141446306)

[Enabled\_Inactive 89](#_Toc141446307)

[0x0 89](#_Toc141446308)

[Active 89](#_Toc141446309)

[0x1 89](#_Toc141446310)

[Disabled 89](#_Toc141446311)

[0x2 89](#_Toc141446312)

[Unused 89](#_Toc141446313)

[0x3 89](#_Toc141446314)

[8.63 interface IAC\_Btn\_SttListener { 制冷返回状态的回调 89](#_Toc141446315)

[8.64 void setLHS\_Temp\_Inc\_Pressed() 主温+ 90](#_Toc141446316)

[8.65 void setLHS\_Temp\_Dec\_Pressed() 主温- 90](#_Toc141446317)

[8.66 void setRHS\_Temp\_Inc\_Pressed() 副温+ 90](#_Toc141446318)

[8.67 void setRHS\_Temp\_Dec\_Pressed() 副温- 90](#_Toc141446319)

[8.68 void setBlwr\_Inc\_Pressed() 风量+ 90](#_Toc141446320)

[8.69 void setBlwr\_Dec\_Pressed() 风量- 90](#_Toc141446321)

[8.70 void setMax\_AC\_Pressed() 最大制冷 90](#_Toc141446322)

[8.71 int getMax\_AC\_Btn\_Stt() 最大制冷返回的状态 90](#_Toc141446323)

[Enabled\_Inactive 90](#_Toc141446324)

[0x0 90](#_Toc141446325)

[Active 90](#_Toc141446326)

[0x1 90](#_Toc141446327)

[Disabled 90](#_Toc141446328)

[0x2 90](#_Toc141446329)

[Unused 90](#_Toc141446330)

[0x3 90](#_Toc141446331)

[8.72 interface IMax\_AC\_Btn\_SttListener { 最大制冷的回值 90](#_Toc141446332)

[8.73 void setRear\_Defrost\_Pressed() 后除霜 90](#_Toc141446333)

[8.74 int getRear\_Defrost\_Btn\_Stt() 后除霜的状态 90](#_Toc141446334)

[Enabled\_Inactive 90](#_Toc141446335)

[0x0 90](#_Toc141446336)

[Active 90](#_Toc141446337)

[0x1 90](#_Toc141446338)

[Disabled 90](#_Toc141446339)

[0x2 90](#_Toc141446340)

[Unused 90](#_Toc141446341)

[0x3 90](#_Toc141446342)

[8.75 interface IRear\_Defrost\_Btn\_SttListener { 后除霜状态的回调 90](#_Toc141446343)

[8.76 void setMax\_Defrost\_Pressed() 最大除霜 91](#_Toc141446344)

[8.77 int getMax\_Defrost\_Btn\_Stt() 最大除霜的状态 91](#_Toc141446345)

[Enabled\_Inactive 91](#_Toc141446346)

[0x0 91](#_Toc141446347)

[Active 91](#_Toc141446348)

[0x1 91](#_Toc141446349)

[Disabled 91](#_Toc141446350)

[0x2 91](#_Toc141446351)

[Unused 91](#_Toc141446352)

[0x3 91](#_Toc141446353)

[8.78 interface IMax\_Defrost\_Btn\_SttListener { 最大除霜状态的回调 91](#_Toc141446354)

[8.79 float getAirConTemperature() 主驾的温度 91](#_Toc141446355)

[8.80 void setAirConTemperature(int temperature) 设置主 驾的温度 91](#_Toc141446356)

[0 91](#_Toc141446357)

[No\_Request 91](#_Toc141446358)

[1 91](#_Toc141446359)

[LO 91](#_Toc141446360)

[2 91](#_Toc141446361)

[15.5 91](#_Toc141446362)

[3 91](#_Toc141446363)

[16 91](#_Toc141446364)

[4 91](#_Toc141446365)

[16.5 91](#_Toc141446366)

[5 91](#_Toc141446367)

[17 91](#_Toc141446368)

[6 91](#_Toc141446369)

[17.5 91](#_Toc141446370)

[7 91](#_Toc141446371)

[18 91](#_Toc141446372)

[8 91](#_Toc141446373)

[18.5 91](#_Toc141446374)

[9 91](#_Toc141446375)

[19 91](#_Toc141446376)

[10 91](#_Toc141446377)

[19.5 91](#_Toc141446378)

[11 91](#_Toc141446379)

[20 91](#_Toc141446380)

[12 91](#_Toc141446381)

[20.5 91](#_Toc141446382)

[13 91](#_Toc141446383)

[21 91](#_Toc141446384)

[14 91](#_Toc141446385)

[21.5 91](#_Toc141446386)

[15 91](#_Toc141446387)

[22 91](#_Toc141446388)

[16 91](#_Toc141446389)

[22.5 91](#_Toc141446390)

[17 91](#_Toc141446391)

[23 91](#_Toc141446392)

[18 91](#_Toc141446393)

[23.5 91](#_Toc141446394)

[19 91](#_Toc141446395)

[24 91](#_Toc141446396)

[20 92](#_Toc141446397)

[24.5 92](#_Toc141446398)

[21 92](#_Toc141446399)

[25 92](#_Toc141446400)

[22 92](#_Toc141446401)

[25.5 92](#_Toc141446402)

[23 92](#_Toc141446403)

[26 92](#_Toc141446404)

[24 92](#_Toc141446405)

[26.5 92](#_Toc141446406)

[25 92](#_Toc141446407)

[27 92](#_Toc141446408)

[26 92](#_Toc141446409)

[27.5 92](#_Toc141446410)

[27 92](#_Toc141446411)

[28 92](#_Toc141446412)

[28 92](#_Toc141446413)

[28.5 92](#_Toc141446414)

[29 92](#_Toc141446415)

[29 92](#_Toc141446416)

[30 92](#_Toc141446417)

[29.5 92](#_Toc141446418)

[31 92](#_Toc141446419)

[HI 92](#_Toc141446420)

[8.81 interface IAirConTemperatureChange { 主驾温度的回调 92](#_Toc141446421)

[8.82 float getPsngrAirConTemperature() 副驾的温度 92](#_Toc141446422)

[8.83 void setPsngrAirConTemperature(int temperature) 设 置副驾的温度 92](#_Toc141446423)

[0x00 92](#_Toc141446424)

[No\_Request 92](#_Toc141446425)

[0x01 92](#_Toc141446426)

[LO 92](#_Toc141446427)

[0x02 92](#_Toc141446428)

[15\_5 92](#_Toc141446429)

[0x03 92](#_Toc141446430)

[16\_0 92](#_Toc141446431)

[0x04 92](#_Toc141446432)

[16\_5 92](#_Toc141446433)

[0x05 92](#_Toc141446434)

[17\_0 92](#_Toc141446435)

[0x06 92](#_Toc141446436)

[17\_5 92](#_Toc141446437)

[…… 92](#_Toc141446438)

[…… 92](#_Toc141446439)

[0x1E 92](#_Toc141446440)

[29\_5 92](#_Toc141446441)

[8.84 interface IPsngrAirConTemperatureChange { 副驾温度的回调 92](#_Toc141446442)

[8.85 int getLHS\_Cld\_Seat\_Btn\_Stt() 主驾座椅制冷的状态 93](#_Toc141446443)

[8.86 interface ILHS\_Cld\_Seat\_Btn\_SttListener { 主驾座椅制冷的回调 93](#_Toc141446444)

[8.87 int getRHS\_Cld\_Seat\_Btn\_Stt() 副驾座椅制冷的状态 93](#_Toc141446445)

[8.88 interface IRHS\_Cld\_Seat\_Btn\_SttListener{ 副驾座椅制冷的回调 93](#_Toc141446446)

[8.89 void setDualButtonPressed() 93](#_Toc141446447)

[8.90 interface IDualButtonStatusChange { 93](#_Toc141446448)

[Enabled\_Inactive 93](#_Toc141446449)

[0x0 93](#_Toc141446450)

[Active 93](#_Toc141446451)

[0x1 93](#_Toc141446452)

[Disabled 93](#_Toc141446453)

[0x2 93](#_Toc141446454)

[Unused 93](#_Toc141446455)

[0x3 93](#_Toc141446456)

[8.91 setAutoGradePress(int grade) Auto的3个等级 93](#_Toc141446457)

[9 com.ford.media.MediaEventManager 94](#_Toc141446458)

[9.1 USB广播 94](#_Toc141446459)

[9.2 public static FordMediaEventManager getInstance(Context context) 94](#_Toc141446460)

[9.3 void sendVRTips(String text) 94](#_Toc141446461)

[9.4 void sendID3Info(ID3Info id3Info) 94](#_Toc141446462)

[9.5 void sendPlayState(int state) 95](#_Toc141446463)

[9.6 void sendPlayMode(int mode) 95](#_Toc141446464)

[9.7 addListener(ActionListener listener) 95](#_Toc141446465)

[9.8 removeListener(ActionListener listener) 95](#_Toc141446466)

[9.9 clearListeners() 95](#_Toc141446467)

[9.10 回调接口 95](#_Toc141446468)

[10 com.ford.camera.EVSCameraManager 96](#_Toc141446469)

[11 com.ford.vendor.FordEnhanceMemoryManager 96](#_Toc141446470)

[11.1 void connect(ConnectionCallback connection)em的同步连接方法，先连接service才有数据返回 96](#_Toc141446471)

[11.2 void disConnect()断开相关service的连接 96](#_Toc141446472)

[11.3 byte[] getPersOptInSt() 96](#_Toc141446473)

[11.4 void sendInfotainmentRecall\_Rq(byte persIndex) 96](#_Toc141446474)

[11.5 interface IOnProfileActive { 97](#_Toc141446475)

[11.6 boolean getPaaKConnection\_St() 97](#_Toc141446476)

[11.7 interface IPaakConnectionChanged { 97](#_Toc141446477)

[11.8 void sendEnhancedMemory\_St(byte status) 97](#_Toc141446478)

[11.9 void sendEnMemProfilePairing\_Rq(byte persIndex, byte buttonPairing, byte keyPairing)驾驶员档案索引（1-4：用户创建的档案5：访客档案） 驾驶员档案关联智能手机钥匙 前提：智能手机钥匙已连接 97](#_Toc141446479)

[11.10 interface IEMButtonPairingSt { 97](#_Toc141446480)

[11.11 void sendFeature\_Rq(byte operation, int featureID, int configuration, byte persIndex) 98](#_Toc141446481)

[11.12 void sendInfotainmentPersStore\_Rq(byte persIndex) 98](#_Toc141446482)

[11.13 interface IInfotainmentPersStoreSt { 98](#_Toc141446483)

[11.14 void sendPersOptInSt(byte[] persStatus) 发送档案得选中状态 98](#_Toc141446484)

[11.15 interface IEnMemKeyPairingSt { 98](#_Toc141446485)

[11.16 interface IPersPhonePairingSt { 99](#_Toc141446486)

[11.17 interface IPersKeyPairingSt { 99](#_Toc141446487)

[11.18 interface ISeatAndMirror { 99](#_Toc141446488)

[11.19 boolean[] getPersPhonePairingSt() 99](#_Toc141446489)

[11.20 interface IOnGearChanged{ 99](#_Toc141446490)

[11.21 int getGear(); 100](#_Toc141446491)

[11.22 List<RadioStation> getCollectRadios() 100](#_Toc141446492)

[11.23 void setCollectRadios(List<RadioStation> radios) 100](#_Toc141446493)

[11.24 interface ICollectedRadiosChange{ 100](#_Toc141446494)

[11.25 List<RadioStation> getAutoSavedRadios(int radioBrand) 自动存储电台的读取，最多18个电台 radioBrand:0:AM 1FM 101](#_Toc141446495)

[11.26 void setAutoSavedRadios(int radioBrand,List<RadioStation> radios) 101](#_Toc141446496)

[11.27 interface IAutoSavedRadiosChange{ 101](#_Toc141446497)

[11.28 int getTheme() 当前主题得读取 主题代表的int :0 1 2 对应的如下：2是默认主题； 101](#_Toc141446498)

[11.29 void setTheme(int Index) 101](#_Toc141446499)

[11.30 interface IThemeChange{ 101](#_Toc141446500)

[11.31 int getDistanceUnit() 获取距离单位 102](#_Toc141446501)

[11.32 void setDistanceUnit(int unit) 设置距离单位 102](#_Toc141446502)

[11.33 interface IDistanceUnitChange{ 102](#_Toc141446503)

[11.34 int getTemperatureUnit() 获取温度单位 102](#_Toc141446504)

[11.35 void setTemperatureUnit(int unit) 设置温度单位 102](#_Toc141446505)

[11.36 interface ITemperatureUnitChange{ 102](#_Toc141446506)

[11.37 int getTyreUnit() 获取胎压单位，默认值kPa 103](#_Toc141446507)

[11.38 void setTyreUnit(int tyreUnit) 设置胎压单位 103](#_Toc141446508)

[11.39 interface ITyreUnitChange{ 103](#_Toc141446509)

[11.40 int getMeasureUnit() 获取测量单位，默认值是km&km/l 103](#_Toc141446510)

[11.41 void setMeasureUnit(int measureUnit ) 设置测量单位 103](#_Toc141446511)

[11.42 interface IMeasureUnitChange{ 103](#_Toc141446512)

[11.43 int getAmbientLampSwitch() 获取氛围灯开关 0是关 对应的can信号如下； 103](#_Toc141446513)

[11.44 void setAmbientLampSwitch(int switch) 设置氛围灯开关 104](#_Toc141446514)

[11.45 interface IAmbientLampSwitchChange{ 104](#_Toc141446515)

[11.46 int getAmbientColor() 获取氛围灯颜色 104](#_Toc141446516)

[11.47 void setAmbientColor(int color ) 设置氛围灯颜色 104](#_Toc141446517)

[11.48 interface IAmbientColorChange{ 104](#_Toc141446518)

[11.49 10.51 int getAmbientBrightness() 获取氛围灯亮度 104](#_Toc141446519)

[11.50 void setAmbientBrightness(int brightness) 设置氛围灯亮度 104](#_Toc141446520)

[11.51 interface IAmbientBrightnessChange{ 104](#_Toc141446521)

[11.52 boolean getReverseImageDelay() 获取倒车影像延迟 105](#_Toc141446522)

[11.53 void setReverseImageDelay(boolean delay) 设置倒车影像延迟 105](#_Toc141446523)

[11.54 interface IReverseImageDelayChange{ 105](#_Toc141446524)

[11.55 int getSpeedVolCompensation() 获取车速音量补偿设置 105](#_Toc141446525)

[11.56 void setSpeedVolCompensation(int compensation) 设置车速音量补偿设置 105](#_Toc141446526)

[11.57 interface ISpeedVolCompensationChange{ 105](#_Toc141446527)

[11.58 int getAirConTemperature() 获取空调温度 106](#_Toc141446528)

[11.59 void setAirConTemperature(int temperature) 设置空调温度 106](#_Toc141446529)

[11.60 interface IAirConTemperatureChange{ 106](#_Toc141446530)

[11.61 void setDriveMode(int mode) 设置驾驶模式 106](#_Toc141446531)

[11.62 interface IPersRecallCountSt { 106](#_Toc141446532)

[11.63 int getProfileActive() 106](#_Toc141446533)

[11.64 void setFreqBand(RadioStation radiostation) 设置最后一次的brand 106](#_Toc141446534)

[11.65 RadioStation getFreqBand()获取最后一次的brand 106](#_Toc141446535)

[11.66 interface IFreqBandChange{ 106](#_Toc141446536)

[11.67 float getPsngrAirConTemperature() 107](#_Toc141446537)

[11.68 void setPsngrAirConTemperature(int temperature) 107](#_Toc141446538)

[11.69 interface IPsngrAirConTemperatureChange { 107](#_Toc141446539)

[11.70 int getAmbientlightmode() 107](#_Toc141446540)

[11.71 interface IAmbientlightmodeChange { 107](#_Toc141446541)

[11.72 int[] getAmbientLightingMode() 107](#_Toc141446542)

[11.73 void setAmbientLightingMode(int[] mode) 108](#_Toc141446543)

[11.74 interface IAmbientLightingModeChange { 108](#_Toc141446544)

[11.75 int getAmbientLightingValue() 109](#_Toc141446545)

[11.76 public void setAmbientLightingValue(int value) 109](#_Toc141446546)

[11.77 public interface IAmbientLightingValueChange { 109](#_Toc141446547)

[11.78 public int[] getCarFragranceSetting() 109](#_Toc141446548)

[0 109](#_Toc141446549)

[0:未知的 109](#_Toc141446550)

[1-253 109](#_Toc141446551)

[1-253:正确的香氛类型ID 109](#_Toc141446552)

[254 110](#_Toc141446553)

[254:未认证的/无效的 110](#_Toc141446554)

[255 110](#_Toc141446555)

[255:未安装 110](#_Toc141446556)

[0 110](#_Toc141446557)

[0:未知的 110](#_Toc141446558)

[1-253 110](#_Toc141446559)

[1-253:正确的香氛类型ID 110](#_Toc141446560)

[254 110](#_Toc141446561)

[254:未认证的/无效的 110](#_Toc141446562)

[255 110](#_Toc141446563)

[255:未安装 110](#_Toc141446564)

[0 110](#_Toc141446565)

[0:未知的 110](#_Toc141446566)

[1-253 110](#_Toc141446567)

[1-253:正确的香氛类型ID 110](#_Toc141446568)

[254 110](#_Toc141446569)

[254:未认证的/无效的 110](#_Toc141446570)

[255 110](#_Toc141446571)

[255:未安装 110](#_Toc141446572)

[11.79 public void setCarFragranceSetting(int[] i) 110](#_Toc141446573)

[11.80 public interface ICarFragranceSetting { 111](#_Toc141446574)

[11.81 public int getInstrumentMainSetting() 111](#_Toc141446575)

[11.82 public void setInstrumentMainSetting(int index) 112](#_Toc141446576)

[11.83 public interface InstrumentMainSetting { 112](#_Toc141446577)

[11.84 void setFeat2ConfigActl(int type) 112](#_Toc141446578)

[11.85 void getFeat2ConfigIpc2Actl() 112](#_Toc141446579)

[11.86 interface IFeat2ConfigIpc2Actl { 112](#_Toc141446580)

[12 com.ford.vendor.FordEVChargeManager 112](#_Toc141446581)

[12.1 byte getPlugConnectStatus();获取车与充电桩的连接状态 112](#_Toc141446582)

[12.2 interface IPlugConnectStatus{ 112](#_Toc141446583)

[12.3 boolean isChargePlanPos();判断车辆位置是否在充电计划的位置 113](#_Toc141446584)

[12.4 IVehiclePosChange { 113](#_Toc141446585)

[12.5 void setChargPlanSwitch(boolean on); 113](#_Toc141446586)

[12.6 float getChargePercentage(); 113](#_Toc141446587)

[12.7 interface IChargeProcessChange{ 113](#_Toc141446588)

[12.8 byte getChrgStat(); 113](#_Toc141446589)

[12.9 interface IChrgStatChange{ 113](#_Toc141446590)

[12.10 int getRemainingMileage(); 114](#_Toc141446591)

[12.11 float getfullyChargeHours(); 114](#_Toc141446592)

[12.12 byte getChargeTargetPercent(); 114](#_Toc141446593)

[12.13 ChargeStartTime getChargeStartTime(); 114](#_Toc141446594)

[12.14 ChargeEndTime getChargeEndTime(); 114](#_Toc141446595)

[12.15 List<DepartAndComfort> getDepartAndComfort(); 114](#_Toc141446596)

[12.16 boolean isVehicleRun(); 115](#_Toc141446597)

[12.17 interface IVehicleRunChange{ 115](#_Toc141446598)

[12.18 List<ChargeLocation> getUnsavedChargeLocations(); 115](#_Toc141446599)

[12.19 void deleteUnSavedChargeLocation(int id) 115](#_Toc141446600)

[12.20 public interface IChargePlanRX 115](#_Toc141446601)

[12.21 void saveDepartAndComforts(List<DepartAndComfort> departAndComforts); 115](#_Toc141446602)

[12.22 interface IChargeAndDepartConflict{ 116](#_Toc141446603)

[12.23 void disConnection() 取消相关service的连接 116](#_Toc141446604)

[12.24 interface IRemainMileageListener{ 116](#_Toc141446605)

[12.25 byte getBattLowWarning(); 116](#_Toc141446606)

[12.26 interface IBattLowWarning{ 116](#_Toc141446607)

[12.27 void sendOnbChrgPrflUpdate\_B\_Rq(byte requesting) 117](#_Toc141446608)

[12.28 void sendOnbChrgLocIdTrgt\_No\_Rq(int locId) 117](#_Toc141446609)

[12.29 void setUnsavedLocationSettingFlag(boolean flag) 117](#_Toc141446610)

[12.30 byte getVehiclePosId(); 117](#_Toc141446611)

[12.31 interface IUnsavedAckRec { 117](#_Toc141446612)

[12.32 interface IChargeLocationInfoReq { 117](#_Toc141446613)

[12.33 void transmitLocationNamesList(int reqType, List<EvChargeLocationRecord> infos) { apim端下发昵称列表 118](#_Toc141446614)

[12.34 boolean getDepartAndComfortSwitch() 获取出发时间设置开关 118](#_Toc141446615)

[12.35 int getFastChargeEndMinutes(byte canType) *\* 获取快速充电用时 \* 当canType==0，使用信号FstChrgBulk\_T\_Est； \* 当canType==1，使用信号FstChrgCmplt\_T\_Est；* 118](#_Toc141446616)

[12.36 void sendUnsavedLocationSettingFlag(boolean flag) 发送未保存位置的flag 118](#_Toc141446617)

[12.37 void sendOnbChrgGoTUpdate\_B\_Rq(int request,int id) 118](#_Toc141446618)

[12.38 interface IFullyChargeHoursListener { 118](#_Toc141446619)

[12.39 interface IChargeTargetPercentListener { 118](#_Toc141446620)

[12.40 interface IChargeStartTimeListener { 119](#_Toc141446621)

[12.41 interface IChargeEndTimeListener { 119](#_Toc141446622)

[12.42 interface IDepartAndComfortListener { 119](#_Toc141446623)

[12.43 void connect(ConnectionCallback connection) 同步service的连接状态 120](#_Toc141446624)

[12.44 void disConnect() 断开service的连接 120](#_Toc141446625)

[12.45 int getDistanceUnit() 120](#_Toc141446626)

[12.46 interface IDistanceUnitChange { 120](#_Toc141446627)

[12.47 float getNxtUsgSocEst() 120](#_Toc141446628)

[12.48 List<ChargePlan> getChargePlan() 120](#_Toc141446629)

[12.49 List<DepartAndComfort> getDepartAndComforts() 120](#_Toc141446630)

[12.50 interface IUnSavedChargeLoaction { 120](#_Toc141446631)

[12.51 interface INextDepartAndComfort { 120](#_Toc141446632)

[12.52 void sendUnSavedLocationReq(int id, int updateRq, int deleteRq, int unsavedRq) 121](#_Toc141446633)

[12.53 void sendChargePlanReq(ChargePlan chargePlan) 保存充电计划 121](#_Toc141446634)

[12.54 int getNtfCtnConflictRq() 121](#_Toc141446635)

[12.55 int getNextDepartId() 121](#_Toc141446636)

[12.56 interface IIgnitionStChange { 121](#_Toc141446637)

[12.57 int getIgnitionStatus() 121](#_Toc141446638)

[12.58 float getRngPerChrgAvg\_L2\_Dsply()满电里程 122](#_Toc141446639)

[12.59 interface IRngPerChrgAvg\_L2\_Dsply{//满电里程 122](#_Toc141446640)

[12.60 sendDepartAndComforts(List<DepartAndComfort> departAndComforts 122](#_Toc141446641)

[12.61 int getBattTracLoMsgTxt\_D\_Rq() 低电量续航 122](#_Toc141446642)

[12.62 interface IBattTracLoMsgTxt\_D\_Rq{ 122](#_Toc141446643)

[12.63 int getFstChrgCmplt\_T\_Est() 充满100%完成时间 123](#_Toc141446644)

[0-1023min 123](#_Toc141446645)

[FstChrgCmplt\_T\_Est 0x238, GWM\_Send\_Signals\_18\_HS3 123](#_Toc141446646)

[12.64 interface IFstChrgCmplt\_T\_EstListener{ 123](#_Toc141446647)

[12.65 float getEstmChrgTimeHP\_St() 充满100%预计完成时间 大功率 123](#_Toc141446648)

[0 123](#_Toc141446649)

[0 123](#_Toc141446650)

[0x479(GWM\_HPCM\_17\_HS3) EstmChrgTimeHP\_St 2. 单位：小时 123](#_Toc141446651)

[1 123](#_Toc141446652)

[0.1 123](#_Toc141446653)

[2 123](#_Toc141446654)

[0.2 123](#_Toc141446655)

[… 123](#_Toc141446656)

[… 123](#_Toc141446657)

[239 123](#_Toc141446658)

[23.9 123](#_Toc141446659)

[240 123](#_Toc141446660)

[24 123](#_Toc141446661)

[241…254 123](#_Toc141446662)

[Reserved 123](#_Toc141446663)

[255 123](#_Toc141446664)

[Invalid 123](#_Toc141446665)

[12.66 interface IEstmChrgTimeHP\_StListener{ 123](#_Toc141446666)

[12.67 float getEstmChrgTimeLP\_St()充满100%预计完成时间 低功率 123](#_Toc141446667)

[0 123](#_Toc141446668)

[0 123](#_Toc141446669)

[0x479(GWM\_HPCM\_17\_HS3) EstmChrgTimeLP\_St 2. 单位：小时 123](#_Toc141446670)

[1 123](#_Toc141446671)

[0.1 123](#_Toc141446672)

[2 123](#_Toc141446673)

[0.2 123](#_Toc141446674)

[… 124](#_Toc141446675)

[… 124](#_Toc141446676)

[239 124](#_Toc141446677)

[23.9 124](#_Toc141446678)

[240 124](#_Toc141446679)

[24 124](#_Toc141446680)

[241…254 124](#_Toc141446681)

[Reserved 124](#_Toc141446682)

[255 124](#_Toc141446683)

[Invalid 124](#_Toc141446684)

[12.68 interface IEstmChrgTimeLP\_StListener{ 124](#_Toc141446685)

[13 com.ford.vendor.FordCeDTEManager 124](#_Toc141446686)

[13.1 void setTripInfoStructure(SoaCedteTripInfo soaCedteTripInfo)发送cedte的数据 124](#_Toc141446687)

[13.2 interface ISoaDTEListener { 124](#_Toc141446688)

[14 com.ford.ea.FordEmergencyManager 124](#_Toc141446689)

[14.1 void sendReqWithRsp(BaseReqMessage baseReqMessage) 124](#_Toc141446690)

[14.2 int getCrashCan() 获取获取车辆的Crash信号 125](#_Toc141446691)

[14.3 int getIgnKeyType\_D\_Actl() 获取车辆的mykey的使用信号 125](#_Toc141446692)

[14.4 *int getMyKey\_e911Override\_St()*获取车辆的mykey的override状态信号 125](#_Toc141446693)

[14.5 int getDrStatDrv\_B\_Actl()获取车辆开门信号DrStatDrv\_B\_Actl 125](#_Toc141446694)

[14.6 int getIgnition\_Status()获取车辆点火信号Ignition\_Status 125](#_Toc141446695)

[14.7 int getPOWER\_MODE()获取车辆电源状态POWER\_MODE的信号 125](#_Toc141446696)

[14.8 int getVEDS()获取车辆发生crash时的车辆的信息信号 125](#_Toc141446697)

[14.9 interface IRspCallback { 125](#_Toc141446698)

[14.10 void registerPowerStateListener(ISvDataBusDispatcher.Stub listener)监听电源模式的回调 126](#_Toc141446699)

[14.11 void unregisterPowerStateListener(ISvDataBusDispatcher.Stub listener取消电源模式的监听回调 126](#_Toc141446700)

[14.12 int getPsngrFrntDetct\_D\_Actl()副驾驶是否有人 126](#_Toc141446701)

[14.13 DIDRspBean getDidRspBean()弹窗对应的类 126](#_Toc141446702)

[15 com.ford.vendor.FordBtManager 126](#_Toc141446703)

[15.1 void initialize(Context context)启动蓝牙服务（这个必须要先初始化，不然下面的所有方法都没用） 126](#_Toc141446704)

[15.2 String getBTUserName()获取已连接用户名称 连接的蓝牙名称 126](#_Toc141446705)

[15.3 int getBluetoothConnectStatus()获取蓝牙连接状态 126](#_Toc141446706)

[15.4 int getMissCallsCount(String address)获取未接电话数目 126](#_Toc141446707)

[15.5 void connect(String address)连接蓝牙 126](#_Toc141446708)

[15.6 void dial(String address, String number)拨打电话 126](#_Toc141446709)

[15.7 void terminateCall(String address)挂断电话 126](#_Toc141446710)

[15.8 void connectAudio(String address) 连接语音/免提 127](#_Toc141446711)

[15.9 void disconnectAudio(String address) 断开语音/挂断 127](#_Toc141446712)

[15.10 List<SVDevice> getHfpConnectedDevicesList()获取连接设备信息 127](#_Toc141446713)

[15.11 interface IBtPbapLoadContactsListener { 127](#_Toc141446714)

[15.12 interface IBtHfpChangeListener { 128](#_Toc141446715)

[15.13 interface IBtMasterDeviceChangeListener { 129](#_Toc141446716)

[15.14 void unBindServive()释放资源 129](#_Toc141446717)

[15.15 String getMasterDeviceAddress()获取蓝牙地址 129](#_Toc141446718)

[15.16 List<com.desay\_svautomotive.commonlib.SVCallInfo> getCallInfoList(String address)通话状态集合 129](#_Toc141446719)

[15.17 获取蓝牙联系人 129](#_Toc141446720)

[15.18 SVDevice getMasterDevice()获取主设备的信息 129](#_Toc141446721)

[15.19 boolean getBluetoothEarphoneConnectStatus()蓝牙耳机是否连接接口 129](#_Toc141446722)

[15.20 interface IBluetoothEarphoneConnectStatusChange { 129](#_Toc141446723)

[15.21 interface IBluetoothConnectStatusChange { 129](#_Toc141446724)

[15.22 boolean playBtMusic() 蓝牙音乐播放 130](#_Toc141446725)

[15.23 boolean pauseBtMusic() 蓝牙音乐暂停 130](#_Toc141446726)

[15.24 interface IBtSwitchChangeListener{ 130](#_Toc141446727)

[15.25 boolean getIsOpen() 蓝牙开关是否打开 130](#_Toc141446728)

[15.26 setBluetoothSwitch(boolean isOpen)设置蓝牙的开关 true 开 false 关 130](#_Toc141446729)

[15.27 void setMasterDevice(SVDevice device)//设置主设备 130](#_Toc141446730)

[15.28 interface IPhoneCallStateChangeListener {//通话设备的集合 130](#_Toc141446731)

[15.29 boolean isPbapSyncing(String address)是的同步联系人 130](#_Toc141446732)

[15.30 void syncPbapFromPhone(String address, int type)//同步联系人 130](#_Toc141446733)

[15.31 void disconnect(String address)//断开蓝牙 130](#_Toc141446734)

[16 com.ford.audio. FordAudioManager 130](#_Toc141446735)

[16.1 public static final String CAR\_AUDIO\_TYPE\_EA= 130](#_Toc141446736)

[16.2 List<android.media.AudioFocusInfo> getAudioFocusInfoForDisplay(int type,int displayId) 131](#_Toc141446737)

[17 com.ford.audio.FordCommonManager 131](#_Toc141446738)

[17.1 void connect(ConnectionCallback connection) 启动服务，并同步将消息发送出去 131](#_Toc141446739)

[17.2 void disConnect()断开服务 131](#_Toc141446740)

[17.3 sendReqWithRsp(BaseReqMessage baseReqMessage) 131](#_Toc141446741)

[17.4 sendRequestNoRsp(BaseReqMessage baseReqMessage) 发送不带返回消息的接口 BaseReqMessage 救援相关的参数 131](#_Toc141446742)

[17.5 void setResponseListener(IResponseCallback responseListener)设置数据回调 131](#_Toc141446743)

[17.6 interface IResponseCallback { 131](#_Toc141446744)

[18 com.ford.vendor.FordHardKeyManager 132](#_Toc141446745)

[18.1 void registerKeyEvent(int AppID, IkeyEventListener listener); 根据相应的AppID注册物理按键 132](#_Toc141446746)

[18.2 void unregisterKeyEvent(int AppID, IkeyEventListener listener) 133](#_Toc141446747)

[18.3 interface IkeyEventListener 133](#_Toc141446748)

[19 com.ford.vendor.FordElectronicHorizonManager 135](#_Toc141446749)

[19.1 void onConnect(FordBaseManager.ConnectionCallback connection) 135](#_Toc141446750)

[19.2 void onDisConnect() 断开服务连接 136](#_Toc141446751)

[19.3 void sendAdasMapEH1(byte[] bytes) 发送 Adas EH1 数据内容 136](#_Toc141446752)

[19.4 void sendAdasMapEH2(byte[] bytes) 发送 Adas EH2 数据内容 136](#_Toc141446753)

[20 com.ford.vendor. FordThemeManager 136](#_Toc141446754)

[20.1 int getTheme() 当前主题得读取 主题int: 0 1 2 目前只有3种主题，对应的如下：2是默认主题； 136](#_Toc141446755)

[20.2 void setTheme( int Index)主题得设置 具体设置见20.1 136](#_Toc141446756)

[20.3 interface IThemeChange{ 136](#_Toc141446757)

[21 com.desaysdk.commonlib.manager. BtA2dpManager 137](#_Toc141446758)

[21.1 List<SVDevice> getA2dpConnectedDevicesList() 137](#_Toc141446759)

[22 intent.getLongExtra (“start\_time”,defaultvalue) 137](#_Toc141446760)

[获取开机启动lastsource appservice的时间。 137](#_Toc141446761)

[23 com.ford.vendor. FordMultiscreenManager 137](#_Toc141446762)

[23.1 void startResize() 分屏 137](#_Toc141446763)

[23.2 void stopResize()全屏 137](#_Toc141446764)

[23.3 void swapDisplay() 交换屏 137](#_Toc141446765)

[23.4 int getDriveMode()获取驾驶模式 137](#_Toc141446766)

[23.5 Display getDisplay(Context context) 封装的接口如下：传入当前Activity的context 137](#_Toc141446767)

[23.6 interface IDriveModeChange { 138](#_Toc141446768)

[23.7 void changeViewDisplay(View view)交换窗口 138](#_Toc141446769)

[23.8 int getDisplayForView(View view) 获取当前窗口view所在的displayId 138](#_Toc141446770)

[23.9 ActivityManager.RunningTaskInfo getTopRunningTaskByDisplayId(int displayid)获取栈顶task的接口（只有27寸有） 138](#_Toc141446771)

[23.10 void setSplitScreenAbility(String pkg, boolean enable) 138](#_Toc141446772)

[23.11 String whoDisAllowSplitScreen()获取是否有设置禁止分屏的应用包名 139](#_Toc141446773)

[23.12 void setSwapScreenEnableForPackage(String packagename,boolean enable) 139](#_Toc141446774)

[23.13 void setDriveMode(int driveMode) 139](#_Toc141446775)

[24 com.ford.car.FordCarLanDunAARManager 139](#_Toc141446776)

[24.1 int getFrontWindshieldHeatingState()获取前窗加热状态 139](#_Toc141446777)

[24.2 interface AARLanDunFrontWindshieldHeatingStateListener { 139](#_Toc141446778)

[24.3 int getRearWindshieldHeatingStatus()获取后窗加热状态 139](#_Toc141446779)

[24.4 interface AARLanDunRearWindsListener { 139](#_Toc141446780)

[24.5 void onRearWindshieldHeatingStatus(boolean state); 139](#_Toc141446781)

[24.6 int getAirddistModedr(int type)获取当前的气流模式 139](#_Toc141446782)

[24.7 interface AARLanDunAirddistModedrListener { 140](#_Toc141446783)

[24.8 int getCompressorRunStatus()获取当前， 压缩机运行状态 140](#_Toc141446784)

[24.9 interface AARCompressorStatusListener { 140](#_Toc141446785)

[24.10 int getUpStatus(获取UP状态 140](#_Toc141446786)

[24.11 interface AARLanDunUpStatusListener { 140](#_Toc141446787)

[24.12 void connect(ConnectionCallback connection)连接服务 141](#_Toc141446788)

[24.13 void disConnect()解除绑定 141](#_Toc141446789)

[25 com.ford.car. CarProperty(解释文档属性类) 141](#_Toc141446790)

[25.1 所有的车型 141](#_Toc141446791)

[25.2 542H,542L,不是542的; 141](#_Toc141446792)

[25.3 CX727的颜色 142](#_Toc141446793)

[25.4 车辆样式 142](#_Toc141446794)

[25.5 油电类型 142](#_Toc141446795)

[25.6 点火信号 142](#_Toc141446796)

[25.7 里程单位 143](#_Toc141446797)

[25.8 车速单位 143](#_Toc141446798)

[25.9 胎压单位 143](#_Toc141446799)

[25.10 挡位 143](#_Toc141446800)

[25.11 油量警告状态 144](#_Toc141446801)

[25.12 续航警告状态 144](#_Toc141446802)

[25.13 车窗打开状态 145](#_Toc141446803)

[25.14 车门开关状态 145](#_Toc141446804)

[25.15 驾驶模式 145](#_Toc141446805)

[25.16 CX727具体的颜色 147](#_Toc141446806)

[25.17 CD542的颜色 147](#_Toc141446807)

[25.18 U554的颜色 147](#_Toc141446808)

[25.19 P702的颜色 148](#_Toc141446809)

[25.20 U625的颜色 149](#_Toc141446810)

[25.21 U611的颜色 149](#_Toc141446811)

[25.22 D568MCA的颜色 149](#_Toc141446812)

[25.23 CX483的颜色 150](#_Toc141446813)

[25.24 CX482的颜色 150](#_Toc141446814)

[25.25 CD539MCA的颜色 151](#_Toc141446815)

[25.26 C519的颜色 151](#_Toc141446816)

[25.27 电源模式 152](#_Toc141446817)

[26 com.ford.vendor. FordIpptClientManager 153](#_Toc141446818)

[26.1 void connect()//连接服务 153](#_Toc141446819)

[26.2 void onDisConnect()//断开服务 153](#_Toc141446820)

[26.3 void cleanup() 153](#_Toc141446821)

[26.4 int registerClient() 153](#_Toc141446822)

[26.5 int unregisterClient() 153](#_Toc141446823)

[26.6 int sendTokenRequest(String transactionId, boolean policyValidation, List<MIpptKeyValueBean> configurations, List<String> scopes) 153](#_Toc141446824)

[26.7 int sendRevocationResponse(String revocationId) 153](#_Toc141446825)

[26.8 int sendRevocationResult(String revocationId, int result, String errorDescription) 153](#_Toc141446826)

[26.9 interface IIpptListener { 153](#_Toc141446827)

[27 com.ford.vendor. FordFaceIdManager 154](#_Toc141446828)

[27.1 void connect(ConnectionCallback connection) 连接服务 154](#_Toc141446829)

[27.2 void disConnect() 断开服务 154](#_Toc141446830)

[27.3 void sendRegister() 激活 154](#_Toc141446831)

[27.4 void sendRecognize() 人脸识别 154](#_Toc141446832)

[27.5 void sendUnbind() 解绑 154](#_Toc141446833)

[27.6 void sendUnbind(int faceId) 解绑 154](#_Toc141446834)

[27.7 int getFunctionResult() 获取功能执行结果 154](#_Toc141446835)

[27.8 interface IFunctionResult { 154](#_Toc141446836)

[27.9 int getQueryDsmcState() 获取 DSMC 功能的状态 154](#_Toc141446837)

[27.10 interface IDsmcStateChangeListener { 154](#_Toc141446838)

[27.11 int getDetectFaceResult() 获取检测人脸检测时状态结果 155](#_Toc141446839)

[27.12 interface IDetectFaceResult { 155](#_Toc141446840)

[27.13 int getRecognizeFaceResult() 获取检测人脸识别时状态结果 155](#_Toc141446841)

[27.14 interface IRecognizeFaceResult { 155](#_Toc141446842)

[27.15 long getFaceId() 获取识别成功的 faceId 155](#_Toc141446843)

[27.16 interface IFaceIdResult { 155](#_Toc141446844)

[27.17 int getChecksumFaceId() 获取 DSMC faceId 值 155](#_Toc141446845)

[27.18 interface IChecksumFaceIdResult { 155](#_Toc141446846)

[27.19 int getCameraState() 获取 camera 状态 155](#_Toc141446847)

[27.20 interface ICameraState { 155](#_Toc141446848)

[27.21 int getCameraDrvIRState() 获取 camera 状态 155](#_Toc141446849)

[27.22 interface ICameraIRState { 155](#_Toc141446850)

[27.23 int getCameraPassIRState() 获取 camera 状态 156](#_Toc141446851)

[27.24 interface ICameraPassIRState { 156](#_Toc141446852)

[27.25 void keepAwake() 156](#_Toc141446853)

[27.26 void idle() 156](#_Toc141446854)

[27.27 void clearData()清空数据 156](#_Toc141446855)

[28 com.ford.vendor. FordConfigManager 156](#_Toc141446856)

[28.1 byte getTSR() DE04 B6b7 156](#_Toc141446857)

[28.2 byte getPowerToTheBox() DE06 B3b7 156](#_Toc141446858)

[28.3 byte getARNavigation() DE04 B5b7 156](#_Toc141446859)

[28.4 byte getLVDSInCluster() DE07 B4b0 156](#_Toc141446860)

[28.5 byte getFacialRecognition() DE06 B5b7 156](#_Toc141446861)

[28.6 byte getCRMDSMC() DE07 B4b1 156](#_Toc141446862)

[28.7 byte getFreshAirCabin() DE03 B3b7 156](#_Toc141446863)

[28.8 byte getFuelType() DE04 B3b7 156](#_Toc141446864)

[28.9 byte getEVRangeRing() DE04 B6b3 156](#_Toc141446865)

[28.10 byte getAdvancedHUD()DE07 B2b2 156](#_Toc141446866)

[28.11 byte getSmart\_DSP()内外置功放 157](#_Toc141446867)

[28.12 byte getEnhanced\_Memory() 157](#_Toc141446868)

[28.13 byte getNavi\_Features() 157](#_Toc141446869)

[28.14 byte getADAS() 157](#_Toc141446870)

[28.15 byte getAmbientLight() 157](#_Toc141446871)

[28.16 byte getSelectableDriveModes() 157](#_Toc141446872)

[28.17 byte getRemoteStartSteeringWheel() 157](#_Toc141446873)

[28.18 byte getDigital\_Scent() 157](#_Toc141446874)

[28.19 byte getTPMS() 157](#_Toc141446875)

[28.20 byte getTPMS\_By\_Location() 157](#_Toc141446876)

[28.21 byte getSui\_Xin\_Pai() 157](#_Toc141446877)

[28.22 byte getRelax() 157](#_Toc141446878)

[28.23 byte getHeat\_cool\_seat() 158](#_Toc141446879)

[28.24 byte getHeated\_SW() 158](#_Toc141446880)

[28.25 byte getDSM() 158](#_Toc141446881)

[28.26 int getBT\_tuning 蓝牙Tuning 158](#_Toc141446882)

[28.27 byte getNavigation\_Prompt\_Protection 导航音量保护配置位 158](#_Toc141446883)

[28.28 byte getAPA() 158](#_Toc141446884)

[28.29 byte getLidget() 158](#_Toc141446885)

[28.30 byte getPicture\_gallery 159](#_Toc141446886)

[28.31 byte getFrunkSoftswitch() 159](#_Toc141446887)

[28.32 byte getPowerLiftgateControlFunction() 159](#_Toc141446888)

[28.33 byte getAEIS\_with\_Override() 159](#_Toc141446889)

[28.34 byte getDaytime\_Running\_Lamps\_Control\_Function() 159](#_Toc141446890)

[28.35 int getNavigation\_floating\_window() 159](#_Toc141446891)

[28.36 byte getProjectionModes() 159](#_Toc141446892)

[28.37 int getMode\_Card() 场景卡片 159](#_Toc141446893)

[28.38 int getCamp\_Mode() 露营模式 159](#_Toc141446894)

[28.39 int getPet\_Mode() 宠物模式 159](#_Toc141446895)

[28.40 int getBattery\_Save\_Mode() 省电模式 159](#_Toc141446896)

[28.41 int getRelax\_Mode() 舒享时分 160](#_Toc141446897)

[28.42 int getGuest\_Mode() 客人模式 160](#_Toc141446898)

[28.43 int getCommute\_Mode() 上下班模式 160](#_Toc141446899)

[28.44 int getGlamour\_Photo\_Studio\_Mode() 魅力摄影模式 160](#_Toc141446900)

[28.45 byte getSound\_Mode\_Stereo\_Surround() 方位选择 160](#_Toc141446901)

[28.46 byte getDSP\_feature\_Quantum\_Logic\_Surrounding() 音效模式 160](#_Toc141446902)

[29 com.ford.vendor.FordV2ILiteManager 161](#_Toc141446903)

[29.1 void onConnect(FordBaseManager.ConnectionCallback connection) *同步接口，等待连接后返回数据* 161](#_Toc141446904)

[29.2 void onDisConnect()*断开连接* 161](#_Toc141446905)

[29.3 void sendTrafficLight\_Data\_1(byte[] bytes) 161](#_Toc141446906)

[29.4 void sendTrafficLight\_Data\_2(byte[] bytes) 161](#_Toc141446907)

[30 com.ford.vendor.FordMultiContourSeatManager 161](#_Toc141446908)

[30.1 void connect(FordBaseManager.ConnectionCallback connection) 连接服务 161](#_Toc141446909)

[30.2 void disConnect() 断开服务连接 161](#_Toc141446910)

[30.3 byte getMCSeat()座椅按摩的配置位 161](#_Toc141446911)

[30.4 void setSeatFnDrv\_D\_RqPara1(int Module) 161](#_Toc141446912)

[30.5 void setSeatFnChngDrv2\_D\_RqPara2(int Module) 161](#_Toc141446913)

[30.6 void setSeatFnChngDrv2\_D\_RqPara3(int Module) 161](#_Toc141446914)

[30.7 void setSeatFnPsgr\_D\_RqPara4(int Module) 161](#_Toc141446915)

[30.8 void setSeatFnChngPsgr2\_D\_RqPara5(int Module) 161](#_Toc141446916)

[30.9 void setSeatMasgPsngr\_D\_RqPara6(int Module) 161](#_Toc141446917)

[30.10 int getSeatFnDrv\_D\_StatPara1() 主驾开关 等于7是开，非7就是关 161](#_Toc141446918)

[ 162](#_Toc141446919)

[30.11 int getSeatPDrv\_B\_StatPara2() 162](#_Toc141446920)

[30.12 int getSeatBlLoDrv\_Pc\_ActlPara3() 162](#_Toc141446921)

[30.13 int getSeatBlUpDrv\_Pc\_ActlPara4() 162](#_Toc141446922)

[30.14 int getSeatLmbrLoDrv\_Pc\_ActlPara5() 162](#_Toc141446923)

[30.15 int getSeatLmbrMidDrv\_Pc\_ActlPara6() 162](#_Toc141446924)

[30.16 int getSeatLmbrUpDrv\_Pc\_ActlPara7() 162](#_Toc141446925)

[30.17 int getSeatIntnsDrv\_D\_StatPara8() 162](#_Toc141446926)

[30.18 int getSeatMasgDrv\_D\_StatPara9() 162](#_Toc141446927)

[30.19 int getSeatSwtchDrv\_B\_StatPara10() 162](#_Toc141446928)

[30.20 int getSeatFnPsgr\_D\_StatPara11() 副驾开关，等于7就是开，非7就是关 162](#_Toc141446929)

[30.21 int getSeatPPsgr\_B\_StatPara12() 163](#_Toc141446930)

[30.22 int getSeatIntnsPsngr\_D\_StatPara13() 163](#_Toc141446931)

[30.23 int getSeatMasgPsngr\_D\_StatPara14() 163](#_Toc141446932)

[30.24 int getSeatSwtchPsgr\_B\_StatPara15() 163](#_Toc141446933)

[30.25 int getSeatBlLoPsgr\_Pc\_ActlPara16() 163](#_Toc141446934)

[30.26 int getSeatBlUpPsgr\_Pc\_ActlPara17() 163](#_Toc141446935)

[30.27 int getSeatLmbrLoPsgr\_Pc\_ActlPara18() 163](#_Toc141446936)

[30.28 int getSeatLmbrMidPsgr\_Pc\_ActlPara19() 163](#_Toc141446937)

[30.29 int getSeatLmbrUpPsgr\_Pc\_ActlPara20() 163](#_Toc141446938)

[30.30 int getSeatScrnPsngrOn\_B\_RqPara21() 163](#_Toc141446939)

[30.31 int getPsgrSeatUpdateFlag()//副驾 1 2在副驾位置； 163](#_Toc141446940)

[30.32 int getDriveSeatUpdateFlag()//主驾位置 除了副驾位置就是主驾位置 163](#_Toc141446941)

[30.33 void setReleaxSeateState(boolean isShow)是否控制releax true：控制 falas；不控制 163](#_Toc141446942)

[30.34 interface IMultiContourSeatListener { 163](#_Toc141446943)

[30.35 void setMasterDrivingMassage(int module1,int module2,int module3)设置主驾 165](#_Toc141446944)

[30.36 void setPassengerDrivingMassage(int module1,int module2,int module3)设置副驾 165](#_Toc141446945)

[31 com.ford.vendor.FordFragranceManager 165](#_Toc141446946)

[31.1 void setSurroundMode(int state) 设置音效 1.立体音 2.环绕音 165](#_Toc141446947)

[31.2 int getSurroundMode() 获取音效 1.立体音 2.环绕音 165](#_Toc141446948)

[31.3 interface ISurroundMode { 165](#_Toc141446949)

[31.4 void connect(ConnectionCallback connection) 165](#_Toc141446950)

[31.5 void disConnect()断开服务 165](#_Toc141446951)

[31.6 byte getDigitalScent()电子香氛得配置位 165](#_Toc141446952)

[31.7 void setAC\_1\_FGA\_OperationReq(int state)香氛开关设置 165](#_Toc141446953)

[31.8 void setAC\_1\_GenderSts(int sex)性别状态 166](#_Toc141446954)

[31.9 void setAC\_1\_FGAChanTypSelect(int channel)香氛通道选择 166](#_Toc141446955)

[31.10 void setAC\_1\_FGAIntensityReq(int density)香氛浓度设置 166](#_Toc141446956)

[31.11 void setAC\_1\_Time\_Year(int year)年份设置 167](#_Toc141446957)

[31.12 void setAC\_1\_Time\_Day(int day)年份设置 167](#_Toc141446958)

[31.13 int getFGA\_1\_FG\_Chan1Typ() 通道1香氛类型 167](#_Toc141446959)

[0 167](#_Toc141446960)

[0:未知的 167](#_Toc141446961)

[1-253 167](#_Toc141446962)

[1-253:正确的香氛类型ID 167](#_Toc141446963)

[254 167](#_Toc141446964)

[254:未认证的/无效的 167](#_Toc141446965)

[255 167](#_Toc141446966)

[255:未安装 167](#_Toc141446967)

[31.14 int getFGA\_1\_FG\_Chan2Typ() 通道2香氛类型 167](#_Toc141446968)

[0 167](#_Toc141446969)

[0:未知的 167](#_Toc141446970)

[1-253 167](#_Toc141446971)

[1-253:正确的香氛类型ID 167](#_Toc141446972)

[254 167](#_Toc141446973)

[254:未认证的/无效的 167](#_Toc141446974)

[255 167](#_Toc141446975)

[255:未安装 167](#_Toc141446976)

[31.15 int getFGA\_1\_FG\_Chan3Typ() 通道3香氛类型 168](#_Toc141446977)

[31.16 int getFGA\_1\_FG\_LifeRemainingChan1() 通道1香氛罐残余百分比 168](#_Toc141446978)

[31.17 int getFGA\_1\_FG\_LifeRemainingChan2() 通道2香氛罐残余百分比 168](#_Toc141446979)

[31.18 int getFGA\_1\_FG\_LifeRemainingChan3() 通道3香氛罐残余百分比 168](#_Toc141446980)

[31.19 int getFGA\_1\_FGACurrentdensity() 当前香氛工作浓度信息 168](#_Toc141446981)

[31.20 int getFGA\_1\_FGACurrentWorkCh() 当前香氛模块工作状态 169](#_Toc141446982)

[31.21 int getFGA\_3\_FGID1Overdue() 通道1香氛超期敬告 169](#_Toc141446983)

[31.22 int getFGA\_3\_FGID2Overdue() 通道2香氛超期敬告 169](#_Toc141446984)

[31.23 int getFGA\_3\_FGID3Overdue() 通道3香氛超期敬告 169](#_Toc141446985)

[31.24 int getFGA\_3\_FGAStatusSts() 香氛系统状态 170](#_Toc141446986)

[31.25 int getFGA\_3\_FRAGTempSts() 香氛温度状态 170](#_Toc141446987)

[31.26 int getFGA\_3\_FRAGPowerSupplySts() 香氛电压状态 170](#_Toc141446988)

[31.27 int getFGA\_3\_FRAGFanSts() 香氛风机状态 170](#_Toc141446989)

[31.28 int getFGA\_3\_FRAGUnKnownErr() 香氛未知错误 170](#_Toc141446990)

[31.29 int getFGA\_2\_HW\_Major() 硬件主版本号 171](#_Toc141446991)

[31.30 int getFGA\_2\_HW\_Minor() 硬件小版本号 171](#_Toc141446992)

[31.31 int getFGA\_2\_SW\_Major() 软件主版本号 171](#_Toc141446993)

[31.32 int getFGA\_2\_SW\_Minor() 软件主版本号 171](#_Toc141446994)

[31.33 interface IFragranceChangeListener { 171](#_Toc141446995)

[31.34 int getPsngrFrntDetct\_D\_Actl()副驾驶是否有人； 173](#_Toc141446996)

[31.35 interface IPsngrFrntDetctChange { void onPsngrFrntDetctChange(int psngrFrntDetct);} 173](#_Toc141446997)

[31.36 void setSoundStage(int dspSoundStage) 设置声音的方位 173](#_Toc141446998)

[31.37 int getSoundStage() 声音得方位获取 173](#_Toc141446999)

[31.38 interface ISoundStageChange { 174](#_Toc141447000)

[31.39 void setStreamVolume(int streamType, int progress) 设置音量 174](#_Toc141447001)

[31.40 int getStreamVolume(int streamType) 获取音量 174](#_Toc141447002)

[31.41 void setStreamVolume(int streamType, int progress)设置蓝牙音量（就是31.39，只是蓝牙耳机的type不一样，取值范围不一样，get就是31.40） 174](#_Toc141447003)

[32 com.ford.vendor.FordSeatRelxPosManager 175](#_Toc141447004)

[32.1 void connect(ConnectionCallback connection)连接服务 175](#_Toc141447005)

[32.2 void disConnect() 断开服务 175](#_Toc141447006)

[32.3 void sendRelaxActivate(int activate)触发 DSM 调节座椅到relax位置 175](#_Toc141447007)

[32.4 int getRelaxActivate() 获取座椅relax的位置 175](#_Toc141447008)

[0x0 175](#_Toc141447009)

[Fail 175](#_Toc141447010)

[0x1 175](#_Toc141447011)

[Success 175](#_Toc141447012)

[32.5 interface IRelaxActivateChange { 座椅relax位置的回调； 175](#_Toc141447013)

[0x0 175](#_Toc141447014)

[Fail 175](#_Toc141447015)

[0x1 175](#_Toc141447016)

[Success 175](#_Toc141447017)

[32.6 void sendRelaxDeactivate(int deactivate)触发 DSM 调节座椅到relax之前的位置 175](#_Toc141447018)

[32.7 int getRelaxDeactivate() 获取座椅relax之前的位置； 176](#_Toc141447019)

[0x0 176](#_Toc141447020)

[Fail 176](#_Toc141447021)

[0x1 176](#_Toc141447022)

[Success 176](#_Toc141447023)

[32.8 public interface IRelaxDeactivateChange { 176](#_Toc141447024)

[0x0 176](#_Toc141447025)

[Fail 176](#_Toc141447026)

[0x1 176](#_Toc141447027)

[Success 176](#_Toc141447028)

[32.9 void sendRelaxPosSet(int posset)保存用户自己设置的放松位置 176](#_Toc141447029)

[32.10 int getRelaxPosSet() 获取用户自己设置的放松位置 176](#_Toc141447030)

[0x0 176](#_Toc141447031)

[Fail 176](#_Toc141447032)

[0x1 176](#_Toc141447033)

[Success 176](#_Toc141447034)

[32.11 interface IRelaxPosSetChange { 176](#_Toc141447035)

[0x0 177](#_Toc141447036)

[Fail 177](#_Toc141447037)

[0x1 177](#_Toc141447038)

[Success 177](#_Toc141447039)

[33 com.ford.vendor.FordTSRCoDriverManager 177](#_Toc141447040)

[33.1 void connect(ConnectionCallback connection)连接服务 177](#_Toc141447041)

[33.2 void disConnect() 断开 177](#_Toc141447042)

[33.3 int getTrnlpcGear() 获取手动挡信息 177](#_Toc141447043)

[0xFF 177](#_Toc141447044)

[0x0 177](#_Toc141447045)

[Neutral 177](#_Toc141447046)

[0x1 177](#_Toc141447047)

[\_1st\_Gear 177](#_Toc141447048)

[0x2 177](#_Toc141447049)

[\_2nd\_Gear 177](#_Toc141447050)

[0x3 177](#_Toc141447051)

[\_3rd\_Gear 177](#_Toc141447052)

[0x4 177](#_Toc141447053)

[\_4th\_Gear 177](#_Toc141447054)

[0x5 177](#_Toc141447055)

[\_5th\_Gear 177](#_Toc141447056)

[0x6 177](#_Toc141447057)

[\_6th\_Gear 177](#_Toc141447058)

[0x7 177](#_Toc141447059)

[\_7th\_Gear 177](#_Toc141447060)

[0x8 177](#_Toc141447061)

[\_8th\_Gear 177](#_Toc141447062)

[0x9 177](#_Toc141447063)

[\_9th\_Gear 177](#_Toc141447064)

[0xA 177](#_Toc141447065)

[\_10th\_Gear 177](#_Toc141447066)

[0xB 177](#_Toc141447067)

[\_11th\_Gear 177](#_Toc141447068)

[0xC 177](#_Toc141447069)

[\_12th\_Gear 177](#_Toc141447070)

[0xD 177](#_Toc141447071)

[\_13th\_Gear 177](#_Toc141447072)

[0xE 177](#_Toc141447073)

[\_14th\_Gear 177](#_Toc141447074)

[0xF 177](#_Toc141447075)

[No\_Gear\_Selected 177](#_Toc141447076)

[33.4 interface ITrnlpcGearChange { 178](#_Toc141447077)

[33.5 boolean getFogFrontLightStatus() 前雾灯开关 178](#_Toc141447078)

[0x0 178](#_Toc141447079)

[OFF 178](#_Toc141447080)

[0x1 178](#_Toc141447081)

[ON 178](#_Toc141447082)

[33.6 interface IFogFrontLightStatusChange { 178](#_Toc141447083)

[0x0 178](#_Toc141447084)

[OFF 178](#_Toc141447085)

[0x1 178](#_Toc141447086)

[ON 178](#_Toc141447087)

[33.7 boolean getFogRearLightStatus() 后雾灯开关状态 178](#_Toc141447088)

[0x0 178](#_Toc141447089)

[OFF 178](#_Toc141447090)

[0x1 178](#_Toc141447091)

[ON 178](#_Toc141447092)

[33.8 interface IFogRearLightStatusChange { 178](#_Toc141447093)

[0x0 178](#_Toc141447094)

[OFF 178](#_Toc141447095)

[0x1 178](#_Toc141447096)

[ON 178](#_Toc141447097)

[33.9 int getParkLampStatus() 车灯开关 179](#_Toc141447098)

[0x0 179](#_Toc141447099)

[OFF 179](#_Toc141447100)

[0x1 179](#_Toc141447101)

[ON 179](#_Toc141447102)

[0x2 179](#_Toc141447103)

[Unknown 179](#_Toc141447104)

[0x3 179](#_Toc141447105)

[Invalid 179](#_Toc141447106)

[33.10 interface IParkLampStatusChange { 179](#_Toc141447107)

[0x0 179](#_Toc141447108)

[OFF 179](#_Toc141447109)

[0x1 179](#_Toc141447110)

[ON 179](#_Toc141447111)

[0x2 179](#_Toc141447112)

[Unknown 179](#_Toc141447113)

[0x3 179](#_Toc141447114)

[Invalid 179](#_Toc141447115)

[33.11 boolean getHeadLightHiOnStatusFromAhb()远光灯开关 179](#_Toc141447116)

[33.12 interface IHeadLightHiOnStatusFromAhbChange { 179](#_Toc141447117)

[33.13 boolean getHeadLightHiOnStatus() 远光灯开关 180](#_Toc141447118)

[33.14 interface IHeadLightHiOnStatusChange { 180](#_Toc141447119)

[33.15 boolean isAutoMaticHighBeam()是否是自动远光灯/自动前照灯 180](#_Toc141447120)

[33.16 int getTirePressSystemStatus() 胎压监测系统状态 180](#_Toc141447121)

[33.17 interface ITirePressSystemStatusChange { 180](#_Toc141447122)

[33.18 int getTsrMsgTxt()获取TSR信息 180](#_Toc141447123)

[33.19 interface ITsrMsgTxtChange { 180](#_Toc141447124)

[33.20 int getTsrOswWarnMsgTxt() 获取Osw警告信息 180](#_Toc141447125)

[33.21 interface ITsrOswWarnMsgTxtChange { 181](#_Toc141447126)

[33.22 int getTsrOvtStatusMsgTxt() 获取Ovt状态信息 181](#_Toc141447127)

[33.23 interface ITsrOvtStatusMsgTxtChange { 181](#_Toc141447128)

[33.24 int getTsrOvtMsgTxt() 获取Ovt信息 181](#_Toc141447129)

[33.25 interface ITsrOvtMsgTxtChange { 181](#_Toc141447130)

[33.26 int getPersCtaSplitView()PersCtaSplitView 181](#_Toc141447131)

[33.27 interface IPersCtaSplitViewChange { 181](#_Toc141447132)

[33.28 int getTsrStatusMsgTxt() 获取Tsr状态信息 181](#_Toc141447133)

[33.29 interface ITsrStatusMsgTxtChange { 181](#_Toc141447134)

[33.30 int getTsrVL1RstrcMsgTxt() 获取TSR VL1限制信息 182](#_Toc141447135)

[33.31 interface ITsrVL1RstrcMsgTxtChange { 182](#_Toc141447136)

[33.32 int getTsrVL1StatusMsgTxt() 获取TSR VL1状态信息 182](#_Toc141447137)

[33.33 interface ITsrVL1StatusMsgTxtChange { 182](#_Toc141447138)

[33.34 int getTsrVL2RstrcMsgTxt() 获取TSR VL2限制信息 182](#_Toc141447139)

[33.35 interface ITsrVL2RstrcMsgTxtChange { 182](#_Toc141447140)

[33.36 int getTsrVL2StatusMsgTxt() 获取TSR VL2状态信息 182](#_Toc141447141)

[33.37 interface ITsrVL2StatusMsgTxtChange { 182](#_Toc141447142)

[33.38 int getTsrVLim1MsgTxt()获取TSR VLIM1信息 182](#_Toc141447143)

[33.39 interface ITsrVLim1MsgTxtChange { 182](#_Toc141447144)

[33.40 int getTsrVLim2MsgTxt()获取TSR VLIM2信息 183](#_Toc141447145)

[33.41 interface ITsrVLim2MsgTxtChange { 183](#_Toc141447146)

[33.42 int getTsrVIUnitMsgTxt() 获取TSR VI单位信息 183](#_Toc141447147)

[33.43 interface ITsrVIUnitMsgTxtChange { 183](#_Toc141447148)

[33.44 int getTsrVI1PrmntMsgTxt()获取TSR VI1常驻信息 183](#_Toc141447149)

[33.45 interface ITsrVI1PrmntMsgTxtChange { 183](#_Toc141447150)

[33.46 int getTsrOvtkMsgTxt2()获取TSR Ovtk信息 183](#_Toc141447151)

[33.47 interface ITsrOvtkMsgTxt2Change { 183](#_Toc141447152)

[33.48 int getTsrVL1RstrcMsgTxt2() 获取TSR TVL1限制信息 183](#_Toc141447153)

[33.49 interface ITsrVL1RstrcMsgTxt2Change { 184](#_Toc141447154)

[33.50 boolean getWwaWarn() WwaWarn 184](#_Toc141447155)

[33.51 interface IWwaWarnChange { 184](#_Toc141447156)

[33.52 int getTsrVL2RstrcMsgTxt2()获取TSR VL2限制信息 184](#_Toc141447157)

[33.53 interface ITsrVL2RstrcMsgTxt2Change { 184](#_Toc141447158)

[33.54 int getTsrVL2PrmntMsgTxt()获取TSR VL2 常驻信息 184](#_Toc141447159)

[33.55 interface ITsrVL2PrmntMsgTxtChange { 184](#_Toc141447160)

[33.56 int getLaSysOffStatus() LaSysOffStatus 184](#_Toc141447161)

[33.57 interface ILaSysOffStatusChange { 184](#_Toc141447162)

[33.58 void setCarDoorOpen() 设置门为打开 185](#_Toc141447163)

[33.59 void setCarDoorClose() 设置门为关闭 185](#_Toc141447164)

[33.60 void setHeadlightOpen()设置大灯为打开 185](#_Toc141447165)

[33.61 void setHeadlightClose()设置大灯为关闭 185](#_Toc141447166)

[33.62 void getCarDoorSwitchState() 查询门得状态 185](#_Toc141447167)

[33.63 void getHeadlightSwitchState() 查询大灯得状态 185](#_Toc141447168)

[33.64 interface ICarDoorSwitchStateChange { 185](#_Toc141447169)

[33.65 interface IHeadlightSwitchStateChange { 185](#_Toc141447170)

[33.66 int getEngineState() 发动机状态 185](#_Toc141447171)

[0 185](#_Toc141447172)

[EngOff 185](#_Toc141447173)

[1 185](#_Toc141447174)

[EngOn 185](#_Toc141447175)

[2 185](#_Toc141447176)

[EngAutoStopped 185](#_Toc141447177)

[3 185](#_Toc141447178)

[NotUsed 185](#_Toc141447179)

[33.67 interface IEngineStateChange { 185](#_Toc141447180)

[33.68 int getClusterSpeed() 获取仪表展示速度 186](#_Toc141447181)

[33.69 interface IClusterSpeed { 186](#_Toc141447182)

[33.70 int getAhbcStat() 自动远光灯得获取 186](#_Toc141447183)

[0x0 186](#_Toc141447184)

[OFF 186](#_Toc141447185)

[0x1 186](#_Toc141447186)

[ON 186](#_Toc141447187)

[33.71 interface IAhbcStatChanged { 186](#_Toc141447188)

[33.72 int getAhbStat() 获取自动远光灯得状态 186](#_Toc141447189)

[0x0 186](#_Toc141447190)

[OFF 186](#_Toc141447191)

[0x1 186](#_Toc141447192)

[ON 186](#_Toc141447193)

[33.73 interface IAhbStatChanged { 186](#_Toc141447194)

[33.74 void setCtrStkFeat2ConfigActl(int type) 设置的状态 186](#_Toc141447195)

[33.75 void getFeatConfigIpc2Actl()查询状态 186](#_Toc141447196)

[33.76 interface IFeatConfigIpc2ActlChange { 186](#_Toc141447197)

[33.77 void setCtrStkFeat2ConfigActlCommon(int featureid, int type) 187](#_Toc141447198)

[33.78 void getFeatConfigIpc2ActlCommon(int featureid) 187](#_Toc141447199)

[33.79 interface IFeatConfigIpc2ActlCommonChange { 187](#_Toc141447200)

[33.80 int getbsBattSOC() 187](#_Toc141447201)

[33.81 interface IBsBattSOCChange{ 187](#_Toc141447202)

[34 com.ford.vendor.FordSunroofManager 188](#_Toc141447203)

[34.1 void connect(ConnectionCallback connection)连接服务 188](#_Toc141447204)

[34.2 void disConnect() 断开服务 188](#_Toc141447205)

[34.3 void setSunroofControl(int TurnAsstSwtch\_D\_Stat)设置天窗的状态 188](#_Toc141447206)

[34.4 void setSunshadeControl(int TurnAsstSwtch\_D\_Stat) 设置遮阳板的状态 188](#_Toc141447207)

[34.5 int getSunroofDSPLStatusAntiPinch()天窗的状态 188](#_Toc141447208)

[34.6 int getSunroofDSPLStatusPosition() 天窗的移动位置 188](#_Toc141447209)

[34.7 int getSunroofDSPLStatusMovement() 天窗的移动状态 188](#_Toc141447210)

[34.8 int getSunroofDSPLStatusAntiPinchSS()遮阳板的状态 188](#_Toc141447211)

[34.9 int getSunroofDSPLStatusPositionSS() 遮阳板的移动位置 188](#_Toc141447212)

[34.10 int getSunroofDSPLStatusMovementSS() 遮阳板的移动状态 188](#_Toc141447213)

[34.11 interface ISunroofStateChange { 188](#_Toc141447214)

[35 com.ford.vendor.FordLiftgateManager 189](#_Toc141447215)

[35.1 void connect(ConnectionCallback connection) 连接服务 189](#_Toc141447216)

[35.2 void disConnect() 断开服务 189](#_Toc141447217)

[35.3 void setLiftgateRelaseCounter(int Counter) 打开后备箱, 参数从1到6循环 189](#_Toc141447218)

[**Param2:LiftgateReleaseCounter\_Rq** 189](#_Toc141447219)

[**Type** 189](#_Toc141447220)

[**Range** 189](#_Toc141447221)

[**Value** 189](#_Toc141447222)

[**功能设置** 189](#_Toc141447223)

[**Can信号名** 189](#_Toc141447224)

[**U8** 189](#_Toc141447225)

[**0** 189](#_Toc141447226)

[**0xFF** 189](#_Toc141447227)

[**0-7** 189](#_Toc141447228)

[**null** 189](#_Toc141447229)

[**BecRleas\_No\_RqMnu (msg:0x32B APIM\_Data** 189](#_Toc141447230)

[35.4 int getLiftgateAjarSt() 后备箱状态 189](#_Toc141447231)

[Param7:LiftgateAjar\_St 189](#_Toc141447232)

[Type 189](#_Toc141447233)

[Range 189](#_Toc141447234)

[Value 189](#_Toc141447235)

[功能设置 189](#_Toc141447236)

[Can信号名 189](#_Toc141447237)

[U8 189](#_Toc141447238)

[0 189](#_Toc141447239)

[0xFF 189](#_Toc141447240)

[0x0 189](#_Toc141447241)

[Closed 189](#_Toc141447242)

[DrStatInnrTgate\_B\_Actl (msg:0x3B2 BodyInfo\_3\_HS3 189](#_Toc141447243)

[0x1 189](#_Toc141447244)

[Ajar 189](#_Toc141447245)

[35.5 interface ILiftgateAjarSt { 189](#_Toc141447246)

[36 com.ford.vendor. FordSoaManager 190](#_Toc141447247)

[36.1 void connect(ConnectionCallback connection) 连接服务 190](#_Toc141447248)

[36.2 String getTcuEsn() 获取tcuesn 190](#_Toc141447249)

[36.3 String getTcuIccId() 获取tcuIccid 190](#_Toc141447250)

[36.4 interface ITcuBezelDiagnosticDataChange { 190](#_Toc141447251)

[37 com.ford.vendor.FordAmbientLightManager 190](#_Toc141447252)

[37.1 void connect(ConnectionCallback connection)连接服务 190](#_Toc141447253)

[37.2 void disConnect()断开服务 190](#_Toc141447254)

[37.3 void setAmbL\_ALM\_Set(int ambL\_ALM\_Set) 氛围灯开关 190](#_Toc141447255)

[0x00 190](#_Toc141447256)

[invalid 190](#_Toc141447257)

[0x01 190](#_Toc141447258)

[close 190](#_Toc141447259)

[0x02 190](#_Toc141447260)

[open 190](#_Toc141447261)

[37.4 void setAmbL\_Color\_Mode(int ambL\_Color\_Mode) 氛围灯模式 190](#_Toc141447262)

[0x00 190](#_Toc141447263)

[invalid 190](#_Toc141447264)

[0x01 190](#_Toc141447265)

[static 190](#_Toc141447266)

[0x02 190](#_Toc141447267)

[dynamic 190](#_Toc141447268)

[0x03 190](#_Toc141447269)

[customize 190](#_Toc141447270)

[0x04 190](#_Toc141447271)

[music 190](#_Toc141447272)

[37.5 void setAmbL\_DrvMde\_D\_Rq(int ambL\_DrvMde\_D\_Rq)驾驶模式设置 191](#_Toc141447273)

[0x00 191](#_Toc141447274)

[manual 191](#_Toc141447275)

[0x01 191](#_Toc141447276)

[auto 191](#_Toc141447277)

[37.6 void setAmbL\_Dynamic\_Color(int ambL\_Dynamic\_Color)动态模式的颜色设置 191](#_Toc141447278)

[0x0-0x7F 191](#_Toc141447279)

[0-127 191](#_Toc141447280)

[37.7 void setAmbL\_Main\_Intensity\_Set(int ambL\_Main\_Intensity\_Set)主亮度设置； 191](#_Toc141447281)

[0x0-0x64 191](#_Toc141447282)

[0-100(%) 191](#_Toc141447283)

[37.8 void setAmbL\_Music\_Frq\_Level(int ambL\_Music\_Frq\_Level)音乐频率； 191](#_Toc141447284)

[0x0 191](#_Toc141447285)

[0-63hz 191](#_Toc141447286)

[0x1 191](#_Toc141447287)

[64-127hz 191](#_Toc141447288)

[0x2 191](#_Toc141447289)

[128-255hz 191](#_Toc141447290)

[0x3 191](#_Toc141447291)

[256-511hz 191](#_Toc141447292)

[0x4 191](#_Toc141447293)

[512-1023hz 191](#_Toc141447294)

[0x5 191](#_Toc141447295)

[1023-2047hz 191](#_Toc141447296)

[0x6 191](#_Toc141447297)

[2048-4095hz 191](#_Toc141447298)

[0x7 191](#_Toc141447299)

[4096-8191hz 191](#_Toc141447300)

[0x8 191](#_Toc141447301)

[8192-16383hz 191](#_Toc141447302)

[37.9 void setAmbL\_Music\_Range\_Level(int ambL\_Music\_Range\_Level)音乐范围： 191](#_Toc141447303)

[37.10 void setAmbL\_Static\_ColorValue\_Set(int ambL\_Static\_ColorValue\_Set) 静态颜色设置： 192](#_Toc141447304)

[37.11 void setAmbL\_CustomizeType\_Set(int ambL\_CustomizeType\_Set)个性化设置： 192](#_Toc141447305)

[37.12 void setAmbL\_Door\_Color\_Value(int ambL\_Door\_Color\_Value)DOOR的氛围灯颜色： 192](#_Toc141447306)

[37.13 void setAmbL\_Door\_Intensity\_Value(int ambL\_Door\_Intensity\_Value)DOOR的氛围灯亮度 192](#_Toc141447307)

[37.14 void setAmbL\_Door\_Swtich(int ambL\_Door\_Swtich) Door的氛围灯开关： 192](#_Toc141447308)

[37.15 void setAmbL\_Foot\_Color\_Value(int ambL\_Foot\_Color\_Value)FOOT的氛围灯颜色： 193](#_Toc141447309)

[37.16 void setAmbL\_Foot\_Intensity\_Value(int ambL\_Foot\_Intensity\_Value) Foot的氛围灯亮度； 193](#_Toc141447310)

[37.17 void setAmbL\_Foot\_Swtich(int ambL\_Foot\_Swtich) Foot的氛围灯开关； 193](#_Toc141447311)

[37.18 void setAmbL\_IP\_Color\_Value(int ambL\_IP\_Color\_Value) IP的氛围灯颜色： 193](#_Toc141447312)

[37.19 void setAmbL\_IP\_Intensity\_Value(int ambL\_IP\_Intensity\_Value)IP的氛围灯亮度; 193](#_Toc141447313)

[37.20 void setAmbL\_IP\_Swtich(int ambL\_IP\_Swtich) IP的氛围灯开关： 193](#_Toc141447314)

[37.21 void setRearLight\_Animation\_Rq(int rearLight\_Animation\_Rq) 尾灯动画设置： 194](#_Toc141447315)

[37.22 int getAmbL\_ALM\_Set() 氛围灯开关 194](#_Toc141447316)

[37.23 int getAmbL\_Color\_Mode() 氛围灯模式： 194](#_Toc141447317)

[37.24 int getAmbL\_DrvMde\_D\_Rq() 驾驶模式设置： 194](#_Toc141447318)

[37.25 int getAmbL\_Dynamic\_Color() 动态模式的颜色; 194](#_Toc141447319)

[37.26 int getAmbL\_Main\_Intensity\_Set() 主亮度设置 195](#_Toc141447320)

[37.27 int getAmbL\_Music\_Frq\_Level()音乐频率： 195](#_Toc141447321)

[37.28 int getAmbL\_Music\_Range\_Level()音乐范围； 195](#_Toc141447322)

[37.29 int getAmbL\_Static\_ColorValue\_Set() 静态颜色： 195](#_Toc141447323)

[37.30 int getAmbL\_CustomizeType\_Set() 个性化设置： 196](#_Toc141447324)

[37.31 int getAmbL\_Door\_Color\_Value() Door的氛围灯颜色； 196](#_Toc141447325)

[37.32 int getAmbL\_Door\_Intensity\_Value() Door的氛围灯亮度： 196](#_Toc141447326)

[37.33 int getAmbL\_Door\_Swtich()Door的氛围灯开关： 196](#_Toc141447327)

[37.34 int getAmbL\_Foot\_Color\_Value() Foot的氛围灯颜色； 196](#_Toc141447328)

[37.35 int getAmbL\_Foot\_Intensity\_Value() Foot的氛围灯亮度； 196](#_Toc141447329)

[37.36 int getAmbL\_Foot\_Swtich() Foot的氛围灯开关： 197](#_Toc141447330)

[37.37 int getAmbL\_IP\_Color\_Value()IP的氛围灯颜色； 197](#_Toc141447331)

[37.38 int getAmbL\_IP\_Intensity\_Value()IP的氛围灯亮度 197](#_Toc141447332)

[37.39 int getAmbL\_IP\_Swtich() IP的氛围灯开关； 197](#_Toc141447333)

[37.40 int getRearLight\_Animation\_Rq() 尾灯动画设置： 197](#_Toc141447334)

[37.41 interface IAmbientLightChange { 197](#_Toc141447335)

[38 com.ford.vendor.FordARPOCManager 199](#_Toc141447336)

[38.1 void connect(ConnectionCallback connection) 199](#_Toc141447337)

[38.2 void disConnect() 199](#_Toc141447338)

[38.3 int getLaActvStats\_D2\_Dsply() 199](#_Toc141447339)

[38.4 int getCcStat\_D\_Actl() 200](#_Toc141447340)

[38.5 int getAccTrgDist2\_D\_Dsply() 201](#_Toc141447341)

[38.6 interface IARPocChange { 201](#_Toc141447342)

[39 com.ford.car.FordNaviautoSpeedManager 202](#_Toc141447343)

[39.1 void connect(ConnectionCallback connection)//连接服务 202](#_Toc141447344)

[39.2 float getCarSpeed()//获取速度 202](#_Toc141447345)

[39.3 interface SensorChangeListener {//速度的回调 202](#_Toc141447346)

[40 com.ford.car.FordCommonUtil 203](#_Toc141447347)

[40.1 public static boolean isU625() 203](#_Toc141447348)

[40.2 public static boolean isU625Timberline() 203](#_Toc141447349)

[40.3 public static boolean isCD542H() 203](#_Toc141447350)

[40.4 public static boolean isCD542ICAH() 203](#_Toc141447351)

[40.5 public static boolean isCD542L() 203](#_Toc141447352)

[40.6 public static boolean isCD542ICAL() 203](#_Toc141447353)

[40.7 public static boolean isCD764() 203](#_Toc141447354)

[40.8 public static boolean isCDX706H() 203](#_Toc141447355)

[40.9 public static boolean isCDX706L() 203](#_Toc141447356)

[40.10 public static boolean isCX483() 203](#_Toc141447357)

[40.11 public static boolean isP702() 203](#_Toc141447358)

[40.12 public static boolean isMY23P702() 203](#_Toc141447359)

[40.13 public static boolean isMY24P702() 203](#_Toc141447360)

[40.14 public static boolean isCX727() 203](#_Toc141447361)

[40.15 public static boolean isU554() 203](#_Toc141447362)

[40.16 public static boolean isS650() 203](#_Toc141447363)

[40.17 public static boolean isCX727ICA() 203](#_Toc141447364)

[40.18 public static boolean isU554MY23() 204](#_Toc141447365)

[40.19 public static boolean isU554MY24() 204](#_Toc141447366)

[40.20 public static boolean isCX727LFP() 204](#_Toc141447367)

[40.21 public static boolean isCD764ICA() 204](#_Toc141447368)

[40.22 public static boolean is483PTICA() 204](#_Toc141447369)

[40.23 public static boolean isU725C() 204](#_Toc141447370)

[40.24 public static CarSWVariantEnum getCarSWVariant() 204](#_Toc141447371)

[41 com.ford.vendor.FordIodMessageManager 204](#_Toc141447372)

[41.1 int getVehElEffTrip\_No\_Dsply() 204](#_Toc141447373)

[0x7F 204](#_Toc141447374)

[0x0 204](#_Toc141447375)

[-100 204](#_Toc141447376)

[单位：wattHr/km Resolution: 10 Offset: -100 204](#_Toc141447377)

[(Msg: 0x39D TripInfoData3\_HS3) 204](#_Toc141447378)

[0x7D 204](#_Toc141447379)

[1150 204](#_Toc141447380)

[0x7E 205](#_Toc141447381)

[NoDataExists 205](#_Toc141447382)

[0x7F 205](#_Toc141447383)

[Faulty 205](#_Toc141447384)

[41.2 void queryEnergyConsumptionUnit() 205](#_Toc141447385)

[41.3 int getEnergyConsumptionUnit() 205](#_Toc141447386)

[41.4 int getElDistTripUnit\_d\_stat() 205](#_Toc141447387)

[0 205](#_Toc141447388)

[Inactive 205](#_Toc141447389)

[ElDistTripUnit\_D\_Stat indicate the unit to display for the Electric Trip Distance (msg: 0x327 InstFuelEconomy\_3) 205](#_Toc141447390)

[1 205](#_Toc141447391)

[Kilometers 205](#_Toc141447392)

[2 205](#_Toc141447393)

[Miles 205](#_Toc141447394)

[3 205](#_Toc141447395)

[Reserved 205](#_Toc141447396)

[0x4-0xFF 205](#_Toc141447397)

[not update 205](#_Toc141447398)

[41.5 double getElDistTripCur\_No\_Dsply() 205](#_Toc141447399)

[0x1FFFF 206](#_Toc141447400)

[NA 206](#_Toc141447401)

[低字节 206](#_Toc141447402)

[ElDistTripCur\_No\_Dsply 206](#_Toc141447403)

[Resolution: 0.1 (msg: 0x327 InstFuelEconomy\_3) 206](#_Toc141447404)

[NA 206](#_Toc141447405)

[中字节 206](#_Toc141447406)

[NA 206](#_Toc141447407)

[高字节 206](#_Toc141447408)

[0x20000-0xFFFFFF 206](#_Toc141447409)

[not update 206](#_Toc141447410)

[41.6 interface IODMessageChangeListener{ 206](#_Toc141447411)

[41.7 interface IElDistTripCur\_No\_DsplyListener{ 206](#_Toc141447412)

[42 com.ford.vendor.FordModeCardManager 206](#_Toc141447413)

[42.1 int[] getGlblClk() 时间 206](#_Toc141447414)

[42.2 int getGearPos\_D\_Trg() 挡位方向 206](#_Toc141447415)

[42.3 void setCtrStkFeatConfig(int type) 设置DLR的开关 207](#_Toc141447416)

[42.4 void getFeatConfig() 查询DRL的开关状态 207](#_Toc141447417)

[42.5 int[] getVedsBelt安全带的状态 207](#_Toc141447418)

[42.6 int getPsngrFrntDetct\_D\_Actl 副驾是否坐人 207](#_Toc141447419)

[42.7 void operateTheFrontTrunk(boolean open) 打开前备箱 207](#_Toc141447420)

[42.8 int getHoodStatus() 前备箱的开关状态 208](#_Toc141447421)

[42.9 void operateTheRearTrunk(boolean open)打开后备箱 208](#_Toc141447422)

[42.10 int getDrStatTgate\_B\_Actl() 后备箱的开关状态 208](#_Toc141447423)

[42.11 int getVeh\_Lock\_Status()//锁车 208](#_Toc141447424)

[42.12 float getCabnAmb\_Te\_Actl//获取车内温度 208](#_Toc141447425)

[42.13 NavSignalsBean getNavSignalsBean()//导航信号 bean 类 208](#_Toc141447426)

[42.14 public interface INavSignalsListener {//导航信号的回调 209](#_Toc141447427)

[42.15 interface IModeCardListener { 209](#_Toc141447428)

[42.16 int getDay\_Night\_Status() 光亮 209](#_Toc141447429)

[0 209](#_Toc141447430)

[null 209](#_Toc141447431)

[Day\_Night\_Status（BodyInfo\_3\_HS3, Rx） 209](#_Toc141447432)

[1 209](#_Toc141447433)

[day 209](#_Toc141447434)

[2 210](#_Toc141447435)

[night 210](#_Toc141447436)

[42.17 void setTheOrderTimeFromFord(int time) Ford设置时间 210](#_Toc141447437)

[42.18 interface ITheReturnResultToFordListener{//返回回调给Ford 210](#_Toc141447438)

[42.19 int getWiprFront\_D\_Stat() 雨量传感器 210](#_Toc141447439)

[42.20 void setAEISFeatConfig(int type) 设置30分钟怠速开关 210](#_Toc141447440)

[42.21 void getAEISFeatConfig() 查询30分钟怠速开关 211](#_Toc141447441)

[42.22 int getPrkBrkStatus() 电子手刹状态（手动挡配置） 211](#_Toc141447442)

[0 211](#_Toc141447443)

[Not\_Supported 211](#_Toc141447444)

[PrkBrkStatus (msg: 0x213 DesiredTorqBrk\_HS3) 211](#_Toc141447445)

[1 211](#_Toc141447446)

[Rear\_Caliper\_Closed 211](#_Toc141447447)

[2 211](#_Toc141447448)

[Rear\_Caliper\_Transition 211](#_Toc141447449)

[3 211](#_Toc141447450)

[RWU\_By\_EPB\_Active 211](#_Toc141447451)

[4 211](#_Toc141447452)

[Rear\_Caliper\_Open 211](#_Toc141447453)

[5 211](#_Toc141447454)

[EPB\_Limphome\_Active 211](#_Toc141447455)

[6 211](#_Toc141447456)

[ECD\_by\_Brake\_ECU\_Active 211](#_Toc141447457)

[7 211](#_Toc141447458)

[GeneralFault\_MaintenceMode 211](#_Toc141447459)

[0xFF 211](#_Toc141447460)

[not update 211](#_Toc141447461)

[42.23 int getTcuAvailableStat()获取TCU的可用状态; 2 是enable，其他当disable处理 211](#_Toc141447462)

[42.24 int getHotspotEnableStat() 获取热点的开关1 关 2开 211](#_Toc141447463)

[42.25 void switchHotspot(int status) 设置热点的开关 //1 关 2开 211](#_Toc141447464)

[42.26 void setBalanceFader(int balance, int fader)//设置平衡衰减 211](#_Toc141447465)

[42.27 int[] getBalanceFaderValue() 211](#_Toc141447466)

[42.28 public interface IRouteActive\_StListener {//导航激活状态指示 211](#_Toc141447467)

[42.29 public interface IDistanceToNextManeuverListener {//下一个机动点距离 212](#_Toc141447468)

[42.30 public interface IStreetName\_StListener {//下一道路名称 212](#_Toc141447469)

[42.31 public interface ISpeedLimitListener {//限速值 212](#_Toc141447470)

[42.32 public interface ICurentStreetNameListener {//道路名称 212](#_Toc141447471)

[42.33 public interface INav\_RemainTTDestMin\_StListener {//剩余分钟 212](#_Toc141447472)

[42.34 public interface INav\_DistancToDestVal\_StListener {//剩余长度 212](#_Toc141447473)

[42.35 public interface IDestinationListener {//目的地名称 212](#_Toc141447474)

[43 com.ford.vendor.FordWiFiManager 212](#_Toc141447475)

[43.1 int initialize()初始化 213](#_Toc141447476)

[43.2 void WiFiON() 打开wifi 213](#_Toc141447477)

[43.3 void WiFiOFF()关闭wifi 213](#_Toc141447478)

[43.4 interface IWifiCallback{ wifi开关的回调 213](#_Toc141447479)

[43.5 void getWifiStatus()//获取wifi开关状态 213](#_Toc141447480)

[43.6 void scanAPs()//扫描 213](#_Toc141447481)

[43.7 boolean getVehicleConnActivityState() //ccs连接状态 213](#_Toc141447482)

[43.8 void connectAP(HALWlanProfile halWlanProfile) //主动连接wifi 213](#_Toc141447483)

[43.9 void disconnectAP() //主动断开wifi 213](#_Toc141447484)

[43.10 void forgetAP(ApInfo apInfo) //忽略wifi 213](#_Toc141447485)

[43.11 void enableWifiSta() //开启wir wifi底层进程 213](#_Toc141447486)

[43.12 void disableWifiSta() //关闭wir wifi底层进程 213](#_Toc141447487)

[43.13 interface IWifiRespListener { 213](#_Toc141447488)

[44 com.ford.vendor.FordConnectivityManager 214](#_Toc141447489)

[44.1 long getUidIfaceRxBytes(String packageName, int networkType) 214](#_Toc141447490)

[44.2 long getUidIfaceTxBytes(String packageName, int networkType) 214](#_Toc141447491)

[44.3 void disableAccessNetwork(String packageName, int networkType) 214](#_Toc141447492)

[44.4 void enableAccessNetwork(String packageName, int networkType) 214](#_Toc141447493)

# Mic抢占

1. VR APP自己定义一个public static final int HOTWORD = 1999;
2. Manifest.xml加上权限”android.permission.CAPTURE\_AUDIO\_HOTWORD”
3. 语音app在收到其他apps抢占mic的消息后，需要将降噪设置设回正常模式，以免影响其他app的mic使用。

# android.media.AudioManager.java

## requestAudioFocus(AudioFocusRequest focusRequest)

Android原生接口，用于申请焦点，需要构造一个audioAttributes，source类型通过audioAttributes的usage传递给系统。

具体实现可以遵循原生的接口使用：https://developer.android.google.cn/reference/android/media/AudioManager?hl=en#requestAudioFocus(android.media.AudioFocusRequest)

其中，导航播报的参数如下：

CONTENT\_TYPE: com.ford.audio.FordAudioManager.STREAM\_NAVIGATION；

正常播报USAGE:  AudioAttributes.USAGE\_ASSISTANCE\_NAVIGATION\_GUIDANCE；

导航复播USAGE: AudioAttributes.USAGE\_ASSISTANCE\_ACCESSIBILITY。

注：导航复播时，焦点优先级最高，比语音都高

## abandonAudioFocusRequest(AudioFocusRequest focusRequest）

abandonAudioFocus的audioAttributes参数需要和其requestAudioFocus时传入的attribute对应  
**特别重要：**request时的attribute和abandon时的attribute必须为同一个attribute。

## setStreamVolume (int streamType, int index, int flags)

Android原生接口，调节音量。其中，导航音量传入的参数streamtype对应com.ford.audio.FordAudioManager.STREAM\_NAVIGATION。

## List<AudioFocusInfo> getAudioFocusInfo(int type)获取所有的音源list：

参数type：传AUDIOFOCUS\_ALL\_TYPE 即 0返回：所有的音源list

## String getCarAudioType(AudioFocusInfo info)

1.获取当前源 2.获取上一个源  
实现过程：获取所有得音源AudioFocusInfo得infoList,即根据1.4的接口获取音源list;

infoList.get(0)获取当前源得AudioFocusInfo，

infoList.get(1)获取上一个源AudioFocusInfo

根据AudioFocusInfo获取这个源得类型car\_audio\_type

return null:没有当前源或者没有前一个源 notnull:源得类型

## 在线收音机的AudioSourceType：

public static final String CAR\_AUDIO\_TYPE\_ONLINE\_NETWORKSTATION = SvCarAudioManager.CAR\_AUDIO\_TYPE\_ONLINE\_NETWORKSTATION;

## AudioAttributes setAttrShowDisplayId(AudioAttributes attr, int displayId)申请source的接口

## int getAttrShowDisplayId(AudioAttributes attr)获取source的接口

# 平台界面跳转

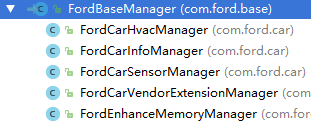
intent.setClassName("com.desay.setting", "com.desay.settings.systemSet.activity.WifiActivity"); //进入WiFi设置界面

intent.setClassName("com.desay.setting", "com.desay.settings.systemSet.bluetooth.activity.BluetoothActivity"); // 进入蓝牙设置界面

intent.setClassName("com.desay.setting", "com.desay.settings.activity.MainActivity"); // 进入车辆控制界面

# com.ford.base.FordBaseManager.java

BaseManager为各个Manager基类，提供connect和disconnect接口。



继承自BaseManager的管理类，调用方式都是xxxxManager.getInstance().connect(ServiceConnection)，收到onServiceConnected回调后，表示Service连接成功，可以开始业务逻辑。

# com.ford.car.FordCarInfoManager.java

## public static FordCarInfoManager getInstance(Context context)

获取单例。

## int getCarModel()获取当前车型。

返回值说明：

**public static final int *CAR\_MODEL\_DEFAULT*** = 0;*//* *Defaul用来做DTC***public static final int *CAR\_MODEL\_CD542*** = 1;*//CD542***public static final int *CAR\_MODEL\_CX727*** = 2;*//* *CX727*

**public static final int *CAR\_MODEL\_U625*** = 3;*//* *U625***public static final int *CAR\_MODEL\_P702*** = 4;*//* *P702***public static final int *CAR\_MODEL\_U725*** = 5;*//* *U725 注：这个不是8155的U725***public static final int *CAR\_MODEL\_U554*** = 6;*//* *U554*

**public static final int *CAR\_MODEL\_CD764***= 7;*//* *CD764*

*public static final int CAR\_MODEL\_CX706 = 8;// CX706*

*public static final int CAR\_MODEL\_S650 = 9;// S650*

*public static final int CAR\_MODEL\_CX483 = 0xA;// CX48*

***注：8155的U725的车型判断请使用FordCommonUtil的CarSWVariantEnum getCarSWVariant()方法***

## String getVinCode()获取Vin码。

## String getCarColor()获取车辆颜色

727:见carProty中对应的颜色值；

## String getESN()获取车机设备电子序列号。

## String getPCBANumber()获取车机PCBA零件号。

## String getAssemblyNumber()获取车机设备型号。

## String getVMCUVersion()获取VMCU版本号。

## String getSerialNo()

获取序列号，等效于SystemProperties.*get*(**"ro.serialno"**);

## String getSystemModel()

获取系统信号，等效于SystemProperties.*get*(**"ro.product.model"**);

## String getSystemVersion()

获取系统版本，等效于SystemProperties.*get*(**"ro.build.id"**);

## int getCarPowerType()获取车辆动力类型

返回值，0：GAS，1：PHEV，2：BEV 3:HEV 4：PHEV

## void setActiveStatus(int status)写激活状态。

## int getActiveStatus()读取激活状态。

## String getSystemType()

获取ROM版本类型，debug版本返回“debug”，release版本返回“release”。

## String getCarType()

获取车辆样式，

0x 0: Sedan 0x 1：SUV 0x 2：Hatchback 0x 3：Wagon 0x 4：Pickup

## String getCarStyle()获取车辆款式

## boolean has360Camera()

是否有360Camera。true表示有360，false表示没有。

## String getCloudDataDir(Context context)返回值：/cert/data/cloud

## interface IProvisionStateResult {

**void onProvisionStateResultChange(HashMap<Integer,Integer> hashMap);**

**Hashmap:Integer1:认证的数字**

**Integer2:认证数字的认证状态 -1：失败 0成功 1超时**

## int get542CarModel()返回值；

public static final int *CAR\_MODEL\_CD542H* = 101;  
public static final int *CAR\_MODEL\_CD542L* = 102;  
public static final int *CAR\_MODEL\_NOT\_CD542* =100;

## int getDisplayvariants()

显示屏接口，判断车型是H或者L 详情见Phase4 配置位文档 DE00 B7 7 4. Display variants

返回值等于0x2是H,非2就是L;

## void setUsbSwitch(String usbSwitch)

*SystemProperties.set("sys.sv.usbswitch", "0"); // 切到host模式  
SystemProperties.set("sys.sv.usbswitch", "1"); // 切到peripheral模式*

# com.ford.car.FordCarVendorExtensionManager.java

## void startServiceAsUser(Intent intent)

对外提供startServiceAsUser接口：



## String getBootAnimIsStopped()

开机动画是否停止。返回“stopped”表示已停止。

## void setNeedLoginCheck(String loginCheck)

设置是否强制检验账号登录。"logincheckclose"表示免登录，"logincheckopen"表示需要强制登录。

## String getNeedLoginCheck()

获取是否强制检验账号登录。"logincheckclose"表示免登录，"logincheckopen"表示需要强制登录。

## void setAuthResult(int authed)

保存账号授权协议的结果。1：已授权，0：未授权。

## String getVolumeState(String mountPoint) ；USB得挂载情况

参数**mountPoint**：**public static final String *STORAGE\_USB* = "/storage/usb";**

返回：unknown 没有挂载上 mounted usb已挂载

## int getDayAndNightMode()获取白天黑夜模式;

public static final int GET\_DAY\_MODE = 2;黑夜

public static final int GET\_NIGHT\_MODE = 1;白天

0.代表null,无效值；

## void registerDayAndNightModeListener(ISvDataBusDispatcher.Stub listener)注册白天黑夜模式的监听

## void unregisterDayAndNightModeListener(ISvDataBusDispatcher.Stub listener)反注册白天黑夜模式的监听

## int getPowerState()获取电源的模式

    public static final int STATE\_NORMAL\_EXT\_PLAY = DesayCommonManager.STATE\_NORMAL\_EXT\_PLAY;

//扩展模式SS\_EXT\_PLAY

     public static final int STATE\_NORMAL\_WORK = DesayCommonManager.STATE\_NORMAL\_WORK;

//正常模式SS\_MM\_ACTIVE

     public static final int STATE\_ABNORMAL = DesayCommonManager.STATE\_ABNORMAL;

//Load Shed模式

     public static final int STATE\_STANDBY = DesayCommonManager.STATE\_STANDBY;

//待机模式SS\_MM\_INACTIVE

     public static final int STATE\_SLEEP = DesayCommonManager.STATE\_SLEEP;//休眠模式SS\_OFF

     public static final int STATE\_TRANSPORT = DesayCommonManager.STATE\_TRANSPORT;//运输模式

## void registerPowerStateListener(ISvDataBusDispatcher.Stub listener) *注册电源模式得回调*}

## void unregisterPowerStateListener(ISvDataBusDispatcher.Stub listener) *取消电源模式得回调*}

## ActivityManager.RunningTaskInfo getTopTaskOfDisplay(int displayId)获取

## interface IBootAnimationCompleted {void onBootAnimationCompleted(boolean state);}开机动画结束的回调

## void setAARState(int state)设置aar的状态

## int initialize\_syncp()syncp的接口 0:失败，1成功

**注：仅限V2I使用**

## byte[] decode(byte[] decodeBuffer, int size)

syncp的接口；先initialize\_syncp()，然后传入需要解密的加密的数据；

decodeBuffer：传入的加密的数据

size:加密数据的size;

**注：仅限V2I使用**

## void connect(ConnectionCallback connection) 连接soa服务

## int sendProfileValueResponse(String transactionId, int commandStatus, ProfileErrorDetail profileErrorDetail)

## interface IPortableResponseListener {

**void OnPortableInfo(String transationId, List<ProfileValueMap> tokenData); }**

## ProfileStructureInfo ：LabelType; //label数据类型 0：是整型，1：String类型  2：Double类型 3：Float 类型 4:  byte 类型，这里设定类型后，下面选择对应的Value值设置，比如 LabelType 设置2，就给DoubleValue设置值

## List<ProfileValueMap> getPortableData()

**Rocket setup的测试数据**

## int getWifiSwitchState()//获取wifi的开关状态

1是开，非1就是关；

## int getWifiConnState()//获取wifi的连接状态

1是开已连接 非1是未连接

## interface IWifiSwitchState {//wifi开关状态的回调

**void onWifiSwitchStateChange(int state);}**

## interface IWifiConnState {//wifi连接状态的回调

**void onWifiConnStateChange(int state);}**

## int getDlnaSwitch()//dlna的总开关1：开， 0：关

## int getPhoneSwitch()

DLNA手机热点模式：phoneSwitch = 1 车辆热点模式：phoneSwitch = 0

## interface IDlnaSwitch{//dlna总开关的回调

**void onDlnaStateChange(int state);}**

## interface IPhoneHotspotSwitchState{

**void onPhoneHotspotSwitchStateChange(int state);}**

## int getStrPowerState //STR的状态；

public static final int SUSPEND\_ENTER = 2;

public static final int SUSPEND\_EXIT = 3;

public static final int SHUTDOWN\_PREPARE = 7;

public static final int SHUTDOWN\_CANCELLED = 8;

## interface IStrPowerStateListener{

**void onStateChanged(int state);}**

**public static final int SUSPEND\_ENTER = 2;**

**public static final int SUSPEND\_EXIT = 3;**

**public static final int SHUTDOWN\_PREPARE = 7;**

**public static final int SHUTDOWN\_CANCELLED = 8;**

## public void registerStrPowerStateListenerWithCompletion(DesayCommonManager.

**CarPowerStateListenerWithCompletion listener) STR 延时的注册调用接口**

**对应接口的解释如下：**

**public interface CarPowerStateListenerWithCompletion {**

**void onStateChanged(int state, CompletableFuture<Void> future);}**

**state:对应的STR类型（和6.31是一样的）**

**future:当延时完成是调用如下代码结束：**

**if (future!=null){  
 future.complete(null);}**

## public void unregisterStrPowerStateListenerWithCompletion(DesayCommonManager.

**CarPowerStateListenerWithCompletion listener) STR延时的反注册接口；**

## int getBtMusicDisplayState()//获取蓝牙音乐的显示/隐藏状态

public static final int DISPLAY\_HIDE=0;//隐藏

public static final int DISPLAY\_SHOW=1;//显示

## int getRadioDisplayState()//获取radio的显示隐藏状态

public static final int DISPLAY\_HIDE=0;//隐藏

public static final int DISPLAY\_SHOW=1;//显示

## int getSuixintingDisplayState() //获取随心听显示隐藏状态

public static final int DISPLAY\_HIDE=0;//隐藏

public static final int DISPLAY\_SHOW=1;//显示

## void setSuixintingDisplayState(int state)//设置随心听的显示隐藏状态

state:

public static final int DISPLAY\_HIDE=0;//隐藏

public static final int DISPLAY\_SHOW=1;//显示

## interface IBtMusicDisplayStateListener {//蓝牙音乐显示隐藏状态的回调

**void onBtMusicDisplayStateChange(int state);}**

**state:**

**public static final int DISPLAY\_HIDE=0;//隐藏**

**public static final int DISPLAY\_SHOW=1;//显示**

## interface IRadioDisplayStateListener {//radio显示隐藏的回调

**void onRadioDisplayStateChange(int state);}**

**state:**

**public static final int DISPLAY\_HIDE=0;//隐藏**

**public static final int DISPLAY\_SHOW=1;//显示**

## interface ISuixintingDisplayStateListener {//随心听显示隐藏的回调

**void onSuixintingDisplayStateChange(int state);}**

**state:**

**public static final int DISPLAY\_HIDE=0;//隐藏**

**public static final int DISPLAY\_SHOW=1;//显示**

## int getCarPlayMusicDisplayState() //carplay显示和隐藏

**public static final int DISPLAY\_HIDE=0;//隐藏**

**public static final int DISPLAY\_SHOW=1;//显示**

## interface ICarPlayMusicDisplayStateListener {

**void onCarPlayMusicDisplayStateChange(int state);}**

**public static final int DISPLAY\_HIDE=0;//隐藏**

**public static final int DISPLAY\_SHOW=1;//显示**

# com.ford.car.FordCarSensorManager.java

## public static FordCarSensorManager getInstance(Context context)

获取Manager单例。

## int getCarSpeedUnit()获取车速单位。

返回值说明：

异步调用，回调回来的值才是车速单位代表的值，

回调回来的值；0x00:OFF 代表单位；MPH 0x01:ON 代表单位Km/h

## float getCarSpeed()获取车速。

## int getCarDriveMode()获取驾驶模式

驾驶模式can返回的值对应的如下表中：

|  |  |
| --- | --- |
| 0x0 | SelDrvMde01 |
| 0x1 | SelDrvMde02 |
| 0x2 | SelDrvMde03 |
| 0x3 | SelDrvMde04 |
| 0x4 | SelDrvMde05 |
| 0x5 | SelDrvMde06 |
| 0x6 | SelDrvMde07 |
| 0x7 | SelDrvMde08 |
| 0x8 | SelDrvMde09 |
| 0x9 | SelDrvMde10 |
| 0xA | SelDrvMde11 |
| 0xB | SelDrvMde12 |
| 0xC | SelDrvMde13 |
| 0xD | SelDrvMde14 |
| 0xE | SelDrvMde15 |
| 0xF | SelDrvMde16 |
| 0x10 | SelDrvMde17 |
| 0x11 | SelDrvMde18 |
| 0x12 | SelDrvMde19 |
| 0x13 | SelDrvMde20 |
| 0x14 | SelDrvMde21 |
| 0x15 | SelDrvMde22 |
| 0x16 | SelDrvMde23 |
| 0x17 | SelDrvMde24 |
| 0x18 | SelDrvMde25 |
| 0x19 | SelDrvMde26 |
| 0x1A | SelDrvMde27 |
| 0x1B | SelDrvMde28 |
| 0x1C | SelDrvMde29 |
| 0x1D | SelDrvMde30 |
| 0x1E | SelDrvMde31 |
| 0x1F | Faulty |
| 0x20-0xFF | not update |

## int getCarDoorOpenStatus(int doorPos)获取各位置的车门打开状态。

参数：

**public static final int *DOOR\_FRONT\_LEFT*** = 0; //左前车门  
**public static final int *DOOR\_FRONT\_RIGHT*** = 1; //右前车门  
**public static final int *DOOR\_REAR\_LEFT*** = 2; //左后车门  
**public static final int *DOOR\_REAR\_RIGHT*** = 3; //右后车门  
**public static final int *DOOR\_OUTSIDE\_TRUNK*** = 4; //后备箱外车门  
**public static final int *DOOR\_INSIDE\_TRUNK*** = 5; //后备箱内车门

返回值，0：关闭，1：打开

Can消息返回对应的值如下表：

|  |  |
| --- | --- |
| 0x0 | Closed |
| 0x1 | Ajar |

## float getCarEnduranceMileage()获取车辆续航里程（电车）

## float getCarEndurancePercent()获取车辆续航百分比

## int getCarEnduranceWarnStatus()获取续航警告状态

Can信号返回的值如下表中：

|  |  |
| --- | --- |
| 0x0 | Null |
| 0x1 | Thres20mi\_32km |
| 0x2 | Thres30mi\_48km |
| 0x3 | Thres50mi\_80km |
| 0x4 | Thres30km\_18mi |
| 0x5 | Thres50km\_31mi |
| 0x6 | Thres80km\_50mi |
| 0x7 | NotUsed |

## float getCarFuelPercent()获取剩余油量百分比。

比如返回值为78.1，代表还剩余78.1%的油量。

## int getCarFuelWarnStatus()获取油量告警状态

返回值，0：正常，1：low 2.verylow

Can消息返回的值见下表中：

|  |  |
| --- | --- |
| 0x0 | OK |
| 0x1 | Low |
| 0x2 | VeryLow |
| 0x3 | DteLevel1MyKey |
| 0x4 | DteLevel2NonMyKey |
| 0x5 | DteLevel3 |
| 0x6 | DteLevel4 |
| 0x7 | DteLevel5Lowest |

## int getCarGear()获取车辆档位

返回值:

Can信号返回的值见下表中:

|  |  |
| --- | --- |
| 0 | Park |
| 1 | Reverse |
| 2 | Neutral |
| 3 | Drive |
| 4 | Sport\_DriveSport |
| 5 | Low |
| 6 | first |
| 7 | second |
| 8 | third |
| 9 | fourth |
| 0xA | fifth |
| 0xB | sixth |
| 0xC | Undefined\_Treat\_as\_Fault |
| 0xD | Undefined\_Treat\_as\_\_Fault1 |
| 0xE | Unknown\_Position |
| 0xF | Fault |
| 0xFF | not update |

## int getCarMileageUnit()获取里程单位

返回值：

Can消息返回的值见下表中：

|  |  |
| --- | --- |
| 0 | TripComputer\_metric 公里 |
| 1 | TripComputer\_imperial 英里 |

## float getCarTirePressure(int tirePos)获取胎压

参数：tirePos

public static final int Tire\_Press\_LF\_Data=0;左前轮胎

public static final int Tire\_Press\_RF\_Data=1;右前轮胎

public static final int Tire\_Press\_RR\_ORR\_Data=2; 右后外部轮胎

public static final int Tire\_Press\_LR\_OLR\_Data=3; 左后外部轮胎

public static final int Tire\_Press\_IRR\_Data=4; 右后内部轮胎

public static final int Tire\_Press\_ILR\_Data=5; 左后内部轮胎

## int getCarTirePressureUnit()获取胎压单位

返回值:异步调用，返回的回调的值才是代表的胎压单位，

## float getCarTireTemperature(int tirePos)获取胎温

参数：

**public static final int *TIRE\_FRONT\_LEFT*** = 0;//前左轮胎  
**public static final int *TIRE\_FRONT\_RIGHT*** = 1;//前右轮胎  
**public static final int *TIRE\_REAR\_LEFT*** = 2;//后左轮胎  
**public static final int *TIRE\_REAR\_RIGHT*** = 3;//后右轮胎

## int getCarTireTemperatureUnit()获取胎温单位

返回值：同温度单位

## int getCarTireStatus(int tirePos)获取轮胎状态

参数：

**public static final int *Tire\_Press\_LF\_Stat*** = 0;//前左轮胎的胎压状态  
**public static final int *Tire\_Press\_RF\_Stat*** = 1;//前右轮胎的胎压状态  
**public static final int Tire\_Press\_RR\_ORR\_Stat=2;右后外部轮胎的胎压状态**

**public static final int Tire\_Press\_LR\_OLR\_Stat=3;左后外部轮胎的胎压状态**

**public static final int Tire\_Press\_IRR\_Stat=4;右后内部轮胎的胎压状态**

**public static final int Tire\_Press\_ILR\_Stat=5;左后内部轮胎的胎压状态**

**Can消息返回的值见下表中：**

|  |  |
| --- | --- |
| 0 | Unknown |
| 1 | Normal |
| 2 | Low |
| 3 | Fault |
| 4 | Alert |
| 0xF | Not\_Supported |

## int getCarTireSupportResult()车辆是否支持获取胎温、胎压

返回值，

**public static final int *SUPPORT\_TIRE\_NO*** = 0; *//都不支持***public static final int *SUPPORT\_TIRE\_TEMP\_PRESSURE*** = 1; *//都支持***public static final int *SUPPORT\_TIRE\_PRESSURE*** = 2; *//仅支持获取胎压*

## float getCarTotalMileage()获取车辆总里程

## int getCarWindowOpenStatus(int windowPos)获取车窗打开状态

参数，

**public static final int *WINDOW\_POS\_DRIVER*** = 0; //驾驶位车窗  
**public static final int *WINDOW\_POS\_FRONT\_PASSENGER*** = 1; //副驾车窗  
**public static final int *WINDOW\_POS\_REAR\_DRIVER*** = 2; //驾驶位后排车窗  
**public static final int *WINDOW\_POS\_REAR\_PASSENGER*** = 3; //副驾位后排车窗

Can消息返回值见下表中：，

|  |  |
| --- | --- |
| 0x0 | Undefined |
| 0x1 | Fully\_Closed |
| 0x2 | BetFully\_10PercentOpen |
| 0x3 | Bet10Percent\_60Percent |
| 0x4 | Bet60Percent\_FullyOpen |
| 0x5 | Fully\_Open |
| 0x6 | Unused1 |
| 0x7 | Unused2 |

## float getChargePercentage()电量百分比

## addListener(SensorChangeListener listener)

增加listener。

## removeListener(SensorChangeListener listener)

移除listener。

## clearListeners()

清空listener集合。

## float getOutCarTemperature()获取车外温度 返回值：温度的值

## 回调接口

**public interface** SensorChangeListener {  
  
 **void** onSensorChange(**int** eventType, **int**[] intValues, **float**[] floatValues);  
}

FordCarSensorManager的各种传感器类信息，例如车速、档位等变化，统一回调onSensorChange接口。eventType表示对应事件类型，intValues保存回调的int型值，floatValues保存回调的float型值。

例如，收到onSensorChange(0, intValues, floatValues)，eventType为0，代表驾驶模式发生了变化，驾驶模式值是一个int型，所以取intValues[0]即可；没有float值，floatValues为null。

## sendNaviToTCU(Bundle bundle)

将处理后的导航信息，发送给车机系统。

Bundle参数，用法与原先保持一致：



## void connect(ConnectionCallback connection) 连接服务

## void disConnect() 断开服务

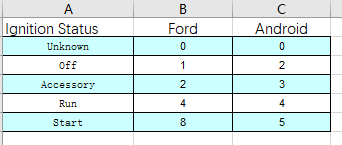
## int getTemperatureUnit()获取温度单位can信号返回的对应的值如下：

|  |  |
| --- | --- |
| 0 | Temperature\_deg\_c |
| 1 | Temperature\_deg\_f |

## 

## int getIgnition\_Status() 获取点火状态 can信号返回的值如下：

点火状态给到java app，走的是Android原生的carsensormanager，所以需要将福特的点火状态与Android carsensormanager定义的点火状态一一映射，映射关系如下：



## float getBearing() 获取车的朝向

## void setDriveMode(int mode) 设置驾驶模式

## int getStabCtlBrkActv\_B\_Actl()

## int getTurnLghtRightOn\_B\_Stat()

## int getApedPos\_Pc\_ActlArb()

## int getStePinCompAnEst\_D\_Qf()

## int getStePinComp\_An\_Est()

## int getStopStrtDrvMde\_D\_Indic()

## int getWhlRotatRr\_No\_Cnt()

## int getWhlRotatRl\_No\_Cnt()

## int getBpedDrvAppl\_D\_Actl()

## int getGearLvrPos\_D\_Actl()

## int getTurnLghtRight\_D\_Rq()

## int getTurnLghtLeft\_D\_Rq()

## int[] getVehicleGGCCDataArr()

## float getFuelCarEnduranceMileage()油量剩余里程（油车）

## int getTeltalWarnData\_No\_Actl()防抱死系统故障;

## int getThe\_tire\_pressure\_detection\_system\_detects\_faults()胎压监测系统警告

## int getThe\_engine\_maintenance\_lamp\_is\_out\_of\_order() 发动机故障

## int getEngine\_overheat\_condition() 冷却液温度过高

## int getEngine\_oil\_pressure\_test() 机油压力低

## int getPower\_system\_failure()动力系统故障

## int getThe\_electric\_power\_steering\_system\_detects\_faults()

电动转向故障

## int getSteep\_descent\_control\_system() 陡坡缓降系统故障

## int getFailure\_detection\_of\_ramp\_starting\_auxiliary\_system()

坡道起步辅助故障

## int getFault\_detection\_by\_external\_light\_system() 照明系统故障

## int getWindshield\_washing\_liquid\_level()挡风玻璃洗涤液液位状态

## int getAll\_wheel\_drive\_system\_detects\_faults()

全轮驱动或四轮驱动系统检测故障(4, 4)

## int getDetection\_of\_stolen\_goods\_in\_air\_filter\_element()

空气滤清器滤芯赃物检测：是否脏污(4, 3)

## void skipToRoadRescueActivity()

道路救援入口跳转

## boolean initAudioOFFState AudioOFF初始状态

## void skipToVHA()

542H的VHA（车辆健康）入口跳转

## 驾驶模式

eventType:

public static final int EVENT\_TYPE\_DRIVE\_MODE = 0;

回调值：

intValues[0]，代表驾驶模式的值，

## 车速

eventType:

public static final int EVENT\_TYPE\_SPEED\_VALUE = 1;

回调值：

floatValues[0]，代表车速值

## 总里程;

eventType:

public static final int EVENT\_TYPE\_TOAL\_MILEAGE\_VALUE = 2;

floatValues[0]代表总里程

## 里程单位

eventType:

public static final int EVENT\_TYPE\_MILEAGE\_UNIT = 3;

回调值：

intValues[0]，代表里程单位，

*0是公里 1是英里*

## 油量

eventType:

public static final int EVENT\_TYPE\_FUEL = 4;

回调值：

intValues[0]，代表油量告警状态，

## 档位

eventType:

public static final int EVENT\_TYPE\_GEAR = 5;

回调值：

intValues[0]，代表档位，

## 车窗

eventType:

public static final int EVENT\_TYPE\_WINDOW = 6;

回调值：

车窗开关状态

ints[0] = driverWindowPosition;主驾车窗的状态

ints[1] = passWindowPosition;副驾车窗的状态

ints[2] = rearDriverWindowPosition;主驾后排车窗的状态

ints[3] = rearPassWindowPosition;副驾后排车窗的状态

## 车门

eventType:

public static final int EVENT\_TYPE\_DOOR = 7;

回调值：

ints[0] = drStatDrv\_b\_act;左前门状态

ints[1] = drStatPsngr\_b\_act;右前门状态

ints[2] = drStatRl\_b\_actl;左后门状态

ints[3] = drStatRr\_b\_actl;右后门状态

ints[4] = drStatInnrTgate\_b\_actl;后备箱内车门状态

ints[5] = drStatTgate\_b\_actl;后备箱外车门状态

## 胎压单位

eventType:

public static final int EVENT\_TYPE\_TIRE\_PRESURE\_VALUE = 8;

回调值：

ints[0] 胎压单位

## 胎压

eventType:

public static final int EVENT\_TYPE\_TIRE\_PRESSURE = 9;

返回值：

ints[0] = Tire\_Press\_LF\_Data;左前轮胎

ints[1] = Tire\_Press\_RF\_Data;右前轮胎

ints[2] = Tire\_Press\_RR\_ORR\_Data; 右后外部轮胎

ints[3] = Tire\_Press\_LR\_OLR\_Data; 左后外部轮胎

ints[4] = Tire\_Press\_IRR\_Data; 右后内部轮胎

ints[5] = Tire\_Press\_ILR\_Data; 左后内部轮胎

float[0] = rsp.getTire\_Press\_LF\_Data();左前轮胎胎压

float[1] = rsp.getTire\_Press\_RF\_Data();右前轮胎胎压

float[2] = rsp.getTire\_Press\_RR\_ORR\_Data();右后外部轮胎胎压

float[3] = rsp.getTire\_Press\_LR\_OLR\_Data();左后外部轮胎胎压

float[4] = rsp.getTire\_Press\_IRR\_Data();右后内部轮胎胎压

float[5] = rsp.getTire\_Press\_ILR\_Data();左后内部轮胎胎压

## 车外温度

eventType:

public static final int EVENT\_TYPE\_OUT\_CAR\_TEMPERATUE\_VALUE = 10;

返回值：

float[0]车外温度

## 轮胎状态

eventType:

public static final int EVENT\_TYPE\_TIRE\_STATUS\_VALUE = 11;

返回值；

intValue[0] = rsp.getTire\_Press\_LF\_Stat();左前轮的胎压状态

intValue[1] =rsp.getTire\_Press\_RF\_Stat();右前轮的胎压状态

intValue[2]= rsp.getTire\_Press\_RR\_ORR\_Stat();右后外部轮胎的胎压状态

intValue[3]= rsp.getTire\_Press\_LR\_OLR\_Stat();左后外部轮胎的胎压状态

intValue[4] = rsp.getTire\_Press\_IRR\_Stat();右后内部轮胎的胎压状态

intValue[5] = rsp.getTire\_Press\_ILR\_Stat();左后内部轮胎的胎压状态

## 剩余里程（电池续航里程）

eventType:

public static final int EVENT\_TYPE\_CARENDURANCEMILEAGE\_VALUE = 12;

返回值；

float[0]剩余里程（电池续航里程）

## 温度单位

eventType:

public static final int EVENT\_TYPE\_TEMPERATURE\_UNIT = 13;

返回值；

int[0]温度单位

## 胎温

eventType:

public static final int EVENT\_TYPE\_TIRE\_TEMPERATUE\_VALUE = 14;

返回值；

ints[0] = TIRE\_FRONT\_LEFT;左前轮胎

ints[1] = TIRE\_FRONT\_RIGHT;右前轮胎

ints[2] = TIRE\_REAR\_LEFT; 左后轮胎

ints[3] = TIRE\_REAR\_RIGHT; 右后轮胎

floats2[0] = tire\_temp\_lf\_data;左前轮胎的胎温

floats2[1] = tire\_temp\_rf\_data;右前轮胎的胎温

floats2[2] = tire\_temp\_olr\_data;左后轮胎的胎温

floats2[3] = tire\_temp\_orr\_data;右后轮胎的胎温

## 速度单位

eventType:

public static final int EVENT\_TYPE\_SPEED\_UNIT\_VALUE = 15;

返回值；

int[0] 速度单位 值代表的含义：0x00:MPH 0x01;Km/h

## 续航警告

eventType:

public static final int EVENT\_TYPE\_ENDURANCE\_WARN\_VALUE = 16;

返回值；

ints[0]续航警告状态

## 续航百分比

eventType:

public static final int EVENT\_TYPE\_ENDURANCE\_MILEAGE\_PERCENT\_VALUE = 17;

返回值；

floats[0] 续航百分比

## 点火状态

eventType:

public static final int EVENT\_TYPE\_IGNITION = 18;

回调值：

intValues[0]，代表点火状态值，

## 油量百分比

eventType:

public static final int *EVENT\_TYPE\_FUEL\_PERCENT* = 19;

返回值：float[0]

## 总里程

eventType:

public static final int *EVENT\_TYPE\_MILEAGE\_VALUE* = 20;

返回值：float[0]

## VehicleGGCCData

eventType:

public static final int EVENT\_TYPE\_VEHICLEGGCCDATA = 21;

返回值：是一个int的数组；共有8个int的值，分别为

int[0]\_int[7]

## TurnLghtLeft\_D\_Rq

eventType:

public static final int EVENT\_TYPE\_TURNLGHTLEFT\_D\_RQ = 22;

返回值：int[0]

## TurnLghtRight\_D\_Rq

eventType:

public static final int EVENT\_TYPE\_TURNLGHTRIGHT\_D\_RQ = 23;

返回值：int[0]

## GearLvrPos\_D\_Actl

eventType:

public static final int EVENT\_TYPE\_GEARLVRPOS\_D\_ACTL = 24;

返回值：int[0]

## BpedDrvAppl\_D\_Actl

eventType:

public static final int EVENT\_TYPE\_BPEDDRVAPPL\_D\_ACTL = 25;

返回值：int[0]

## WhlRotatRl\_No\_Cnt

eventType:

public static final int EVENT\_TYPE\_WHLROTATRL\_NO\_CNT = 26;

返回值：int[0]

## WhlRotatRr\_No\_Cnt

eventType:

public static final int EVENT\_TYPE\_WHLROTATRR\_NO\_CNT = 27;

返回值：int[0]

## StopStrtDrvMde\_D\_Indic

eventType:

public static final int EVENT\_TYPE\_STOPSTRTDRVMDE\_D\_INDIC = 28;

返回值：int[0]

## StePinComp\_An\_Est

eventType:

public static final int EVENT\_TYPE\_STEPINCOMP\_AN\_EST = 29;

返回值：int[0]

## StePinCompAnEst\_D\_Qf

eventType:

public static final int EVENT\_TYPE\_STEPINCOMPANEST\_D\_QF = 30;

返回值：int[0]

## ApedPos\_Pc\_ActlArb

eventType:

public static final int EVENT\_TYPE\_APEDPOS\_PC\_ACTLARB = 31;

返回值：int[0]

## TurnLghtRightOn\_B\_Stat

eventType:

public static final int EVENT\_TYPE\_TURNLGHTRIGHTON\_B\_STAT = 32;

返回值：int[0]

## StabCtlBrkActv\_B\_Actl

eventType:

public static final int EVENT\_TYPE\_STABCTLBRKACTV\_B\_ACTL = 33;

返回值：int[0]

## 油量剩余里程

eventType:

public static final int EVENT\_TYPE\_FUELCARENDURANCEMILEAGE= 34;

返回值：float[0]

## 胎温单位

eventType:

public static final int *EVENT\_TYPE\_TIRE\_TEMP\_UNIT* = 45

返回值：int[0]

注;同温度单位的实现

## 防抱死系统故障;

eventType:

public static final int *EVENT\_TYPE\_TELTALWARNDATA\_NO\_ACTL* = 35;

返回值：int[0]

## 胎压监测系统警告;

eventType:

public static final int *EVENT\_TYPE\_THE\_TIRE\_PRESSURE\_DETECTION\_SYSTEM\_DETECTS\_FAULTS* = 36;

返回值：int[0]

## 发动机故障

eventType:

public static final int *EVENT\_TYPE\_THE\_ENGINE\_MAINTENANCE\_LAMP\_IS\_OUT\_OF\_ORDER* = 37;

返回值：int[0]

## 冷却液温度过高;

eventType:

public static final int *EVENT\_TYPE\_ENGINE\_OVERHEAT\_CONDITION* = 38;

返回值：int[0]

## 机油压力低;

eventType:

public static final int *EVENT\_TYPE\_ENGINE\_OIL\_PRESSURE\_TEST* = 39;

返回值：int[0]

## 动力系统故障;

eventType:

public static final int *EVENT\_TYPE\_POWER\_SYSTEM\_FAILURE* = 40;

返回值：int[0]

## 电动转向故障;

eventType:

public static final int *EVENT\_TYPE\_THE\_ELECTRIC\_POWER\_STEERING\_SYSTEM\_DETECTS\_FAULTS* = 41;

返回值：int[0]

## 陡坡缓降系统故障;

eventType:

public static final int *EVENT\_TYPE\_STEEP\_DESCENT\_CONTROL\_SYSTEM* = 42;

返回值：int[0]

## 坡道起步辅助故障;

eventType:

public static final int *EVENT\_TYPE\_FAILURE\_DETECTION\_OF\_RAMP\_STARTING\_AUXILIARY\_SYSTEM* = 43;

返回值：int[0]

## 照明系统故障;

eventType:

public static final int *EVENT\_TYPE\_FAULT\_DETECTION\_BY\_EXTERNAL\_LIGHT\_SYSTEM* = 44;

返回值：int[0]

## 胎温单位(同温度单位)

public static final int *EVENT\_TYPE\_TIRE\_TEMP\_UNIT* = 45;

返回值：int[0]

## 电量百分比

public static final int *EVENT\_TYPE\_CHARGEPERCENTAGE* = 47;

返回值：float[0]

## 挡风玻璃洗涤液液位状态

public static final int EVENT\_TYPE\_WINDSHIELD\_WASHING\_LIQUID\_LEVEL = 49;

返回值：int[0]

|  |  |
| --- | --- |
| 0 | 正常 |
| 1 | 低 |
| 2 | 正在检测中 |

## 全轮驱动或四轮驱动系统检测故障

public static final int EVENT\_TYPE\_ALL\_WHEEL\_DRIVE\_SYSTEM\_DETECTS\_FAULTS = 50;

返回值：int[0]

|  |  |
| --- | --- |
| 0 | 正常 |
| 1 | 故障告警 |
| 2 | 正在检测中 |

## 空气滤清器滤芯赃物检测

public static final int EVENT\_TYPE\_DETECTION\_OF\_STOLEN\_GOODS\_IN\_AIR\_FILTER\_ELEMENT = 51;

返回值：int[0]

|  |  |
| --- | --- |
| 0 | 正常 |
| 1 | 故障告警 |
| 2 | 正在检测中 |

## 定义的宏：

public static final int *SPEED\_UNIT\_CMS* = 0;// "cm/s";  
public static final int *SPEED\_UNIT\_MS* = 1;//"m/s";  
public static final int *SPEED\_UNIT\_KMH* = 2;//"km/h";

public static final int *MILEAGE\_UNIT\_MILE* = 0;//"英里"  
public static final int *MILEAGE\_UNIT\_KM* = 1;//"公里"

public static final int *TIRE\_PRESSURE\_UNIT\_BAR* = 0;//bar  
public static final int *TIRE\_PRESSURE\_UNIT\_KPA* = 1;//kpa

public static final int *TIRE\_TEMPERATURE\_UNIT\_CEN* = 0;//centigrade  
  
  
public static final int *DRIVE\_MODE\_SOLO* = 0;//solo mode  
public static final int *DRIVE\_MODE\_CO\_PILOT* = 1;//co-pilot mode  
public static final int *DRIVE\_MODE\_INDIVIDUAL* = 2;//individual mode

public static final int *IGNITION\_STATUS\_OFF* = 2;//ignition-off  
public static final int *IGNITION\_STATUS\_ON* = 4;//ignition-on  
  
public static final int *FUEL\_STATUS\_OK* = 0; //fuel status is ok  
public static final int *FUEL\_STATUS\_WARN* = 1; //fuel status is warning  
  
public static final int *ENDURANCE\_STATUS\_OK* = 0; //endurance status is ok  
public static final int *ENDURANCE\_STATUS\_WARN* = 1; //endurance status is warning  
  
public static final int *GEAR\_PARKING* = 1; //P  
public static final int *GEAR\_REVERSE* = 2; //R  
public static final int *GEAR\_NEUTRAL* = 3; //N  
public static final int *GEAR\_SPORT* = 4; //S  
public static final int *GEAR\_DRIVE* = 5; //D  
public static final int *GEAR\_LOW* = 6; //L  
  
public static final int *TIRE\_STATUS\_NORMAL* = 1;  
public static final int *TIRE\_STATUS\_UNNORMAL* = 2;  
  
public static final int *WINDOW\_DOOR\_STATUS\_CLOSE* = 1;  
public static final int *WINDOW\_DOOR\_STATUS\_OPEN* = 2;  
  
public static final int *SUPPORT\_TIRE\_NO* = 0; //get tire temperature or pressure is not supported  
public static final int *SUPPORT\_TIRE\_TEMP\_PRESSURE* = 1; //get tire temperature or pressure is supported  
public static final int *SUPPORT\_TIRE\_PRESSURE* = 2; //only get tire pressure is supported  
  
public static final int *TIRE\_FRONT\_LEFT* = 0;  
public static final int *TIRE\_FRONT\_RIGHT* = 1;  
public static final int *TIRE\_REAR\_LEFT* = 2;  
public static final int *TIRE\_REAR\_RIGHT* = 3;  
  
public static final int *Tire\_Press\_LF\_Stat* = 0;//左前  
public static final int *Tire\_Press\_RF\_Stat* = 1;//右前  
public static final int *Tire\_Press\_RR\_ORR\_Stat* = 2;//  
public static final int *Tire\_Press\_LR\_OLR\_Stat* = 3;//  
public static final int *Tire\_Press\_IRR\_Stat* = 4;//  
public static final int *Tire\_Press\_ILR\_Stat* = 5;//  
  
public static final int *Tire\_Press\_LF\_Data* = 0;  
public static final int *Tire\_Press\_RF\_Data* = 1;  
public static final int *Tire\_Press\_RR\_ORR\_Data* = 2;  
public static final int *Tire\_Press\_LR\_OLR\_Data* = 3;  
public static final int *Tire\_Press\_IRR\_Data* = 4;  
public static final int *Tire\_Press\_ILR\_Data* = 5;

public static final int *WINDOW\_POS\_DRIVER* = 0;  
public static final int *WINDOW\_POS\_FRONT\_PASSENGER* = 1;  
public static final int *WINDOW\_POS\_REAR\_DRIVER* = 2;  
public static final int *WINDOW\_POS\_REAR\_PASSENGER* = 3;  
  
public static final int *DOOR\_FRONT\_LEFT* = 0;  
public static final int *DOOR\_FRONT\_RIGHT* = 1;  
public static final int *DOOR\_REAR\_LEFT* = 2;  
public static final int *DOOR\_REAR\_RIGHT* = 3;  
public static final int *DOOR\_OUTSIDE\_TRUNK* = 4;  
public static final int *DOOR\_INSIDE\_TRUNK* = 5;

# com.ford.car.FordCarHvacManager.java

## public static FordCarHvacManager getInstance(Context context)获取Manager单例。

## boolean isCarFrontClimateOpen()前排空调是否打开。

返回值，true：打开，false：关闭

## boolean isCarRearClimateOpen()后排空调是否打开。

返回值，true：打开，false：关闭

## boolean isSupportAAR()是否有AAR功能。

返回值，true：有，false：无

## int getCarClimateCycleMode()获取空调循环模式。

返回值，0是外循环  1是内循环也就是1是内循环时开 ，非1时关

can信号返回的值如下：

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## void setCarClimateCycleMode(int mode)设置空调循环模式。

参数，1：内循环，2：外循环

**public static final int *CLIMATE\_CYCLE\_MODE\_INNER*** = 1;  
**public static final int *CLIMATE\_CYCLE\_MODE\_OUTER*** = 2;

## void setCarFrontClimateOnOff(boolean isOpen) 设置前排空调开关。

参数，true：打开空调，false：关闭空调

## void setCarRearClimateOnOff(boolean isOpen)设置后排空调开关。

参数，true：打开空调，false：关闭空调

## addListener(ClimateChangeListener listener)设置空调状态变化的监听器。

**public interface** ClimateChangeListener {  
  
 */\*\*  
 \* Called when the climate's open status has changed.  
 \** ***@param postion*** *which climate's open status has changed.  
 \* 0: the front climate  
 \* 1: the rear climate  
 \** ***@param isOpen*** *climate's open status value.  
 \* 0: open  
 \* 1: close  
 \** ***@param operation*** *How did the user turn the climate on/off  
 \* 0: use app on the screen, default value.  
 \* 1: use hardKey on the car.  
 \*/* **void** onOpenStatusChange(**int** postion, **boolean** isOpen, **int** operation);  
  
 */\*\*  
 \* Called when the climate's cycle mode has changed.  
 \** ***@param mode*** *value after mode change  
 \*/* **void** onCycleModeChange(**int** mode);  
  
 */\*\*  
 \* Called when the climate's change has error.  
 \** ***@param errorCode*** *error code  
 \* 100: the front climate open error  
 \* 101: the front climate close error  
 \* 110: the rear climate open error  
 \* 111: the rear climate close error  
 \* 200: the climate cycle mode change error  
 \*/* **void** onError(**int** errorCode);  
}

## removeListener(ClimateChangeListener listener)移除添加的listener。

## clearListeners()清空listener集合。

## int getNewAirRefresh()获取aar得开关状态

## int getGlobalProperty(int type)当前时刻的PM2.5值

参数.int type:为要获取数据类型

TYPE\_CCM\_MESSAGE\_MISSING=0--空调发出的携带PM2.5传感器（滤芯）数据的CAN信号是否有效

TYPE\_PM\_MESSAGE\_MISSING=1--空调开启和关闭触发的CAN信号是否有效

TYPE\_PMCABN\_CONC\_ACTL=2--当前PM2.5的值

TYPE\_PMSNSCABN\_D\_STAT=3--PM2.5传感器的状态

## interface AARChangeListener

**{**

**void onChangeEvent(int eventType, int[] intValues, float[] floatValues);**

**}  
当前时刻PM2.5变化的回调**

参数说明：

1.int eventType

TYPE\_CCM\_MESSAGE\_MISSING=0x00--空调发出的携带PM2.5传感器（滤芯）数据的CAN信号是否有效

TYPE\_PM\_MESSAGE\_MISSING=0x01--空调开启和关闭触发的CAN信号是否有效

TYPE\_PMCABN\_CONC\_ACTL=0x02--当前PM2.5的值

TYPE\_PMSNSCABN\_D\_STAT=0x03--PM2.5传感器的状态

2.int[] intValue:

intValue[0]:几个类型信号的值

## int getGlobalProperty1(int type) X分钟前的PM2.5值

参数：int type:为要获取数据类型

TYPE\_PMCABN02MNTE\_CONC\_ACTL=0x10--2分钟之前的PM2.5值

TYPE\_PMCABN04MNTE\_CONC\_ACTL=0x11--4分钟之前的PM2.5值

TYPE\_PMCABN06MNTE\_CONC\_ACTL=0x12--6分钟之前的PM2.5值

TYPE\_PMCABN08MNTE\_CONC\_ACTL=0x13--8分钟之前的PM2.5值

TYPE\_PMCABN10MNTE\_CONC\_ACTL=0x14--10分钟之前的PM2.5值

TYPE\_PMCABN12MNTE\_CONC\_ACTL=0x15--12分钟之前的PM2.5值

TYPE\_PMCABN14MNTE\_CONC\_ACTL=0x16--14分钟之前的PM2.5值

TYPE\_PMCABN16MNTE\_CONC\_ACTL=0x17--16分钟之前的PM2.5值

TYPE\_PMCABN18MNTE\_CONC\_ACTL=0x18--18分钟之前的PM2.5值

TYPE\_PMCABN20MNTE\_CONC\_ACTL=0x19--20分钟之前的PM2.5值

## void setGlobalProperty(int type, int value) 发送当前空气过滤状态发送当前空气过滤状态

参数：

1.int type:TYPE\_PMCABN\_D\_STAT=0x20--发送空气状态

2.int vale：空气状态

发送当前PM2.5的等级

1.int type:TYPE\_PMCABNLVL\_D\_STAT=0x21--发送PM2.5的等级

2.int vale:PM2.5的等级

PmCabn\_D\_Stat

|  |  |  |
| --- | --- | --- |
| 0x0 | NotKnown | PmCabn\_D\_Stat (Msg:0x227 APIM\_Request\_Signals\_1) |
| 0x1 | FilteringOff |
| 0x2 | FilteringOn |
| 0x3 | FilteringComplete |
| 0xFF | Not Update |

PmCabnLvl\_D\_Stat

|  |  |  |
| --- | --- | --- |
| 0x0 | NotKnown | PmCabnLvl\_D\_Stat (Msg:0x227 APIM\_Request\_Signals\_1) |
| 0x1 | Level\_1\_Best |
| 0x2 | Level\_2 |
| 0x3 | Level\_3 |
| 0x4 | Level\_4 |
| 0x5 | Level\_5 |
| 0x6 | Level\_6\_Worst |
| 0x7 | NotUsed\_1 |
| 0xFF | Not Update |

## void setNewAirRefresh()开启/关闭座舱新风按钮

## int getAcConfig()空调是否可用

返回值：

HAVAC\_CONFIG\_NOT\_AVAILABLE = 0--空调不可用

HAVAC\_CONFIG\_AVAILABLE = 1--空调可用

## void onConnect(FordBaseManager.ConnectionCallback connection) 同步数据返回，需要先连接服务

## void onDisConnect()断开服务的连接

## interface INewAirRefresh{

**void onNewAirRefreshChange(int aar\_btn);}**

座舱新风的回调

## void setAutoPress()设置前auto

## int getAutoButtState(int front\_AUTO) int front\_AUTO:

**public static final int *Front\_AUTO\_Btn\_Stt* = 0;//高亮  
public static final int *Front\_AUTO\_Blwr\_Lvl* = 1;//3个杠**

**获取的2种auto btn的状态**

## interface IAutoButtState { void onAutoButtStateChange(int btn, int state);}auto的回调

**int btn;**

**public static final int *Front\_AUTO\_Btn\_Stt* = 0;//高亮  
public static final int *Front\_AUTO\_Blwr\_Lvl* = 1;//3个杠**

**int state：****e;**

## void setWindSpeed(int speed) 设置风量

## int getWindSpeed() 获取风量的值

## interface IWindSpeedChange {

**void onWindpeedChange(int speed);}风量的值的回调**

## void setPreSeatHeatingGearSwitchStatus(int status) 设置前座椅的挡位关闭：

**public static final int LHS\_SEAT\_OFF\_PRESSED = 0x26;**

**设置前座椅的挡位：**

**public static final int LHS\_HTD\_SEAT1\_PRESSED = 0x27;**

**public static final int LHS\_HTD\_SEAT2\_PRESSED = 0x28;**

**public static final int LHS\_HTD\_SEAT3\_PRESSED = 0x29;**

## void setRearSeatHeatingGearSwitchStatus(int status) 设置后座椅的挡位关闭：

**public static final int RHS\_SEAT\_OFF\_PRESSED = 0x2B;**

**设置前座椅的挡位关闭：**

**public static final int RHS\_HTD\_SEAT1\_PRESSED = 0x2C;**

**public static final int RHS\_HTD\_SEAT2\_PRESSED = 0x2D;**

**public static final int RHS\_HTD\_SEAT3\_PRESSED = 0x2E;**

## void setPreSeatHeatingSwitchStatus() 设置前座椅加热开关状态

## int getPreSeatHeatingSwitchStatus() 获取前座椅加热开关的状态

## interface IPreSeatHeatingSwitchStatusChange {

**void onPreSeatHeatingSwitchStatusChange(int state);}**

**前座椅加热开关状态的回调**

## void setRearSeatHeatingSwitchStatus() 设置后座椅加热的开关状态

## int getRearSeatHeatingSwitchStatus()

获取后座椅加热的开关状态

## interface IRearSeatHeatingSwitchStatusListener {

**void onRearSeatHeatingSwitchStatusChange(int state);}**

**后座椅加热状态的回调；**

## void setSteeringWheelHeating() 设置方向盘加热的开关状态

## int getSteeringWheelHeating() 获取加热方向盘开关的状态

## elHeatingChange {

**void onSteeringWheelHeatingChange(int state);}方向盘加热开关的回调**

## void setDefrost() 设置除雾的状态

## int getDefrost() 获取除雾的状态

## interface IDefrostChange {

**void onDefrostChange(int state);}除雾状态的回调**

## void setLHS\_Cld\_Seat\_Pressed()主驾制冷

## int getLHS\_Cond\_Seat\_Status() 主驾制冷状态

## interface ILHS\_Cond\_Seat\_StatusChange {

**void onLHS\_Cond\_Seat\_StatusChange(int status);}主驾制冷状态的回调**

## void setRHS\_Cld\_Seat\_Pressed()副驾制冷

## int getRHS\_Cond\_Seat\_Status() 副驾制冷的状态

## interface IRHS\_Cond\_Seat\_StatusChange {

**void onRHS\_Cond\_Seat\_StatusChange(int status);}副驾制冷状态的回调**

## int getDualButtonStatus() 双开模式

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## void setNone\_Pressed() None\_Pressed开关

## void setPanel\_Pressed() 吹脸

## int getPanel\_Btn\_Stt() 吹脸的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IPanel\_Btn\_SttListener { 吹脸状态的回调

**void onPanel\_Btn\_SttChange(int value);}**

## void setPanel\_Floor\_Pressed() 吹脸+地板

## void setFloor\_Pressed() 地板

## int getFloor\_Pressed() 吹地的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IFloor\_PressedListener { 吹地的状态的回调

**void onFloor\_PressedChange(int value);}**

## setWindscreen\_Pressed（）前挡

## int getWindscreen\_Btn\_Stt() 挡风返回的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IWindscreen\_Btn\_SttListener { 挡风返回状态的回调

**void onWindscreen\_Btn\_SttChange(int value);}**

## void setFloor\_Ws\_Pressed() 地板+前挡

## void setAC\_Pressed() 制冷

## int getAC\_Btn\_Stt() 制冷返回的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IAC\_Btn\_SttListener { 制冷返回状态的回调

**void onAC\_Btn\_SttChange(int value);}**

## void setLHS\_Temp\_Inc\_Pressed() 主温+

## void setLHS\_Temp\_Dec\_Pressed() 主温-

## void setRHS\_Temp\_Inc\_Pressed() 副温+

## void setRHS\_Temp\_Dec\_Pressed() 副温-

## void setBlwr\_Inc\_Pressed() 风量+

## void setBlwr\_Dec\_Pressed() 风量-

## void setMax\_AC\_Pressed() 最大制冷

## int getMax\_AC\_Btn\_Stt() 最大制冷返回的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IMax\_AC\_Btn\_SttListener { 最大制冷的回值

**void onMax\_AC\_Btn\_SttChange(int value);}**

## void setRear\_Defrost\_Pressed() 后除霜

## int getRear\_Defrost\_Btn\_Stt() 后除霜的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IRear\_Defrost\_Btn\_SttListener { 后除霜状态的回调

**void onRear\_Defrost\_Btn\_SttChange(int value);}**

## void setMax\_Defrost\_Pressed() 最大除霜

## int getMax\_Defrost\_Btn\_Stt() 最大除霜的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IMax\_Defrost\_Btn\_SttListener { 最大除霜状态的回调

**void onMax\_Defrost\_Btn\_SttChange(int value);}**

## float getAirConTemperature() 主驾的温度

## void setAirConTemperature(int temperature) 设置主 驾的温度

对应的temperature是：0-31 对应右边的温度

|  |  |
| --- | --- |
| 0 | No\_Request |
| 1 | LO |
| 2 | 15.5 |
| 3 | 16 |
| 4 | 16.5 |
| 5 | 17 |
| 6 | 17.5 |
| 7 | 18 |
| 8 | 18.5 |
| 9 | 19 |
| 10 | 19.5 |
| 11 | 20 |
| 12 | 20.5 |
| 13 | 21 |
| 14 | 21.5 |
| 15 | 22 |
| 16 | 22.5 |
| 17 | 23 |
| 18 | 23.5 |
| 19 | 24 |
| 20 | 24.5 |
| 21 | 25 |
| 22 | 25.5 |
| 23 | 26 |
| 24 | 26.5 |
| 25 | 27 |
| 26 | 27.5 |
| 27 | 28 |
| 28 | 28.5 |
| 29 | 29 |
| 30 | 29.5 |
| 31 | HI |

## interface IAirConTemperatureChange { 主驾温度的回调

**void onAirConTemperatureChange(float temperature);}**

## float getPsngrAirConTemperature() 副驾的温度

## void setPsngrAirConTemperature(int temperature) 设 置副驾的温度

设置的温度是0-31，对应右边的值

|  |  |
| --- | --- |
| 0x00 | No\_Request |
| 0x01 | LO |
| 0x02 | 15\_5 |
| 0x03 | 16\_0 |
| 0x04 | 16\_5 |
| 0x05 | 17\_0 |
| 0x06 | 17\_5 |
| …… | …… |
| 0x1E | 29\_5 |

## interface IPsngrAirConTemperatureChange { 副驾温度的回调

**void onPsngrAirConTemperatureChange(float temperature);}**

## int getLHS\_Cld\_Seat\_Btn\_Stt() 主驾座椅制冷的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface ILHS\_Cld\_Seat\_Btn\_SttListener { 主驾座椅制冷的回调

**void onLHS\_Cld\_Seat\_Btn\_SttChange(int value);}**

## int getRHS\_Cld\_Seat\_Btn\_Stt() 副驾座椅制冷的状态

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## interface IRHS\_Cld\_Seat\_Btn\_SttListener{ 副驾座椅制冷的回调

**void onRHS\_Cld\_Seat\_Btn\_SttChange(int value);}**

## void setDualButtonPressed()

## interface IDualButtonStatusChange {

**void onDualButtonStatusChange(int status);}**

|  |  |
| --- | --- |
| Enabled\_Inactive | 0x0 |
| Active | 0x1 |
| Disabled | 0x2 |
| Unused | 0x3 |

## setAutoGradePress(int grade) Auto的3个等级

grade:如下：

public static final int *AUTO\_Lo\_Pressed*=0x36; 低  
public static final int *AUTO\_Med\_Pressed*=0x37;中  
public static final int *AUTO\_Hi\_Pressed*=0x38;高

# com.ford.media.MediaEventManager

## USB广播

增加了USB扫描失败、设备不支持的广播Action，其余的USB插入、拔出、扫描等，使用原生广播Action即可。用法与原生保持一致。

*/\*\*  
 \* Broadcast Action: The media scanner has failed to scan a directory.  
 \* The path to the failure scanned directory is contained in the Intent.mData field.  
 \*/***public static final** String ***ACTION\_MEDIA\_SCANNER\_SCAN\_FAILED*** = **"ford.intent.action.MEDIA\_SCANNER\_SCAN\_FAILED"**;  
  
*/\*\*  
 \* Broadcast Action: The device is not supported  
 \* The path to the mount point for the unsupport device is contained in the Intent.mData field.  
 \*/***public static final** String ***ACTION\_MEDIA\_UNSUPPORT*** = **"ford.intent.action.MEDIA\_UNSUPPORT"**;

## public static FordMediaEventManager getInstance(Context context)

获取单例

## void sendVRTips(String text)

发送VR提示文本给车机底部导航栏显示。

## void sendID3Info(ID3Info id3Info)

将随心听播放的ID3信息，发送给仪表显示。ID3Info的字段定义，与百度《随心听交互协议\_3.8.pdf》一致。

## void sendPlayState(int state)

将随心听播放状态，发送给仪表显示。state定义，

**public static final int *PLAY\_STATE\_BUFFERING*** = 1; *//buffering***public static final int *PLAY\_STATE\_PLAYING*** = 2; *//play***public static final int *PLAY\_STATE\_PAUSE*** = 3; *//pause***public static final int *PLAY\_STATE\_STOP*** = 4; *//stop*

## void sendPlayMode(int mode)

将随心听播放模式，发送给仪表显示。mode定义：

**public static final int *PLAY\_MODE\_SINGLE\_CYCLE*** = 0; *//single cycle***public static final int *PLAY\_MODE\_SHUFFLE\_PLAY*** = 1; *//shuffle play***public static final int *PLAY\_MODE\_LIST\_LOOP*** = 2; *//list loop***public static final int *PLAY\_MODE\_ORDER\_PLAY*** = 3; *//order play*

## addListener(ActionListener listener)

增加listener。

## removeListener(ActionListener listener)

移除listener。

## clearListeners()

清空listener。

## 回调接口

**public interface** ActionListener {  
 **void** onAction(**int** action);  
}

当用户通过仪表控制多媒体时，会回调onAction方法，action参数定义：

**public static final int *ACTION\_PLAY*** = 0;*//播放***public static final int *ACTION\_PAUSE*** = 1;*//暂停***public static final int *ACTION\_PREVIOUS*** = 2;*//上一曲***public static final int *ACTION\_NEXT*** = 3;*//下一曲*

# com.ford.camera.EVSCameraManager

图像evs接口，请阅读《evc\_camera-api\_description-v2.1-20200303172110-kkovev.pdf》接口文档。

0 ：恢复正常

face id camera错误码:

1：led\_failure;

2：over\_temperature;

3：general\_signal\_failure;

4：missing\_message;

ar camera错误码

1： general\_signal\_failure;

2： over\_temperature;

3： not\_configured;

4： invalid\_serial\_data;

5： missing\_message;

# com.ford.vendor.FordEnhanceMemoryManager

em接口，按之前邮件excel表格定义的，请参考邮件。

## void connect(ConnectionCallback connection)em的同步连接方法，先连接service才有数据返回

## void disConnect()断开相关service的连接

## byte[] getPersOptInSt()

## void sendInfotainmentRecall\_Rq(byte persIndex)

*/\*\*  
 \* persIndex：驾驶员档案索引（1-4：用户创建的档案  
 \* 5：访客档案）  
 \*  
 \** ***@param*** *persIndex  
 \*/*

## interface IOnProfileActive {

void onProfileActive(int index);}

*/\*\*  
 \* 形参：驾驶员档案索引，合法值大于等于0  
 \*/*

## boolean getPaaKConnection\_St()

*/\*\*  
 \* 返回值：智能手机钥匙连接状态  
 \*  
 \** ***@return*** *\*/*

## interface IPaakConnectionChanged {

void onPaakConnectionChanged(boolean state);

*/\*\*  
 \* 形参列表：  
 \* state：智能手机钥匙连接状态，true：连接，false：未连接  
 \*/*

## void sendEnhancedMemory\_St(byte status)

## void sendEnMemProfilePairing\_Rq(byte persIndex, byte buttonPairing, byte keyPairing)驾驶员档案索引（1-4：用户创建的档案5：访客档案） 驾驶员档案关联智能手机钥匙 前提：智能手机钥匙已连接

## interface IEMButtonPairingSt {

void onEMButtonPairingStChanged(byte buttonPairing);

}

## void sendFeature\_Rq(byte operation, int featureID, int configuration, byte persIndex)

*/\*\*  
 \* persIndex：驾驶员档案索引（1-4：用户创建的档案  
 \* 5：访客档案）  
 \**

## void sendInfotainmentPersStore\_Rq(byte persIndex)

设置驾驶员得档案

void*\* persIndex：驾驶员档案索引（1-4：用户创建的档案  
\* 5：访客档案）*

## interface IInfotainmentPersStoreSt {

void onImPersStoreStChanged(byte status);

}

## void sendPersOptInSt(byte[] persStatus) 发送档案得选中状态

*\* 形参：  
\* 4个个性化档案的选中状态；  
\* true：OptedIn  
\* false：NotOptedIn；*

## interface IEnMemKeyPairingSt {

void onEMKeyPairingStChanged(byte persIndex, byte keyPairing);

}

## interface IPersPhonePairingSt {

void onPersPhonePairingStChanged(byte[] persPhoneStatus);

档案和手机得关联状态

}  
*\* 形参：  
\* 4个个性化档案的手机关联状态；*

## interface IPersKeyPairingSt {

void onPersKeyPairingStChanged(byte[] persKeyStatus);

车钥匙关联状态得回调

}*\* 形参：  
\* 4个个性化档案的车钥匙关联状态；*

## interface ISeatAndMirror {

void onSeatAndMirrorChange(byte type, int value);}

值的含义：

type:

public static final int TYPE\_SEAT = 0;//座椅

public static final int TYPE\_MIRROR = 1;//后视镜

value；

座椅后视镜对应的值

## boolean[] getPersPhonePairingSt()

手机和IVI的连接状态

## interface IOnGearChanged{

void onGearChanged(int gear);

形参列表：当前档位：

GEAR\_NEUTRAL = 0;

GEAR\_PARKING = 1;

GEAR\_REVERSE = 2;

GEAR\_DRIVE = 3.

} 档位变化的回调

## int getGear();

返回值：当前档位：-2是默认得无效值

GEAR\_NEUTRAL = 0;

GEAR\_PARKING = 1;

GEAR\_REVERSE = 2;

GEAR\_DRIVE = 3.  
获取档位

## List<RadioStation> getCollectRadios()

收藏电台的读取，最多18个电台

## void setCollectRadios(List<RadioStation> radios)

形参：收藏电台的集合  
收藏电台的设置

## interface ICollectedRadiosChange{

void onCollectedRadioChange(List<RadioStation> radios);

形参：收藏电台的集合

} 收藏电台变化的回调

## List<RadioStation> getAutoSavedRadios(int radioBrand) 自动存储电台的读取，最多18个电台 radioBrand:0:AM 1FM

## void setAutoSavedRadios(int radioBrand,List<RadioStation> radios)

形参：自动存储电台的集合  
自动存储电台的设置  
radioBrand:0am 1Fm

## interface IAutoSavedRadiosChange{

void onAutoSavedRadiosChange(List<RadioStation> radios);

形参：自动存储电台的集合

} 自动存储电台变化的回调

## int getTheme() 当前主题得读取 主题代表的int :0 1 2 对应的如下：2是默认主题；

private List<ThemeBean> beanList = Arrays.*asList*(  
 new ThemeBean(0, "静谧之旅",  
 "com.desay.setting.overlay.golden",  
 "com.desay.setting"),  
 new ThemeBean(0, "静雅天成",  
 "com.desay.setting.overlay.green",  
 "com.desay.setting"),  
 new ThemeBean(0, "默认主题",  
 "com.desay.setting.overlay",  
 "com.desay.setting")  
);

## void setTheme(int Index)

主题得设置 具体如上；

## interface IThemeChange{

void onThemeChange(int theme);

} 主题变化的回调

## int getDistanceUnit() 获取距离单位

## void setDistanceUnit(int unit) 设置距离单位

## interface IDistanceUnitChange{

void onDistanceUnitChange(int unit);  
距离单位的回调

## int getTemperatureUnit() 获取温度单位

## void setTemperatureUnit(int unit) 设置温度单位

## interface ITemperatureUnitChange{

void onTemperatureUnitChange(int unit);  
温度单位的回调

## int getTyreUnit() 获取胎压单位，默认值kPa

## void setTyreUnit(int tyreUnit) 设置胎压单位

## interface ITyreUnitChange{

void onTyreUnitChange(int tyreUnit);  
胎压单位的回调

## int getMeasureUnit() 获取测量单位，默认值是km&km/l

## void setMeasureUnit(int measureUnit ) 设置测量单位

## interface IMeasureUnitChange{

void onMeasureUnitChange(int measureUnit);  
测量单位的回调

## int getAmbientLampSwitch() 获取氛围灯开关 0是关 对应的can信号如下；



## void setAmbientLampSwitch(int switch) 设置氛围灯开关

## interface IAmbientLampSwitchChange{

void onAmbientLampSwitchChange(int swicth);  
氛围灯开关变化的回调

## int getAmbientColor() 获取氛围灯颜色

## void setAmbientColor(int color ) 设置氛围灯颜色

## interface IAmbientColorChange{

void onAmbientColorChange(int color);  
氛围灯颜色变化的回调

## 10.51 int getAmbientBrightness() 获取氛围灯亮度

## void setAmbientBrightness(int brightness) 设置氛围灯亮度

## interface IAmbientBrightnessChange{

void onAmbientBrightnessChange(int brightness);  
氛围灯亮度变化的回调

## boolean getReverseImageDelay() 获取倒车影像延迟

## void setReverseImageDelay(boolean delay) 设置倒车影像延迟

## interface IReverseImageDelayChange{

void onReverseImageDelayChange(boolean delay);

}倒车影像延迟变化的回调

## int getSpeedVolCompensation() 获取车速音量补偿设置

## void setSpeedVolCompensation(int compensation) 设置车速音量补偿设置

## interface ISpeedVolCompensationChange{

void onSpeedVolCompensationChange(int compensation);  
车速音量补偿设置变化的回调

## int getAirConTemperature() 获取空调温度

## void setAirConTemperature(int temperature) 设置空调温度

## interface IAirConTemperatureChange{

void onAirConTemperatureChange(int temperature);  
空调温度变化的回调

## void setDriveMode(int mode) 设置驾驶模式

## interface IPersRecallCountSt {

void onPersRecallCountChange(int count);

}*count:bcm中档案被操作的次数*

## int getProfileActive()

## void setFreqBand(RadioStation radiostation) 设置最后一次的brand

## RadioStation getFreqBand()获取最后一次的brand

## interface IFreqBandChange{

**void onFreqBandChange(RadioStation radioStation);}**

**最后一次记录电台的回调**

## float getPsngrAirConTemperature()

获取副驾得温度

## void setPsngrAirConTemperature(int temperature)

设置副驾得温度

## interface IPsngrAirConTemperatureChange {

**void onPsngrAirConTemperatureChange(float temperature);}**

**副驾温度得回调**

## int getAmbientlightmode()

**当前Ambient灯光的模式**

**0 Manual**

**1 Automatic**

绑定情况你去调切换SDM的CAN信号，1是绑定的状态

## interface IAmbientlightmodeChange {

**void onAmbientlightmodeChange(int mode);}**

绑定情况你去调切换SDM的CAN信号，1是绑定的状态

## int[] getAmbientLightingMode()

**//返回氛围灯模式 数组，长度为2，参数1 （1静态模式，2 动态颜色 ，3 自定义颜色，4.music）， 参数2（此模式下，返回自定义颜色）**

**0x00 invalid 0x10 AmbL\_Color\_Mode**

**0x01 static**

**0x02 dynamic**

**0x03 customize**

**0x04 music**

**0xFF NoRequest**

**静态/动态颜色设置**

**Can信号名: 0x10 AmbL\_Static\_ColorValue\_Set**

**动态can信号：0x10 AmbL\_Dynamic\_Color**

**Value:(0x0-0x7F 0-127 )**

**个性化设置:**

**Can信号名 0x16 AmbL\_CustomizeType\_Set**

**Value:**

**0x00 Invalid**

**0x01 Customize1**

**0x02 Customize2**

**0x03 Customize3**

**0xFF NoRequest**

## void setAmbientLightingMode(int[] mode)

**参数为2；参数1 颜色模式 参数2：模式对应的颜色**

**同11.72**

## interface IAmbientLightingModeChange {

**void onchangeALMode(int[] mode);}**

**返回氛围灯模式的回调**

**同11.72**

## int getAmbientLightingValue()

**0x0-0x64 0-100(%) 0x10 AmbL\_Main\_Intensity\_Set**

## public void setAmbientLightingValue(int value)

**同11.75 0x0-0x64 0-100(%) 0x10 AmbL\_Main\_Intensity\_Set**

## public interface IAmbientLightingValueChange {

**void onchangeALValue(int value);**

**}**

**同11.75 0x0-0x64 0-100(%) 0x10 AmbL\_Main\_Intensity\_Set**

## public int[] getCarFragranceSetting()

**参数个数是5个：参数1:工作状态 参数2：通道1 参数3:通道2 参数4：通道3： 参数4：浓度**

**工作状态：**

|  |  |
| --- | --- |
| 0 | 0x0:Off |
| 1 | 0x1:Channel 1 Working |
| 2 | 0x2:Channel 2 Working |
| 3 | 0x3:Channel 3 Working |

**通道1：**

|  |  |
| --- | --- |
| 0 | 0:未知的 |
| 1-253 | 1-253:正确的香氛类型ID |
| 254 | 254:未认证的/无效的 |
| 255 | 255:未安装 |

**通道2：**

|  |  |
| --- | --- |
| 0 | 0:未知的 |
| 1-253 | 1-253:正确的香氛类型ID |
| 254 | 254:未认证的/无效的 |
| 255 | 255:未安装 |

**通道3：**

|  |  |
| --- | --- |
| 0 | 0:未知的 |
| 1-253 | 1-253:正确的香氛类型ID |
| 254 | 254:未认证的/无效的 |
| 255 | 255:未安装 |

**浓度;**

|  |  |
| --- | --- |
| 0 | 0x0= Unknown |
| 1 | 0x1= Off |
| 2 | 0x2= Reserved |
| 3 | 0x3= Low Intensity |
| 4 | 0x4= Reserved |
| 5 | 0x5= Medium Intensity |
| 6 | 0x6= Reserved |
| 7 | 0x7= High Intensity |

## public void setCarFragranceSetting(int[] i)

**//车载香氛设置项 数组，长度为3 参数1 总开关(ture flase)，参数2 香型 参数3 浓度**

**开关：**

|  |  |
| --- | --- |
| 0 | 0x0:初始状态 |
| 1 | 0x1:香氛开始运行 |
| 2 | 0x2:香氛停止工作 |
| 3 | 0x3: 主节点异常 |

**香型：**

|  |  |
| --- | --- |
| 0 | 0x0:香氛通道关闭 |
| 1 | 0x1:开启通道1的香氛 |
| 2 | 0x2:开启通道2的香氛 |
| 3 | 0x3:开启通道3的香氛 |

**浓度：**

|  |  |
| --- | --- |
| 0 | 0x0:未知 |
| 1 | 0x1:香氛浓度关闭 |
| 2 | 0x2:预留 |
| 3 | 0x3:开启的通道对应的香氛浓度为低浓度 |
| 4 | 0x4:预留 |
| 5 | 0x5:开启的通道对应的香氛浓度为中浓度 |
| 6 | 0x6:预留 |
| 7 | 0x7:开启的通道对应的香氛浓度为高浓度 |

## public interface ICarFragranceSetting {

**void onchangeCFSetting(int[] i);**

**}**

**参数个数是5个：参数1:工作状态 参数2：通道1 参数3:通道2 参数4：通道3： 参数4：浓度**

**同11.79**

## public int getInstrumentMainSetting()

绑定情况你去调切换SDM的CAN信号，不绑定的情况你直接调用这个

**本地记录主题index:经典单:5;运动:0;节能:1;经典双:2**

## public void setInstrumentMainSetting(int index)

绑定情况你去调切换SDM的CAN信号，不绑定的情况你直接调用这个

## public interface InstrumentMainSetting {

**void onchangeIM(int i);**

**}**

绑定情况你去调切换SDM的CAN信号，不绑定的情况你直接调用这个

## void setFeat2ConfigActl(int type)

**0x09C0信号对应主题：经典单：2；运动：3；节能：4;经典双：5**

## void getFeat2ConfigIpc2Actl()

**0x09C0信号对应主题：经典单：2；运动：3；节能：4;经典双：5**

## interface IFeat2ConfigIpc2Actl {

**void onFeatConfigIpc2ActlCommon(int result);**

**}**

# com.ford.vendor.FordEVChargeManager

EV接口，按之前邮件excel表格定义的，请参考邮件。

## byte getPlugConnectStatus();获取车与充电桩的连接状态

## interface IPlugConnectStatus{

void onPlugConnectStatus(boolean status){

status:true：连接

false：未连接}车与充电桩的连接状态的回调

## boolean isChargePlanPos();判断车辆位置是否在充电计划的位置

返回值：当前位置是否在充电计划位置的状态，

true：当前位置在充电计划内；

false：当前位置不在充电计划内

## IVehiclePosChange {

void onVehiclePosChange(byte posId){

posId：Saved Charge Location ID，0：不在收藏充电位置，1-10：在收藏的充电位置

} 车辆位置变化的回调

## void setChargPlanSwitch(boolean on);

形参:充电计划的开关，true：开启，false：关闭

打开或者关闭当前位置的充电计划

## float getChargePercentage();

返回值：电池当前电量百分比值，单位为“%”，值为截尾数据，例：97.5% -> “97%”

获取电池当前电量百分比值

## interface IChargeProcessChange{

void onChargeProcessChange(float process){

process:电池当前电量百分比值，单位为“%”，值为截尾数据，例：97.5% -> “97%”。

} 电池当前电量百分比值变化的回调

## byte getChrgStat();

返回值：0: Not Ready,7:charging

获取车的充电状态

## interface IChrgStatChange{

void onChrgStatChange(byte state){

state:车的充电状态

} 车的充电状态的回调

## int getRemainingMileage();

返回值：剩余里程值，单位为“km”

获取剩余里程，若有CE-DTE的数据，优先展示CE-DTE为剩余里程，

否则，展示普通DTE

## float getfullyChargeHours();

返回值：充满电预测所用小时数，单位是小时，取数字小数点后一位

获取充满电预测所用小时数

前提：无连接充电桩\在已收藏充电位置\无使用或打开充电计划\电池电量<充电目标电量，否则返回0.0

## byte getChargeTargetPercent();

返回值：充电的目标电量百分比，单位为“%”

获取充电的目标电量百分比

前提：当前位置在充电计划的位置\充电计划开启，返回充电计划的设置数据，否则返回默认值；

## ChargeStartTime getChargeStartTime();

形参：充电开始时间，

获取充电开始时间

## ChargeEndTime getChargeEndTime();

返回值：充电结束时间，

获取充电结束时间

## List<DepartAndComfort> getDepartAndComfort();

返回值：出发时间及温度水平，DepartAndComfort

获取最近的出发时间及温度水平

前提：出发时间设置开关开启且设置了出发时间，返回最近的出发时间，否则返回null；

## boolean isVehicleRun();

返回值：获取车辆的运转状态，true：运行；false：不运行

获取车辆的运转状态（点火开关）

## interface IVehicleRunChange{

void onVehicleRunChange(boolean state){}

车辆的运转状态的回调

## List<ChargeLocation> getUnsavedChargeLocations();

返回值：未保存的历史充电位置的列表，至多返回最近的10个充电位置信息，按时间倒序排列

获取未保存地址列表

## void deleteUnSavedChargeLocation(int id)

删除未保存充电位置 id:未保存位置得id

## public interface IChargePlanRX

void onChargePlan(List<ChargePlan> chargeplans);

//（desay返回ChargePlan，针对modify或者其他）  
 充电计划有变化返回的回调

}

## void saveDepartAndComforts(List<DepartAndComfort> departAndComforts);

形参：出发时间和温度水平的数据，

保存出发时间和温度水平的数据（一个或者多个）

## interface IChargeAndDepartConflict{

void onChargeAndDepartConflict(int type);

0:Charging outside time window（在充电计划预设时间段外充电）

1:Not reaching desired level（无法完成充电至目标电量）

}

冲突提醒的回调

## void disConnection() 取消相关service的连接

## interface IRemainMileageListener{

void onRemainingMileage(int mileage){

mileage:剩余里程值

}

剩余里程值回调

## byte getBattLowWarning();

返回值：0: NULL,1:LoDte,2:ZeroDteDepletedBattery,3:Not used  
获取车辆低电池警告

## interface IBattLowWarning{

void onBattLowWarning(int type);

0:NULL

1:LoDte（低电量）

2:ZeroDteDepletedBattery (电量为0)

3:Not used

} 车辆低电池警告回调

## void sendOnbChrgPrflUpdate\_B\_Rq(byte requesting)

*\* apim与hpcm、ecg的请求标志  
\*  
\** ***@param*** *requesting 形参：请求的标志， 0= No Request  
\* 1 = Request  
\* OnbChrgPrflUpdate\_B\_Rq*

## void sendOnbChrgLocIdTrgt\_No\_Rq(int locId)

*/\*\*  
 \** ***@param*** *locId apim通知hpcm更新的收藏位置id  
 \*/*

## void setUnsavedLocationSettingFlag(boolean flag)

*/\*\*  
 \* 未保留位置的收藏标志  
 \*  
 \** ***@param*** *flag  
 \*/*

## byte getVehiclePosId();

返回值：Saved Charge Location ID，0：不在收藏充电位置，1-10：在收藏的充电位置

## interface IUnsavedAckRec {

void onUnsavedAckRec(boolean flag);

}未保留位置收藏的ack回调

## interface IChargeLocationInfoReq {

void onChargeLocationInfoReq(int reqType, List<EvChargeLocationRecord> info);

}

Ecg端昵称更改的上报与昵称列表的请求

## void transmitLocationNamesList(int reqType, List<EvChargeLocationRecord> infos) { apim端下发昵称列表

## boolean getDepartAndComfortSwitch() 获取出发时间设置开关

## int getFastChargeEndMinutes(byte canType) *\* 获取快速充电用时 \* 当canType==0，使用信号FstChrgBulk\_T\_Est； \* 当canType==1，使用信号FstChrgCmplt\_T\_Est；*

## void sendUnsavedLocationSettingFlag(boolean flag) 发送未保存位置的flag

## void sendOnbChrgGoTUpdate\_B\_Rq(int request,int id)

发送位置更新的请求

## interface IFullyChargeHoursListener {

void onFullyChargeHoursChange(float hours);

}  
*/\*\*  
 \* 形参：充满电预测所用小时数，单位是小时，取数字小数点后一位  
 \*/*

## interface IChargeTargetPercentListener {

void onChargeTargetPercentChange(byte percent);

}  
*/\*\*  
 \* 返回值：充电的目标电量百分比，单位为“%”  
 \*/*

## interface IChargeStartTimeListener {

void onChargeStartTimeChange(EvPlugCharge\_ChargeStartTime\_Rsp startTime);

}*/\*\*  
 \* 形参：充电开始时间，见sheet-ChargeStartTime  
 \*/*

## interface IChargeEndTimeListener {

void onChargeEndTimeChange(EvPlugCharge\_ChargeEndTime\_Rsp endTime);

}

*/\*\*  
 \* 形参：充电结束时间，见sheet-ChargeEndTime  
 \*/*

## interface IDepartAndComfortListener {

void onDepartAndComfortChange(List<DepartAndComfort> depart);

}

*/\*\*  
 \* 形参：出发时间及温度水平，DepartAndComfort见sheet-DepartAndComfort  
 \*/*

## void connect(ConnectionCallback connection) 同步service的连接状态

## void disConnect() 断开service的连接

## int getDistanceUnit()

获取距离单位

## interface IDistanceUnitChange {

void onDistanceUnitChange(int unit);

距离单位得回调

## float getNxtUsgSocEst()

获取出发时间预估电量

## List<ChargePlan> getChargePlan()

获取充电计划

## List<DepartAndComfort> getDepartAndComforts()

获取出发时间

## interface IUnSavedChargeLoaction {

void onUnSavedChargeLocation(List<ChargeLocation> unSavedChargeLocation);

}

获取未保存充电位置

## interface INextDepartAndComfort {

void onNextDepartAndComfort(int nextId);

}获取下一次出发时间id

## void sendUnSavedLocationReq(int id, int updateRq, int deleteRq, int unsavedRq)

未保存充电位置

## void sendChargePlanReq(ChargePlan chargePlan) 保存充电计划

## int getNtfCtnConflictRq()

获取充电冲突状态

## int getNextDepartId()

下次出发时间

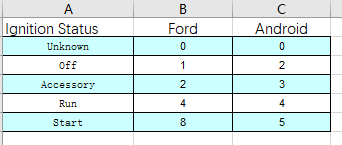
## interface IIgnitionStChange {

**void onIgnitionStChange(int state);}**

## int getIgnitionStatus()

获取车辆点火信号Ignition\_Status

点火状态给到java app，走的是Android原生的carsensormanager，所以需要将福特的点火状态与Android carsensormanager定义的点火状态一一映射，映射关系如下：



## float getRngPerChrgAvg\_L2\_Dsply()满电里程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0xFFFF | 0 | Factor: 0.1 Offset: 0 | RngPerChrgAvg\_L2\_Dsply (Msg: 0x37D GWM\_Send\_Signals\_13\_HS3) |
| 1 |
| … | … |
| 0x1FFE | NoDataExists |
| 0x1FFF | Faulty |

## interface IRngPerChrgAvg\_L2\_Dsply{//满电里程

void onRngPerChrgAvg\_L2\_DsplyChange(float RngPerChrgAvg\_L2\_Dsply);}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0xFFFF | 0 | Factor: 0.1 Offset: 0 | RngPerChrgAvg\_L2\_Dsply (Msg: 0x37D GWM\_Send\_Signals\_13\_HS3) |
| 1 |
| … | … |
| 0x1FFE | NoDataExists |
| 0x1FFF | Faulty |

## sendDepartAndComforts(List<DepartAndComfort> departAndComforts

## int getBattTracLoMsgTxt\_D\_Rq() 低电量续航

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0x07 | 0 | Null | BattTracLoMsgTxt\_D\_Rq (Msg: 0x311 Cluster\_HEV\_Dat7\_HS3) |
| 1 | LowDteFirstAlert |
| 2 | LowDteSecondAlert |
| 3 | ClimateLowDteFirstAlert |
| 4 | ClimateLowDteSecondAlert |
| 5 | ZeroDteBatteryDepleted |
| 6 | NotUsed\_1 |
| 7 | NotUsed\_1 |
|  |  |

## interface IBattTracLoMsgTxt\_D\_Rq{

**void onBattTracLoMsgTxt\_D\_RqChange(int BattTracLoMsgTxt\_D\_Rq);}**

同12.61；

## int getFstChrgCmplt\_T\_Est() 充满100%完成时间

|  |  |  |
| --- | --- | --- |
|  | 0-1023min | FstChrgCmplt\_T\_Est 0x238, GWM\_Send\_Signals\_18\_HS3 |
|  |  |

## interface IFstChrgCmplt\_T\_EstListener{

**void onFstChrgCmplt\_T\_EstChange(int FstChrgCmplt\_T\_Est);}**

## float getEstmChrgTimeHP\_St() 充满100%预计完成时间 大功率

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0x479(GWM\_HPCM\_17\_HS3) EstmChrgTimeHP\_St 2. 单位：小时 |
| 1 | 0.1 |
| 2 | 0.2 |
| … | … |
| 239 | 23.9 |
| 240 | 24 |
| 241…254 | Reserved |
| 255 | Invalid |
|  |  |

## interface IEstmChrgTimeHP\_StListener{

**void onEstmChrgTimeHP\_StChange(float EstmChrgTimeHP\_St);}**

充满100%预计完成时间 大功率

## float getEstmChrgTimeLP\_St()充满100%预计完成时间 低功率

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0x479(GWM\_HPCM\_17\_HS3) EstmChrgTimeLP\_St 2. 单位：小时 |
| 1 | 0.1 |
| 2 | 0.2 |
| … | … |
| 239 | 23.9 |
| 240 | 24 |
| 241…254 | Reserved |
| 255 | Invalid |
|  |  |

## interface IEstmChrgTimeLP\_StListener{

**void onEstmChrgTimeLP\_StChange(float EstmChrgTimeLP\_St);}**

充满100%预计完成时间 低功率

# com.ford.vendor.FordCeDTEManager

## void setTripInfoStructure(SoaCedteTripInfo soaCedteTripInfo)发送cedte的数据

## interface ISoaDTEListener {

**void Callback\_SoaCedteRefreshReq();}**

cedte回来的回调；

# com.ford.ea.FordEmergencyManager

## void sendReqWithRsp(BaseReqMessage baseReqMessage)

发送紧急救援相关的参数并带返回消息

void sendRequestNoRsp(BaseReqMessage baseReqMessage)

发送紧急救援相关的参数不带返回消息

BaseReqMessage 救援相关的参数

## int getCrashCan() 获取获取车辆的Crash信号

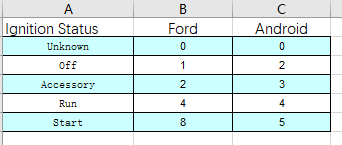
## int getIgnKeyType\_D\_Actl() 获取车辆的mykey的使用信号

## *int getMyKey\_e911Override\_St()*获取车辆的mykey的override状态信号

## int getDrStatDrv\_B\_Actl()获取车辆开门信号DrStatDrv\_B\_Actl

## int getIgnition\_Status()获取车辆点火信号Ignition\_Status

点火状态给到java app，走的是Android原生的carsensormanager，所以需要将福特的点火状态与Android carsensormanager定义的点火状态一一映射，映射关系如下：



## int getPOWER\_MODE()获取车辆电源状态POWER\_MODE的信号

## int getVEDS()获取车辆发生crash时的车辆的信息信号

## interface IRspCallback {

void changed(EmergencyAssistance\_Rsp assistance\_rsp);

void onSensorChange(int eventType, int[] intValues, float[] floatValues);

void onDidRspChange(DIDRspBean didRspBean);

} can信号的变化和点火信号的变化

## void registerPowerStateListener(ISvDataBusDispatcher.Stub listener)监听电源模式的回调

## void unregisterPowerStateListener(ISvDataBusDispatcher.Stub listener取消电源模式的监听回调

## int getPsngrFrntDetct\_D\_Actl()副驾驶是否有人

## DIDRspBean getDidRspBean()弹窗对应的类

# com.ford.vendor.FordBtManager

## void initialize(Context context)启动蓝牙服务（这个必须要先初始化，不然下面的所有方法都没用）

## String getBTUserName()获取已连接用户名称 连接的蓝牙名称

## int getBluetoothConnectStatus()获取蓝牙连接状态

连接是1 断开是0； 这里是判断连接状态 回调见15.10

## int getMissCallsCount(String address)获取未接电话数目

## void connect(String address)连接蓝牙

address 蓝牙地址

## void dial(String address, String number)拨打电话

address 蓝牙地址 number 电话号码

## void terminateCall(String address)挂断电话

Address蓝牙地址

## void connectAudio(String address) 连接语音/免提

Address 蓝牙地址

## void disconnectAudio(String address) 断开语音/挂断

Address 蓝牙地址

## List<SVDevice> getHfpConnectedDevicesList()获取连接设备信息

## interface IBtPbapLoadContactsListener {

**void onPbapSyncStateChanged(String address, int syncType, int syncState);从手机下；**

**syncState=0，同步联系人开始 2.完成  
 void onCotactDownloadProgress(SVContact contact, int currentCount, int totalSize);  
contact;联系人**

**currentCount;进度；**

**totalSize：总进度  
 void onCallLogDownloadProgress(SVCallLog callLog, int currentCount, int totalSize);  
callLog：calllog**

**urrentCount;进度；**

**totalSize：总进度**

**void onLoadPbapFromDBStatusChange(String address, int loadType, int LoadState);  
从数据库取：**

**LoadState：0开始 2完成；  
 void onLoadContactsFromDB(String address, List<SVContact> list, int currentGroup, int groupSize);  
list；联系人列表；**

**urrentCount;进度；**

**totalSize：总进度**

**void onLoadCallLogsFromDB(String address, List<SVCallLog> list, int currentGroup, int groupSize);**

**list：calllog列表**

**currentCount;进度；**

**totalSize：总进度**

**}**

## interface IBtHfpChangeListener {

void onHfpDeviceConnectedStatusChange(String address, int state);  
 蓝牙连接状态的回调

Address： 蓝牙地址

State： 连接状态 0:未连接 1：已连接 2：连接中 10：连接失败

void onPhoneCallAudioStatusChanged(String address, int status);

语音连接状态的回调

Status 连接状态  
 void onPhoneCallStateChange(SVCallInfo callInfo);

电话连接变化的回调

SVCallInfo 电话相关信息 根据mstate判断下面的8种通话状态；  
}

通话状态：

**static class** CallState {  
 **public static final int *CALL\_STATE\_ACTIVE*** = 0;  
 **public static final int *CALL\_STATE\_HELD*** = 1;  
 **public static final int *CALL\_STATE\_DIALING*** = 2;  
 **public static final int *CALL\_STATE\_ALERTING*** = 3;  
 **public static final int *CALL\_STATE\_INCOMING*** = 4;  
 **public static final int *CALL\_STATE\_WAITING*** = 5;  
 **public static final int *CALL\_STATE\_HELD\_BY\_RESPONSE\_AND\_HOLD*** = 6;  
 **public static final int *CALL\_STATE\_TERMINATED*** = 7;  
 **public static final int *CALL\_STATE\_MISSCALL*** = 8;  
}

## interface IBtMasterDeviceChangeListener {

void onMasterDeviceChange(SVDevice device);

根据设备获取蓝牙地址

}

## void unBindServive()释放资源

## String getMasterDeviceAddress()获取蓝牙地址

## List<com.desay\_svautomotive.commonlib.SVCallInfo> getCallInfoList(String address)通话状态集合

## 获取蓝牙联系人

void loadContactPerson(String address,int type)//从手机下

void loadDBContactPerson(String address,int type)//从本地数据库下

获取联系人时先从本地数据库拿，本地数据库没有从手机下；

参数；

Address :蓝牙地址

Type; 类型 ; -1：无效 0：联系人 1：CALLLOG 2：all(1,2)

## SVDevice getMasterDevice()获取主设备的信息

## boolean getBluetoothEarphoneConnectStatus()蓝牙耳机是否连接接口

## interface IBluetoothEarphoneConnectStatusChange {

**void onBluetoothEarphoneConnectStatus(boolean connect);}**

**蓝牙耳机是否连接得回调接口；**

## interface IBluetoothConnectStatusChange {

**void onBluetoothConnectStatusChange(int state);}**

**主设备得连接状态 0断开 1连接**

## boolean playBtMusic() 蓝牙音乐播放

## boolean pauseBtMusic() 蓝牙音乐暂停

## interface IBtSwitchChangeListener{

**void onBtSwitchChange(int status);} 蓝牙开关的回调 1是打开 0是关闭**

## boolean getIsOpen() 蓝牙开关是否打开

## setBluetoothSwitch(boolean isOpen)设置蓝牙的开关 true 开 false 关

## void setMasterDevice(SVDevice device)//设置主设备

## interface IPhoneCallStateChangeListener {//通话设备的集合

**void onPhoneCallStateChange(com.desay\_svautomotive.commonlib.SVCallInfo callInfo);}**

## boolean isPbapSyncing(String address)是的同步联系人

## void syncPbapFromPhone(String address, int type)//同步联系人

type: public static final int TYPE\_SYNC\_CONTACT\_FORM\_PHONE = 0;

## void disconnect(String address)//断开蓝牙

# com.ford.audio. FordAudioManager

## public static final String CAR\_AUDIO\_TYPE\_EA=

SvCarAudioManager.CAR\_AUDIO\_TYPE\_EA;

public static final String CAR\_AUDIO\_TYPE\_EA\_PHONE = SvCarAudioManager.CAR\_AUDIO\_TYPE\_EA\_PHONE;

public static final String CAR\_AUDIO\_TYPE\_EA\_PROMPT =SvCarAudioManager.CAR\_AUDIO\_TYPE\_EA\_PROMPT;

对EA audio接口有以下几点说明：

1、 申请/释放焦点采用原生的接口，不过还需要适配下car\_audio\_type

2、 在播放合成音时，new AudioTrack 必须将申请焦点时定义的AudioAttributes传进来。

3、 开始播放合成音，调用AudioTrack play，当合成音播放完时要调用AudioTrack的stop方法。

## List<android.media.AudioFocusInfo> getAudioFocusInfoForDisplay(int type,int displayId)

获取副驾的source接口

# com.ford.audio.FordCommonManager

## void connect(ConnectionCallback connection) 启动服务，并同步将消息发送出去

## void disConnect()断开服务

## sendReqWithRsp(BaseReqMessage baseReqMessage)

发送带返回消息的接口 BaseReqMessage 救援相关的参数

## sendRequestNoRsp(BaseReqMessage baseReqMessage) 发送不带返回消息的接口 BaseReqMessage 救援相关的参数

## void setResponseListener(IResponseCallback responseListener)设置数据回调

## interface IResponseCallback {

void changed(int commandid, BaseResponse baseResponse);

} can信号得返回接口

# com.ford.vendor.FordHardKeyManager

## void registerKeyEvent(int AppID, IkeyEventListener listener); 根据相应的AppID注册物理按键

AppID: //自定义按键模块AppID

//本地收音机

public static final int APP\_ID\_RADIO = DesayHardKeyManager.APP\_ID\_RADIO;

//蓝牙音乐

public static final int APP\_ID\_BT\_MUSIC = DesayHardKeyManager.APP\_ID\_BT\_MUSIC;

//随心听

public static final int APP\_ID\_BAIDU\_RADIO = DesayHardKeyManager.APP\_ID\_BAIDU\_RADIO;

//蓝牙电话

public static final int APP\_ID\_BT\_PHONE = DesayHardKeyManager.APP\_ID\_BT\_PHONE;

//语音

public static final int APP\_ID\_VR = DesayHardKeyManager.APP\_ID\_VR;

//设置

public static final int APP\_ID\_SETTINGS = DesayHardKeyManager.APP\_ID\_SETTINGS;

//电源管理

public static final int APP\_ID\_POWER\_MANAGEMENT = DesayHardKeyManager.APP\_ID\_POWER\_MANAGEMENT;

//导航

public static final int APP\_ID\_NAVIGATION = DesayHardKeyManager.APP\_ID\_NAVIGATION;

//rvc

public static final int APP\_ID\_RVC = DesayHardKeyManager.APP\_ID\_RVC;

//EA

public static final int APP\_ID\_EA = DesayHardKeyManager.APP\_ID\_EA;

listener见19.3

## void unregisterKeyEvent(int AppID, IkeyEventListener listener)

**取消物理按键的注册 Appid见19.1 listener见19.3**

## interface IkeyEventListener

**{**

**void onCustomKeyEvent(int keyCode, int keyAction, String value);**

**}**

**监听物理按键的回调  
 相关参数如下：**

//自定义KeyCode

//按键定义如下，对应MRD（Ford\_MRD\_Switch\_Input\_Matrix\_V08.xlsx）中的label列：

/\* SWCs (CAN)方向盘按键 \*/

public static final int KEYCODE\_SWC\_SEEK\_MINUS = DesayHardKeyManager.KEYCODE\_SWC\_SEEK\_MINUS;//Seek-

public static final int KEYCODE\_SWC\_SEEK\_PLUS = DesayHardKeyManager.KEYCODE\_SWC\_SEEK\_PLUS;//Seek +

public static final int KEYCODE\_SWC\_SEEK\_LEFT\_PHONE = DesayHardKeyManager.KEYCODE\_SWC\_SEEK\_LEFT\_PHONE;//Seek Left + Phone "SEND"

public static final int KEYCODE\_SWC\_SEEK\_RIGHT\_PHONE = DesayHardKeyManager.KEYCODE\_SWC\_SEEK\_RIGHT\_PHONE;//Seek Right + Phone "End"

public static final int KEYCODE\_SWC\_PUSH\_TO\_TALK = DesayHardKeyManager.KEYCODE\_SWC\_PUSH\_TO\_TALK;//Push to Talk (Voice)(PTT)

public static final int KEYCODE\_SWC\_PHONE\_SEND\_END = DesayHardKeyManager.KEYCODE\_SWC\_PHONE\_SEND\_END;//Phone "Send&End"

public static final int KEYCODE\_SWC\_MENU = DesayHardKeyManager.KEYCODE\_SWC\_MENU;//MENU

public static final int KEYCODE\_SWC\_DISP = DesayHardKeyManager.KEYCODE\_SWC\_DISP;//DISP

public static final int KEYCODE\_SWC\_AUDIO = DesayHardKeyManager.KEYCODE\_SWC\_AUDIO;//Audio

public static final int KEYCODE\_SWC\_SETTING = DesayHardKeyManager.KEYCODE\_SWC\_SETTING;//Setting

public static final int KEYCODE\_SWC\_NAVIGATION = DesayHardKeyManager.KEYCODE\_SWC\_NAVIGATION;//Navigation

/\* Mini ICP (LIN) 中控按键\*/

public static final int KEYCODE\_ICP\_POWER\_VOLUME = DesayHardKeyManager.KEYCODE\_ICP\_POWER\_VOLUME;//Power Volume

public static final int KEYCODE\_ICP\_DAT = DesayHardKeyManager.KEYCODE\_ICP\_DAT;//DAT Shortcut

public static final int KEYCODE\_ICP\_AUDIO\_POWER\_ONOFF = DesayHardKeyManager.KEYCODE\_ICP\_AUDIO\_POWER\_ONOFF;//Audio "Powor On/Off"

public static final int KEYCODE\_ICP\_SEEK\_LEFT = DesayHardKeyManager.KEYCODE\_ICP\_SEEK\_LEFT;//Seek Left

public static final int KEYCODE\_ICP\_SEEK\_RIGHT = DesayHardKeyManager.KEYCODE\_ICP\_SEEK\_RIGHT;//Seek Right

public static final int KEYCODE\_ICP\_PLAY\_PAUSE = DesayHardKeyManager.KEYCODE\_ICP\_PLAY\_PAUSE;//Play / Pause icon

public static final int KEYCODE\_ICP\_REAR\_AUDIO\_LOCK = DesayHardKeyManager.KEYCODE\_ICP\_REAR\_AUDIO\_LOCK;//Rear Audio Lock

public static final int KEYCODE\_ICP\_PARKING\_CAMERA = DesayHardKeyManager.KEYCODE\_ICP\_PARKING\_CAMERA;//Parking+Camera Hot Key

public static final int KEYCODE\_ICP\_SOUND\_MENU = DesayHardKeyManager.KEYCODE\_ICP\_SOUND\_MENU;//Sound Menu

public static final int KEYCODE\_ICP\_SOURCE\_ALL\_MEDIA = DesayHardKeyManager.KEYCODE\_ICP\_SOURCE\_ALL\_MEDIA;//Source (All MEDIA)

public static final int KEYCODE\_ICP\_DISPLAY = DesayHardKeyManager.KEYCODE\_ICP\_DISPLAY;//Display

/\* RCCM CCH/EFP \*/

// 暂时无定义

/\* RACM (CAN)后排按键 \*/

public static final int KEYCODE\_RACM\_SOURCE = DesayHardKeyManager.KEYCODE\_RACM\_SOURCE;//Source

public static final int KEYCODE\_RACM\_SEEK\_FORWARD = DesayHardKeyManager.KEYCODE\_RACM\_SEEK\_FORWARD;//Seek Forward

public static final int KEYCODE\_RACM\_SEEK\_BACK = DesayHardKeyManager.KEYCODE\_RACM\_SEEK\_BACK;//Seek Back

//自定义Key的KeyAction

public static final int KEYACTION\_UP = DesayHardKeyManager.KEYACTION\_UP; //短键的up事件

public static final int KEYACTION\_DOWN = DesayHardKeyManager.KEYACTION\_DOWN; //短按的down事件

public static final int KEYACTION\_LONG\_UP = DesayHardKeyManager.KEYACTION\_LONG\_UP; //长按的up事件

public static final int KEYACTION\_LONG\_DOWN = DesayHardKeyManager.KEYACTION\_LONG\_DOWN; //长按的down事件

public static final int KEYACTION\_ERROR = DesayHardKeyManager.KEYACTION\_ERROR; //错误事件

# com.ford.vendor.FordElectronicHorizonManager

## void onConnect(FordBaseManager.ConnectionCallback connection)

连接服务，同步数据返回接口，连接后才有数据返回

## void onDisConnect() 断开服务连接

## void sendAdasMapEH1(byte[] bytes) 发送 Adas EH1 数据内容

## void sendAdasMapEH2(byte[] bytes) 发送 Adas EH2 数据内容

# com.ford.vendor. FordThemeManager

## int getTheme() 当前主题得读取 主题int: 0 1 2 目前只有3种主题，对应的如下：2是默认主题；

private List<ThemeBean> beanList = Arrays.*asList*(  
 new ThemeBean(0, "静谧之旅",  
 "com.desay.setting.overlay.golden",  
 "com.desay.setting"),  
 new ThemeBean(0, "静雅天成",  
 "com.desay.setting.overlay.green",  
 "com.desay.setting"),  
 new ThemeBean(0, "默认主题",  
 "com.desay.setting.overlay",  
 "com.desay.setting")  
);

## void setTheme( int Index)主题得设置 具体设置见20.1

## interface IThemeChange{

void onThemeChange(int theme);

} 主题变化的回调

# com.desaysdk.commonlib.manager. BtA2dpManager

## List<SVDevice> getA2dpConnectedDevicesList()

获取副驾蓝牙已连接的设备列表。

# intent.getLongExtra (“start\_time”,defaultvalue)

# 获取开机启动lastsource appservice的时间。

# com.ford.vendor. FordMultiscreenManager

## void startResize() 分屏

## void stopResize()全屏

## void swapDisplay() 交换屏

## int getDriveMode()获取驾驶模式

## Display getDisplay(Context context) 封装的接口如下：传入当前Activity的context

注意：

另外，针对27寸屏中相关的一些场景，会存在如下情况需要app做相应的适配，如下：

1、 单实例task分别显示在主屏或副屏

这种情况，该单实例task不会重新走生命周期，但是会接收到onConfigurationChanged回调，此时app可以在该回调中通过getDisplay的方式获取到准确的displayId

01-01 10:32:55.905 29442 29442 D www : displayId: 0;for:com.example.android.helloactivity2.HelloActivity2@2d6528f

01-01 10:33:07.357 29442 29442 D www : displayId: 3;for:com.example.android.helloactivity2.HelloActivity2@2d6528f

2、主副屏左右task互换

主屏上的应用不会重新走生命周期，可以接收到onConfigurationChanged回调，此时app可以在该回调中通过getDisplay的方式获取到准确的displayId

但是副屏会重新走onStart，onResume的生命周期，无法在onConfigurationChanged回调中获取准确的displayId，但是可以在onResume生命周期中获取准确的displayId

01-01 08:01:45.308 4282 4282 D www : displayId: 3;for:com.demo.launcher.TestActivity@811a6aa-----------交换之后原先主屏上的Activity显示准取的displayId

01-01 08:01:45.471 4871 4871 D www : onStart displayId: 3;for:com.example.android.helloactivity2.HelloActivity2@2d6528f

01-01 08:01:45.491 4871 4871 D www : displayId: 3;for:com.example.android.helloactivity2.HelloActivity2@2d6528f

01-01 08:01:45.529 4871 4871 D www : onResume displayId: 0;for:com.example.android.helloactivity2.HelloActivity2@2d6528f-----交换之后原先副屏上的的Activity要在onResume中才能获取准确的displayId

主副驾应用左右切换task的时候需要应用做对应的适配获取自己当前所属的displayId

## interface IDriveModeChange {

**void onDriveModeChange(int driveMode);**

**}获取驾驶模式的回调的**

## void changeViewDisplay(View view)交换窗口

## int getDisplayForView(View view) 获取当前窗口view所在的displayId

## ActivityManager.RunningTaskInfo getTopRunningTaskByDisplayId(int displayid)获取栈顶task的接口（只有27寸有）

## void setSplitScreenAbility(String pkg, boolean enable)

**pkg：设置禁止分屏的应用包名；不能为null或者空字符串，否则app会crash**

**enable ：true，可以分屏，false，禁止分屏；**

## String whoDisAllowSplitScreen()获取是否有设置禁止分屏的应用包名

**返回值不为null，表示该应用设置了禁止分屏**

**返回null，可以分屏。**

## void setSwapScreenEnableForPackage(String packagename,boolean enable)

packagename：禁止/使能换屏的应用的包名；

enable：true，可以换屏，false，禁止换屏。

## void setDriveMode(int driveMode)

driveMode:0,全屏 1,合作模式 2.分屏；

# com.ford.car.FordCarLanDunAARManager

## int getFrontWindshieldHeatingState()获取前窗加热状态

## interface AARLanDunFrontWindshieldHeatingStateListener {

**void onFrontWindshieldHeatingState(int state);**

**}前窗加热状态的回调**

## int getRearWindshieldHeatingStatus()获取后窗加热状态

## interface AARLanDunRearWindsListener {

## void onRearWindshieldHeatingStatus(boolean state);

}后窗是否加热的回调

## int getAirddistModedr(int type)获取当前的气流模式

接口参数说明：

1.int type取值

WS\_TYPE=0; WS模式，对应信号为Defrost\_Btn\_Stt = 1

FL\_TYPE=1; FL模式，对应信号为Floor\_Btn\_Stt =1

接口返回值：

0；模式关闭

1：模式开启

## interface AARLanDunAirddistModedrListener {

void onAirddistModedrChange(int type, int state); }空调的气流模式回调

接口参数说明：

1.int type为气流模式枚举

WS\_TYPE=0; WS模式，对应信号为Defrost\_Btn\_Stt = 1

FL\_TYPE=1; FL模式，对应信号为Floor\_Btn\_Stt =1

2.int state

state=0；模式关闭

state=1；模式开启

## int getCompressorRunStatus()获取当前， 压缩机运行状态

## interface AARCompressorStatusListener {

void onCompressorRunStatus(int state); }压缩机的状态

## int getUpStatus(获取UP状态

## interface AARLanDunUpStatusListener {

void onUpStatus(int status);

}实时获取Up状态

## void connect(ConnectionCallback connection)连接服务

## void disConnect()解除绑定

# com.ford.car. CarProperty(解释文档属性类)

## 所有的车型

public static final int CAR\_MODEL\_DEFAULT = 0;// Defaul用来做DTC

public static final int CAR\_MODEL\_CD542 = 1;//CD542

public static final int CAR\_MODEL\_CX727 = 2;// CX727

public static final int CAR\_MODEL\_U625 = 3;// U625

public static final int CAR\_MODEL\_P702 = 4;// P702

public static final int CAR\_MODEL\_U725 = 5;// U725

public static final int CAR\_MODEL\_U554 = 6;// U554

public static final int CAR\_MODEL\_CD764 = 7;// CD764

public static final int CAR\_MODEL\_CX706 = 8;// CX706

public static final int CAR\_MODEL\_S650 = 9;// S650

public static final int CAR\_MODEL\_CX483 = 0xA;// CX483

public static final int CAR\_MODEL\_CD542\_ICA = 0xB;// CD542 ICA

public static final int CAR\_MODEL\_CX727\_ICA = 0xC;// 727 ICA

public static final int CAR\_MODEL\_P702\_MCA = 0xD;// P702 MCA

public static final int CAR\_MODEL\_CD764\_ICA = 0xE;// 764 ICA

public static final int CAR\_MODEL\_U725C = 0xF;// U725C

**注：红色对应的属性没有车型对应；需要判断这几个车型可以用变体的车型判断接口；**

## 542H,542L,不是542的;

public static final int CAR\_MODEL\_CD542H = 101;

public static final int CAR\_MODEL\_CD542L = 102;

public static final int CAR\_MODEL\_NOT\_CD542 = 100;

## CX727的颜色

public static final int CAR\_COLOR\_CX727\_WHITE = 0x01;

public static final int CAR\_COLOR\_CX727\_BLACK = 0x30;

public static final int CAR\_COLOR\_CX727\_BLUE = 0x45;

public static final int CAR\_COLOR\_CX727\_RED = 0x63;

## 车辆样式

public static final int CAR\_TYPE\_SEDAN = 0x0;

public static final int CAR\_TYPE\_SUV = 0x1;

public static final int CAR\_TYPE\_HATCHBACK = 0x2;

public static final int CAR\_TYPE\_WAGON = 0x3;

public static final int CAR\_TYPE\_PICKUP = 0x4;

## 油电类型

public static final int POWER\_TYPE\_GAS = 0x0;

public static final int POWER\_TYPE\_PHEV = 0x1;

public static final int POWER\_TYPE\_BEV = 0x2;

public static final int POWER\_TYPE\_HEV = 0x3;

public static final int POWER\_TYPE\_FHEV = 0x4;

## 点火信号

public static final int IGNITION\_STATUS\_UNKNOWN = 0;

public static final int IGNITION\_STATUS\_OFF = 1;

public static final int IGNITION\_STATUS\_ACCESSORY = 2;//熄火

public static final int IGNITION\_STATUS\_RUN = 4;//点火

public static final int IGNITION\_STATUS\_START = 8;

public static final int IGNITION\_STATUS\_INVALID = 0xF;

## 里程单位

0是公里 1是英里

## 车速单位

public static final int SPEED\_UNIT\_MPH = 0;// MPH

public static final int SPEED\_UNIT\_KMH = 1;//Km/h

## 胎压单位

public static final int *TIRE\_PRESSURE\_UNIT\_PSI* = 0;//psi  
public static final int *TIRE\_PRESSURE\_UNIT\_KPA* = 1;//kpa  
public static final int *TIRE\_PRESSURE\_UNIT\_BAR* = 2;//bar

## 挡位

public static final int GEAR\_PARK = 0;

public static final int GEAR\_REVERSE = 1;

public static final int GEAR\_NEUTRAL = 2;

public static final int GEAR\_DRIVE = 3;

public static final int GEAR\_SPORT\_DRIVESPORT = 4;

public static final int GEAR\_LOW = 5;

public static final int GEAR\_FIRST = 6;

public static final int GEAR\_SECOND = 7;

public static final int GEAR\_THIRD = 8;

public static final int GEAR\_FOURTH = 9;

public static final int GEAR\_FIFTH = 0XA;

public static final int GEAR\_SIXTH = 0XB;

public static final int GEAR\_UNDEFINED\_TREAT\_AS\_FAULT = 0xC;

public static final int GEAR\_UNDEFINED\_TREAT\_AS\_\_FAULT1 = 0xD;

public static final int GEAR\_UNKNOWN\_POSITION = 0xE;

public static final int GEAR\_FAULT = 0xF;

## 油量警告状态

public static final int FUEL\_STATUS\_OK = 0x0;

public static final int FUEL\_STATUS\_LOW = 0x1;

public static final int FUEL\_STATUS\_VERYLOW = 0x2;

public static final int FUEL\_STATUS\_DTELEVEL1MYKEY = 0x3;

public static final int FUEL\_STATUS\_DTELEVEL2NONMYKEY = 0x4;

public static final int FUEL\_STATUS\_DTELEVEL3 = 0x5;

public static final int FUEL\_STATUS\_DTELEVEL4 = 0x6;

public static final int FUEL\_STATUS\_DTELEVEL5LOWEST = 0x7;

## 续航警告状态

public static final int ENDURANCE\_STATUS\_NULL = 0x0;//

public static final int ENDURANCE\_STATUS\_THRES20MI\_32KM = 0x1;

public static final int ENDURANCE\_STATUS\_THRES30MI\_48KM = 0x2;

public static final int ENDURANCE\_STATUS\_THRES50MI\_80KM = 0x3;

public static final int ENDURANCE\_STATUS\_THRES30KM\_18MI = 0x4;

public static final int ENDURANCE\_STATUS\_THRES50KM\_31MI = 0x5;

public static final int ENDURANCE\_STATUS\_THRES80KM\_50MI = 0x6;

public static final int ENDURANCE\_STATUS\_NOTUSED = 0x7;

## 车窗打开状态

public static final int WINDOW\_STATUS\_UNDEFINED = 0x0;

public static final int WINDOW\_STATUS\_FULLY\_CLOSED = 0x1;

public static final int WINDOW\_STATUS\_BETFULLY\_10PERCENTOPEN = 0x2;

public static final int WINDOW\_STATUS\_BET10PERCENT\_60PERCENT = 0x3;

public static final int WINDOW\_STATUS\_BET60PERCENT\_FULLYOPEN = 0x4;

public static final int WINDOW\_STATUS\_FULLY\_OPEN = 0x5;

public static final int WINDOW\_STATUS\_UNUSED1 = 0x6;

public static final int WINDOW\_STATUS\_UNUSED2 = 0x7;

## 车门开关状态

public static final int DOOR\_STATUS\_CLOSED = 0x0;

public static final int DOOR\_STATUS\_AJAR = 0x1;

## 驾驶模式

public static final int DRIVE\_MODE\_SELDRVMDE01 = 0x0;

public static final int DRIVE\_MODE\_SELDRVMDE02 = 0x1;

public static final int DRIVE\_MODE\_SELDRVMDE03 = 0x2;

public static final int DRIVE\_MODE\_SELDRVMDE04 = 0x3;

public static final int DRIVE\_MODE\_SELDRVMDE05 = 0x4;

public static final int DRIVE\_MODE\_SELDRVMDE06 = 0x5;

public static final int DRIVE\_MODE\_SELDRVMDE07 = 0x6;

public static final int DRIVE\_MODE\_SELDRVMDE08 = 0x7;

public static final int DRIVE\_MODE\_SELDRVMDE09 = 0x8;

public static final int DRIVE\_MODE\_SELDRVMDE10 = 0x9;

public static final int DRIVE\_MODE\_SELDRVMDE11 = 0xA;

public static final int DRIVE\_MODE\_SELDRVMDE12 = 0xB;

public static final int DRIVE\_MODE\_SELDRVMDE13 = 0xC;

public static final int DRIVE\_MODE\_SELDRVMDE14 = 0xD;

public static final int DRIVE\_MODE\_SELDRVMDE15 = 0xE;

public static final int DRIVE\_MODE\_SELDRVMDE16 = 0xF;

public static final int DRIVE\_MODE\_SELDRVMDE17 = 0x10;

public static final int DRIVE\_MODE\_SELDRVMDE18 = 0x11;

public static final int DRIVE\_MODE\_SELDRVMDE19 = 0x12;

public static final int DRIVE\_MODE\_SELDRVMDE20 = 0x13;

public static final int DRIVE\_MODE\_SELDRVMDE21 = 0x14;

public static final int DRIVE\_MODE\_SELDRVMDE22 = 0x15;

public static final int DRIVE\_MODE\_SELDRVMDE23 = 0x16;

public static final int DRIVE\_MODE\_SELDRVMDE24 = 0x17;

public static final int DRIVE\_MODE\_SELDRVMDE25 = 0x18;

public static final int DRIVE\_MODE\_SELDRVMDE26 = 0x19;

public static final int DRIVE\_MODE\_SELDRVMDE27 = 0x1A;

public static final int DRIVE\_MODE\_SELDRVMDE28 = 0x1B;

public static final int DRIVE\_MODE\_SELDRVMDE29 = 0x1C;

public static final int DRIVE\_MODE\_SELDRVMDE30 = 0x1D;

public static final int DRIVE\_MODE\_SELDRVMDE31 = 0x1E;

public static final int DRIVE\_MODE\_FAULTY = 0x1F;

## CX727具体的颜色

public static final int CAR\_COLOR\_CX727\_RAPID\_RED\_TC\_RED = 0X63;//快速红色TC 红色

public static final int CAR\_COLOR\_CX727\_GRABBER\_BLUE\_METALLIC\_BLUE = 0X45;//抓手蓝色金属 蓝色

public static final int CAR\_COLOR\_CX727\_ABSOLUTE\_BLACK = 0X30;//绝对黑 黑色

public static final int CAR\_COLOR\_CX727\_PLATINUM\_WHITE\_3C\_WHITE = 0X01;//白金白3C 白色

## CD542的颜色

public static final int CAR\_COLOR\_CD542\_RAPID\_RED\_TC\_RED = 0X63;//快速红色TC 红色

public static final int CAR\_COLOR\_CD542\_BLUE\_METALLIC\_BLUE = 0X44;//蓝色金属 蓝色

public static final int CAR\_COLOR\_CD542\_DESERT\_ISLAND\_BLUE = 0X42;//荒岛蓝 蓝色

public static final int CAR\_COLOR\_CD542\_SHADOW\_BLACK = 0X30;//暗黑 黑色

public static final int CAR\_COLOR\_CD542\_LUSTROUS\_GREY = 0X25;//光泽灰色 灰色

public static final int CAR\_COLOR\_CD542\_WHITE\_PLATINUM\_3C\_WHITE = 0X01;//白金3C 白色

public static final int CAR\_COLOR\_CD542\_CRYSTAL\_SOLID\_WHITE = 0X00;//水晶纯白 白色

## U554的颜色

public static final int CAR\_COLOR\_U554\_BURGUNDY\_VELVET\_TC\_RED = 0X70;//勃艮第天鹅绒TC 红色

public static final int CAR\_COLOR\_U554\_CHROMA\_FLAME\_RED = 0X64;//色度火焰 红色

public static final int CAR\_COLOR\_U554\_OCEAN\_DRIVE\_BLUE\_TC\_BLUE = 0X48;//Ocean Drive蓝色TC 蓝色

public static final int CAR\_COLOR\_U554\_FLIGHT\_BLUE = 0X47;//飞行蓝 蓝色

public static final int CAR\_COLOR\_U554\_CHROMA\_CRYSTAL\_BLUE = 0X46;//色度晶蓝 蓝色

public static final int CAR\_COLOR\_U554\_INFINITE\_BLACK = 0X31;//无限黑 黑色

public static final int CAR\_COLOR\_U554\_STARLIGHT\_GRAY\_PC\_GRAY = 0X28;//星光灰PC 灰色

public static final int CAR\_COLOR\_U554\_ASHER\_GRAY = 0X27;//灰灰 灰色

public static final int CAR\_COLOR\_U554\_SILVER\_RADIANCE\_GRAY = 0X26;//银光辉 灰色

public static final int CAR\_COLOR\_U554\_CERAMIC\_PEARL\_3C\_BEIGE = 0X23;//陶瓷珍珠3C 米色

public static final int CAR\_COLOR\_U554\_MANHATTAN\_GREEN\_BROWN = 0X16;//曼哈顿格林 棕色

public static final int CAR\_COLOR\_U554\_PRISTINE\_WHITE\_3C\_WHITE = 0X03;//原始白3C 白色

## P702的颜色

public static final int CAR\_COLOR\_P702\_RAPID\_RED\_TC\_X = 0X63;//快速红色TC（也称为清醒红色TC）

public static final int CAR\_COLOR\_P702\_CODE\_ORANGE\_X = 0X53;//橙色代码

public static final int CAR\_COLOR\_P702\_ANTIMATTER\_BLUE\_X = 0X4A;//反物质蓝

public static final int CAR\_COLOR\_P702\_VELOCITY\_BLUE\_X = 0X49;//速度蓝

public static final int CAR\_COLOR\_P702\_AGATE\_BLACK\_X = 0X31;//玛瑙黑

public static final int CAR\_COLOR\_P702\_LEAD\_FOOT\_X = 0X29;//铅脚

public static final int CAR\_COLOR\_P702\_ICONIC\_SILVER\_X = 0X26;//标志性银

public static final int CAR\_COLOR\_P702\_OXFORD\_WHITE\_X = 0X05;//牛津白

## U625的颜色

public static final int CAR\_COLOR\_U625\_BURGUNDY\_VELVET\_TC\_RED = 0X70;//勃艮第天鹅绒TC 红色

public static final int CAR\_COLOR\_U625\_BLUE\_PANTHER\_BLUE = 0X41;//蓝豹 蓝色

public static final int CAR\_COLOR\_U625\_FORD\_PERFORMANCE\_BLUE = 0X40;//福特性能蓝 蓝色

public static final int CAR\_COLOR\_U625\_ABSOLUTE\_BLACK = 0X30;//绝对黑 黑色

public static final int CAR\_COLOR\_U625\_WHITE\_PLATINUM\_WHITE = 0X01;//白金 白色

## U611的颜色

public static final int CAR\_COLOR\_U611\_BLUE\_DIAMOND\_BC\_CC\_BLUE = 0X44;//蓝钻BC / CCNA-除BL外的所有中国总统独特 蓝色

public static final int CAR\_COLOR\_U611\_ICED\_MOCHA\_BROWN = 0X12;//冰摩卡 棕色

public static final int CAR\_COLOR\_U611\_PRISTINE\_WHITE\_3C\_WHITE = 0X03;//原始白3C 白色

## D568MCA的颜色

public static final int CAR\_COLOR\_D568MCA\_BURGUNDY\_VELVET\_TC\_RED = 0X70;//勃艮第天鹅绒TC 红色

public static final int CAR\_COLOR\_D568MCA\_MAGMA\_RED = 0X62;//岩浆红 红色

public static final int CAR\_COLOR\_D568MCA\_BLUE\_PANTHER\_BLUE = 0X41;//蓝豹 蓝色

public static final int CAR\_COLOR\_D568MCA\_SHADOW\_BLACK = 0X30;//暗黑 黑色

public static final int CAR\_COLOR\_D568MCA\_STONE\_GRAY = 0X22;//石灰色 灰色

public static final int CAR\_COLOR\_D568MCA\_MICRO\_SILVER\_GREY = 0X20;//微银灰色 灰色

public static final int CAR\_COLOR\_D568MCA\_WHITE\_PLATINUM\_WHITE = 0X01;//白金 白色

public static final int CAR\_COLOR\_D568MCA\_CRYSTAL\_SOLID\_WHITE = 0X00;//水晶纯白 白色

## CX483的颜色

public static final int CAR\_COLOR\_CX483\_BURGUNDY\_VELVET\_TC\_RED = 0X70;//勃艮第天鹅绒TC 红色

public static final int CAR\_COLOR\_CX483\_ARTISAN\_BLUE = 0X43;//工匠蓝 蓝色

public static final int CAR\_COLOR\_CX483\_INFINITE\_BLACK = 0X31;//无限黑 黑色

public static final int CAR\_COLOR\_CX483\_INGOT\_SILVER\_METALLIC\_GREY = 0X24;//金属锭银 灰色

public static final int CAR\_COLOR\_CX483\_CERAMIC\_PEARL\_3C\_GREY = 0X23;//陶瓷珍珠3C 灰色

public static final int CAR\_COLOR\_CX483\_ICED\_MOCHA\_BROWN = 0X12;//冰摩卡 棕色

public static final int CAR\_COLOR\_CX483\_OCHRE\_BROWN\_METALLIC\_BROWN = 0X11;//茶黄棕色金属 棕色

public static final int CAR\_COLOR\_CX483\_PRISTINE\_WHITE\_3C\_WHITE = 0X03;//原始白3C 白色

## CX482的颜色

public static final int CAR\_COLOR\_CX482\_RACE\_RED = 0X60;//种族红 红色

public static final int CAR\_COLOR\_CX482\_BLUE\_METALLIC\_BLUE = 0X44;//蓝色金属 蓝色

public static final int CAR\_COLOR\_CX482\_ABSOLUTE\_BLACK = 0X30;//绝对黑 黑色

public static final int CAR\_COLOR\_CX482\_RICH\_COPPER\_TC\_BROWN = 0X15;//富铜TC 棕色

public static final int CAR\_COLOR\_CX482\_DESERT\_GOLD\_BROWN = 0X14;//沙漠金 棕色

public static final int CAR\_COLOR\_CX482\_CARIBOU\_BROWN = 0X13;//驯鹿 棕色

public static final int CAR\_COLOR\_CX482\_OXFORD\_WHITE\_SOLIC\_C\_C\_WHITE = 0x05;//牛津白索尔C / C 白色

public static final int CAR\_COLOR\_CX482\_CRYSTAL\_PEARL\_WHITE = 0X04;//水晶珍珠白 白色

public static final int CAR\_COLOR\_CX482\_CRYSTAL\_SOLID\_WHITE = 0X00;//水晶纯白 白色

## CD539MCA的颜色

public static final int CAR\_COLOR\_CD539MCA\_BURGUNDY\_VELVET\_TC\_RED=0X70;//勃艮第天鹅绒TC 红色

public static final int CAR\_COLOR\_CD539MCA\_MAGMA\_RED=0X62;//岩浆红 红色

public static final int CAR\_COLOR\_CD539MCA\_CANYON\_RIDGE\_ORANGE=0X52;//峡谷岭 橙色😍

public static final int CAR\_COLOR\_CD539MCA\_BLUE\_PANTHER\_BLUE=0X41;//蓝豹 蓝色

public static final int CAR\_COLOR\_CD539MCA\_FORD\_PERFORMANCE\_BLUE=0X40;//福特性能蓝 蓝色

public static final int CAR\_COLOR\_CD539MCA\_SHADOW\_BLACK=0X30;//暗黑 黑色

public static final int CAR\_COLOR\_CD539MCA\_STONE\_GRAY=0X22;//石灰色 灰色

public static final int CAR\_COLOR\_CD539MCA\_MICRO\_SILVER\_GRAY=0X20;//微银灰色 灰色

public static final int CAR\_COLOR\_CD539MCA\_WHITE\_PLATINUM\_TRI\_COAT\_WHITE=0X01;//白色铂金三层外套 白色

public static final int CAR\_COLOR\_CD539MCA\_CRYSTAL\_SOLID\_WHITE=0X00;//水晶纯白 白色

## C519的颜色

public static final int CAR\_COLOR\_C519\_RED\_CANDY\_TC\_RED=0X61;//红色糖果Tc 红色

public static final int CAR\_COLOR\_C519\_RACE\_RED=0X60;//种族红 红色

public static final int CAR\_COLOR\_C519\_DARK\_MULBERRY\_4C\_ORANGE=0X51;//黑桑4c 橙色

public static final int CAR\_COLOR\_C519\_ORANGE\_FURY\_3C\_ORANGE=0X50;//橙怒3c 橙色

public static final int CAR\_COLOR\_C519\_DESERT\_ISLAND\_BLUE=0X42;//荒岛蓝 蓝色

public static final int CAR\_COLOR\_C519\_FORD\_PERFORMANCE\_BLUE=0X40;//福特性能蓝 蓝色

public static final int CAR\_COLOR\_C519\_ABSOLUTE\_BLACK=0X30;//绝对黑 黑色

public static final int CAR\_COLOR\_C519\_DIFFUSED\_SILVER\_GREY=0X21;//扩散银 灰色

public static final int CAR\_COLOR\_C519\_CANYON\_RIDGE\_BROWN=0X10;//峡谷岭 棕色

public static final int CAR\_COLOR\_C519\_CRYSTAL\_PEARL\_WHITE\_3C\_WHITE=0X02;//水晶珍珠白3c 白色

public static final int CAR\_COLOR\_C519\_CRYSTAL\_SOLID\_WHITE=0X00;//水晶纯白 白色

## 电源模式

public static final int POWER\_STATE\_NORMAL\_EXT\_PLAY = DesayCommonManager.STATE\_NORMAL\_EXT\_PLAY;//扩展模式SS\_EXT\_PLAY

public static final int POWER\_STATE\_NORMAL\_PHONE = DesayCommonManager.STATE\_NORMAL\_PHONE;//电话模式SS\_PHONE

public static final int POWER\_STATE\_NORMAL\_WORK = DesayCommonManager.STATE\_NORMAL\_WORK;//正常模式SS\_MM\_ACTIVE

public static final int POWER\_STATE\_ABNORMAL = DesayCommonManager.STATE\_ABNORMAL;//Load Shed模式

public static final int POWER\_STATE\_OTA = DesayCommonManager.STATE\_OTA;//升级模式SS\_SWDL

public static final int POWER\_STATE\_STANDBY = DesayCommonManager.STATE\_STANDBY;//待机模式SS\_MM\_INACTIVE

public static final int POWER\_STATE\_SLEEP = DesayCommonManager.STATE\_SLEEP;//休眠模式SS\_OFF

public static final int POWER\_STATE\_TRANSPORT = DesayCommonManager.STATE\_TRANSPORT;//运输模式

public static final int POWER\_STATE\_FACTORY = DesayCommonManager.STATE\_FACTORY;//工厂模式

# com.ford.vendor. FordIpptClientManager

## void connect()//连接服务

## void onDisConnect()//断开服务

## void cleanup()

## int registerClient()

## int unregisterClient()

## int sendTokenRequest(String transactionId, boolean policyValidation, List<MIpptKeyValueBean> configurations, List<String> scopes)

## int sendRevocationResponse(String revocationId)

## int sendRevocationResult(String revocationId, int result, String errorDescription)

## interface IIpptListener {

**void registerCallback(int i);**

**void cloudConnectivityStatusCallback(boolean b);**

**void serverStatusCallback(int i);**

**void tokenRequestStatusCallback(String s, int i, String s1);**

**void tokenResponseCallback(String s, TokenResponseData tokenResponseData);**

**void tokenResponseErrorCallback(String s, int i);**

**void tokenRevocationCallback(String s);**

**}**

**}**

# com.ford.vendor. FordFaceIdManager

## void connect(ConnectionCallback connection) 连接服务

## void disConnect() 断开服务

## void sendRegister() 激活

## void sendRecognize() 人脸识别

## void sendUnbind() 解绑

## void sendUnbind(int faceId) 解绑

## int getFunctionResult() 获取功能执行结果

## interface IFunctionResult {

**void onResult(int resultCode);}**获取功能执行结果的回调

## int getQueryDsmcState() 获取 DSMC 功能的状态

## interface IDsmcStateChangeListener {

**void onResult(int code);}**获取 DSMC 功能的状态的回调

## int getDetectFaceResult() 获取检测人脸检测时状态结果

## interface IDetectFaceResult {

**void onResult(int code);}**获取检测人脸检测时状态结果的回调

## int getRecognizeFaceResult() 获取检测人脸识别时状态结果

## interface IRecognizeFaceResult {

**void onResult(int code);}**取检测人脸识别时状态结果的回调

## long getFaceId() 获取识别成功的 faceId

## interface IFaceIdResult {

**void onResult(long faceId);}**获取识别成功的 faceId 的回调

## int getChecksumFaceId() 获取 DSMC faceId 值

## interface IChecksumFaceIdResult {

**void onResult(int faceId);}**获取 DSMC faceId 值 的回调

## int getCameraState() 获取 camera 状态

## interface ICameraState {

**void onResult(int code);}**获取 camera 状态的回调

## int getCameraDrvIRState() 获取 camera 状态

## interface ICameraIRState {

**void onResult(int code);}**获取 camera 状态的回调

## int getCameraPassIRState() 获取 camera 状态

## interface ICameraPassIRState {

**void onResult(int code);}**获取 camera 状态的回调

## void keepAwake()

## void idle()

## void clearData()清空数据

# com.ford.vendor. FordConfigManager

## byte getTSR() DE04 B6b7

## byte getPowerToTheBox() DE06 B3b7

## byte getARNavigation() DE04 B5b7

## byte getLVDSInCluster() DE07 B4b0

## byte getFacialRecognition() DE06 B5b7

## byte getCRMDSMC() DE07 B4b1

## byte getFreshAirCabin() DE03 B3b7

## byte getFuelType() DE04 B3b7

## byte getEVRangeRing() DE04 B6b3

## byte getAdvancedHUD()DE07 B2b2

## byte getSmart\_DSP()内外置功放

2是外置

## byte getEnhanced\_Memory()

|  |  |  |
| --- | --- | --- |
| Disabled=0 | 3 person hard key=1 | 3 person soft key=2 |

个性化

## byte getNavi\_Features()

## byte getADAS()

## byte getAmbientLight()

## byte getSelectableDriveModes()

## byte getRemoteStartSteeringWheel()

## byte getDigital\_Scent()

## byte getTPMS()

## byte getTPMS\_By\_Location()

## byte getSui\_Xin\_Pai()

随心拍

## byte getRelax()

## byte getHeat\_cool\_seat()

## byte getHeated\_SW()

## byte getDSM()

## int getBT\_tuning 蓝牙Tuning

0x0 CD542 5Dr 6 Speaker 0x1 CD542 5Dr 10 Speaker 0x2 CX727 6 Speaker

0x3 CX727 10 Speaker 0x4 U554 20 Speaker 0x5 P702 18 Speaker

0x6 U625 6 Speaker 0x7 U625 12 Speaker 0x8 CD764 9 Speaker

0x9 CD764 13 Speaker 0xA U625 14 Speaker 0xB CD542 4Dr 6 Speaker

0xC CD542 4Dr 10 Speaker 0xD CDX706 6 Speaker 0xE CDX706 12 Speaker

0xF S650 6 Speaker 0x10 S650 12 Speaker 0x11 CX483MCA 9 Speaker

0x12 CX483MCA 13 Speaker

## byte getNavigation\_Prompt\_Protection 导航音量保护配置位

|  |  |
| --- | --- |
| 0：Present | 1：Not Present |

## byte getAPA()

0:Disabled 1:Semi Auto U540 2:EAPA (Semi Auto)

3:FAPA Fully auto 4:FAPA w RePA 5:FAPA w RePA w Deluxe

## byte getLidget()

0:Disabled 1:Variant1 2:Variant2 3:Variant3

4:Variant4 5:Variant5 6:Variant6 7:Variant7

8:Variant8 9:Variant9 10:Variant10 11:Variant11

12:Variant12 13:Variant13 14:Variant14 15:Variant15

## byte getPicture\_gallery

0是Present，1是Not Present.

## byte getFrunkSoftswitch()

## byte getPowerLiftgateControlFunction()

## byte getAEIS\_with\_Override()

## byte getDaytime\_Running\_Lamps\_Control\_Function()

## int getNavigation\_floating\_window()

## byte getProjectionModes()

|  |  |
| --- | --- |
| 0：Disabled | 1：Wireless Carplay |

## int getMode\_Card() 场景卡片

0：Disabled 1：BGC only 2：BGC+UGC 3：Reserved

## int getCamp\_Mode() 露营模式

0:Disabled 1:Enabled

## int getPet\_Mode() 宠物模式

0:Disabled 1:Enabled

## int getBattery\_Save\_Mode() 省电模式

0:Disabled 1:Enabled

## int getRelax\_Mode() 舒享时分

0:Disabled 1:Enabled

## int getGuest\_Mode() 客人模式

0:Disabled 1:Enabled

## int getCommute\_Mode() 上下班模式

0:Disabled 1:Enabled

## int getGlamour\_Photo\_Studio\_Mode() 魅力摄影模式

0：Disabled 1：Enabled

## byte getSound\_Mode\_Stereo\_Surround() 方位选择

0：Postion selection 1：Stereo/Surround 2：Not support

## byte getDSP\_feature\_Quantum\_Logic\_Surrounding() 音效模式

0：Disabled 1：Surround 2：3D Surround

## int getBluetooth\_earphone()

## int getSDM\_And\_Theme\_Linkage()

## int getTrailer\_Maintenance\_and\_Reminders()

## byte getStabilizer\_Bar\_Disconnect()

## byte getOff\_Road\_Screen()

## byte getSteering\_Gear\_Ratio()

## byte get2WD\_4x4\_AWD()

# com.ford.vendor.FordV2ILiteManager

## void onConnect(FordBaseManager.ConnectionCallback connection) *同步接口，等待连接后返回数据*

## void onDisConnect()*断开连接*

## void sendTrafficLight\_Data\_1(byte[] bytes)

## void sendTrafficLight\_Data\_2(byte[] bytes)

# com.ford.vendor.FordMultiContourSeatManager

## void connect(FordBaseManager.ConnectionCallback connection) 连接服务

## void disConnect() 断开服务连接

## byte getMCSeat()座椅按摩的配置位

## void setSeatFnDrv\_D\_RqPara1(int Module)

## void setSeatFnChngDrv2\_D\_RqPara2(int Module)

## void setSeatFnChngDrv2\_D\_RqPara3(int Module)

## void setSeatFnPsgr\_D\_RqPara4(int Module)

## void setSeatFnChngPsgr2\_D\_RqPara5(int Module)

## void setSeatMasgPsngr\_D\_RqPara6(int Module)

## int getSeatFnDrv\_D\_StatPara1() 主驾开关 等于7是开，非7就是关

## 

## int getSeatPDrv\_B\_StatPara2()

## int getSeatBlLoDrv\_Pc\_ActlPara3()

## int getSeatBlUpDrv\_Pc\_ActlPara4()

## int getSeatLmbrLoDrv\_Pc\_ActlPara5()

## int getSeatLmbrMidDrv\_Pc\_ActlPara6()

## int getSeatLmbrUpDrv\_Pc\_ActlPara7()

## int getSeatIntnsDrv\_D\_StatPara8()

## int getSeatMasgDrv\_D\_StatPara9()

## int getSeatSwtchDrv\_B\_StatPara10()

## int getSeatFnPsgr\_D\_StatPara11() 副驾开关，等于7就是开，非7就是关



## int getSeatPPsgr\_B\_StatPara12()

## int getSeatIntnsPsngr\_D\_StatPara13()

## int getSeatMasgPsngr\_D\_StatPara14()

## int getSeatSwtchPsgr\_B\_StatPara15()

## int getSeatBlLoPsgr\_Pc\_ActlPara16()

## int getSeatBlUpPsgr\_Pc\_ActlPara17()

## int getSeatLmbrLoPsgr\_Pc\_ActlPara18()

## int getSeatLmbrMidPsgr\_Pc\_ActlPara19()

## int getSeatLmbrUpPsgr\_Pc\_ActlPara20()

## int getSeatScrnPsngrOn\_B\_RqPara21()

## int getPsgrSeatUpdateFlag()//副驾 1 2在副驾位置；

## int getDriveSeatUpdateFlag()//主驾位置 除了副驾位置就是主驾位置

## void setReleaxSeateState(boolean isShow)是否控制releax true：控制 falas；不控制

## interface IMultiContourSeatListener {

**void onMultiContourSeatChange(int seatParam, int value);}**

**接口定义的值;**

**seatParam:对应21个信号的param;**

**public static final int SEAT\_PARAM\_SEATFNDRV\_D\_STAT = 1;**

**public static final int SEAT\_PARAM\_SEATPDRV\_B\_STAT = 2;**

**public static final int SEAT\_PARAM\_SEATBLLODRV\_PC\_ACTL = 3;**

**public static final int SEAT\_PARAM\_SEATBLUPDRV\_PC\_ACTL = 4;**

**public static final int SEAT\_PARAM\_SEATLMBRLODRV\_PC\_ACTL = 5;**

**public static final int SEAT\_PARAM\_SEATLMBRMIDDRV\_PC\_ACTL = 6;**

**public static final int SEAT\_PARAM\_SEATLMBRUPDRV\_PC\_ACTL = 7;**

**public static final int SEAT\_PARAM\_SEATINTNSDRV\_D\_STAT = 8;**

**public static final int SEAT\_PARAM\_SEATMASGDRV\_D\_STAT = 9;**

**public static final int SEAT\_PARAM\_SEATSWTCHDRV\_B\_STAT = 10;**

**public static final int SEAT\_PARAM\_SEATFNPSGR\_D\_STAT = 11;**

**public static final int SEAT\_PARAM\_SEATPPSGR\_B\_STAT = 12;**

**public static final int SEAT\_PARAM\_SEATINTNSPSNGR\_D\_STAT = 13;**

**public static final int SEAT\_PARAM\_SEATMASGPSNGR\_D\_STAT = 14;**

**public static final int SEAT\_PARAM\_SEATSWTCHPSGR\_B\_STAT = 15;**

**public static final int SEAT\_PARAM\_SEATBLLOPSGR\_PC\_ACTL = 16;**

**public static final int SEAT\_PARAM\_SEATBLUPPSGR\_PC\_ACTL = 17;**

**public static final int SEAT\_PARAM\_SEATLMBRLOPSGR\_PC\_ACTL = 18;**

**public static final int SEAT\_PARAM\_SEATLMBRMIDPSGR\_PC\_ACTL = 19;**

**public static final int SEAT\_PARAM\_SEATLMBRUPPSGR\_PC\_ACTL = 20;**

**public static final int SEAT\_PARAM\_SEATSCRNPSNGRON\_B\_RQ = 21;**

**public static final int *SEAT\_PARAM\_DRIVESEATUPDATEFLAG* = 22;  
public static final int *SEAT\_PARAM\_PSGRSEATUPDATEFLAG* = 23;**

**value:23个信号param对应的值；**

## void setMasterDrivingMassage(int module1,int module2,int module3)设置主驾

## void setPassengerDrivingMassage(int module1,int module2,int module3)设置副驾

# com.ford.vendor.FordFragranceManager

## void setSurroundMode(int state) 设置音效 1.立体音 2.环绕音

public static final int *STEREO* = SurroundMode.*STEREO*;//立体声  
public static final int *SURROUND* = SurroundMode.*SURROUND*;//环绕声

## int getSurroundMode() 获取音效 1.立体音 2.环绕音

public static final int *STEREO* = SurroundMode.*STEREO*;//立体声  
public static final int *SURROUND* = SurroundMode.*SURROUND*;//环绕声

## interface ISurroundMode {

**void onSurroundModeChange(int mode);}音效的回调**

## void setQLSurround(int qLSurround)

## interface IQLSurroundListener{

**void onQLSurroundChange(int QLSurround);}**

public static final int *QL\_SURROUND\_OFF* = QLSurround.*QL\_SURROUND\_OFF*;  
public static final int *QL\_STAGE\_EFFECT* = QLSurround.*QL\_STAGE\_EFFECT*;  
public static final int *QL\_AUDIENCE* = QLSurround.*QL\_AUDIENCE*;

## int getQLSurround()

public static final int *QL\_SURROUND\_OFF* = QLSurround.*QL\_SURROUND\_OFF*;  
public static final int *QL\_STAGE\_EFFECT* = QLSurround.*QL\_STAGE\_EFFECT*;  
public static final int *QL\_AUDIENCE* = QLSurround.*QL\_AUDIENCE*;

## void connect(ConnectionCallback connection)

**设置和获取香氛的接口以及回调需要先连接服务，设置音效相关的接口不需要连接；**

## void disConnect()断开服务

## byte getDigitalScent()电子香氛得配置位

## void setAC\_1\_FGA\_OperationReq(int state)香氛开关设置

|  |  |
| --- | --- |
| 0 | 0x0:初始状态 |
| 1 | 0x1:香氛开始运行 |
| 2 | 0x2:香氛停止工作 |
| 3 | 0x3: 主节点异常 |
| 0xFF | 不请求 |

## void setAC\_1\_GenderSts(int sex)性别状态

|  |  |
| --- | --- |
| 0 | 0x0:未知性别 |
| 1 | 0x1:性别女 |
| 2 | 0x2:性别男 |
| 3 | 0x3:其它 |
| 0xFF | 不请求 |

## void setAC\_1\_FGAChanTypSelect(int channel)香氛通道选择

|  |  |
| --- | --- |
| 0 | 0x0:香氛通道关闭 |
| 1 | 0x1:开启通道1的香氛 |
| 2 | 0x2:开启通道2的香氛 |
| 3 | 0x3:开启通道3的香氛 |
| 0xFF | 不请求 |

## void setAC\_1\_FGAIntensityReq(int density)香氛浓度设置

|  |  |
| --- | --- |
| 0 | 0x0:未知 |
| 1 | 0x1:香氛浓度关闭 |
| 2 | 0x2:预留 |
| 3 | 0x3:开启的通道对应的香氛浓度为低浓度 |
| 4 | 0x4:预留 |
| 5 | 0x5:开启的通道对应的香氛浓度为中浓度 |
| 6 | 0x6:预留 |
| 7 | 0x7:开启的通道对应的香氛浓度为高浓度 |
| 0xFF | 不请求 |

## void setAC\_1\_Time\_Year(int year)年份设置

|  |  |
| --- | --- |
| 0-51 | 最小值=2020 最大值=2051 精度 = 1 偏移量 = 2020 [Year] |
| 0xFF | 不请求 |

## void setAC\_1\_Time\_Day(int day)年份设置

|  |  |
| --- | --- |
| 0-511 | 最小值=0 最大值=511 精度 = 1 偏移量 = 0 [儒略日] |
| 0xFFFF | 不请求 |

## int getFGA\_1\_FG\_Chan1Typ() 通道1香氛类型

|  |  |
| --- | --- |
| 0 | 0:未知的 |
| 1-253 | 1-253:正确的香氛类型ID |
| 254 | 254:未认证的/无效的 |
| 255 | 255:未安装 |

## int getFGA\_1\_FG\_Chan2Typ() 通道2香氛类型

|  |  |
| --- | --- |
| 0 | 0:未知的 |
| 1-253 | 1-253:正确的香氛类型ID |
| 254 | 254:未认证的/无效的 |
| 255 | 255:未安装 |

## int getFGA\_1\_FG\_Chan3Typ() 通道3香氛类型

|  |  |
| --- | --- |
| 0 | 0:未知的 |
| 1-253 | 1-253:正确的香氛类型ID |
| 254 | 254:未认证的/无效的 |
| 255 | 255:未安装 |

## int getFGA\_1\_FG\_LifeRemainingChan1() 通道1香氛罐残余百分比

|  |  |
| --- | --- |
| 0-100 | 0-100:代表通道1香氛剩余使用寿命占比 |
| 101-127 | 101-127:未知/无效 |

## int getFGA\_1\_FG\_LifeRemainingChan2() 通道2香氛罐残余百分比

|  |  |
| --- | --- |
| 0-100 | 0-100:代表通道1香氛剩余使用寿命占比 |
| 101-127 | 101-127:未知/无效 |

## int getFGA\_1\_FG\_LifeRemainingChan3() 通道3香氛罐残余百分比

|  |  |
| --- | --- |
| 0-100 | 0-100:代表通道1香氛剩余使用寿命占比 |
| 101-127 | 101-127:未知/无效 |

## int getFGA\_1\_FGACurrentdensity() 当前香氛工作浓度信息

|  |  |
| --- | --- |
| 0 | 0x0= Unknown |
| 1 | 0x1= Off |
| 2 | 0x2= Reserved |
| 3 | 0x3= Low Intensity |
| 4 | 0x4= Reserved |
| 5 | 0x5= Medium Intensity |
| 6 | 0x6= Reserved |
| 7 | 0x7= High Intensity |

## int getFGA\_1\_FGACurrentWorkCh() 当前香氛模块工作状态

|  |  |
| --- | --- |
| 0 | 0x0:Off |
| 1 | 0x1:Channel 1 Working |
| 2 | 0x2:Channel 2 Working |
| 3 | 0x3:Channel 3 Working |

## int getFGA\_3\_FGID1Overdue() 通道1香氛超期敬告

|  |  |
| --- | --- |
| 0 | 0:通道1香氛已过期 |
| 1~30 | 1-30:通道1香氛即将过期的剩余天数 |
| 31 | 31:通道1未过期（包括香氛盒未安装、未认证） |

## int getFGA\_3\_FGID2Overdue() 通道2香氛超期敬告

|  |  |
| --- | --- |
| 0 | 0:通道2香氛已过期 |
| 1~30 | 1-30:通道2香氛即将过期的剩余天数 |
| 31 | 31:通道2未过期（包括香氛盒未安装、未认证） |

## int getFGA\_3\_FGID3Overdue() 通道3香氛超期敬告

|  |  |
| --- | --- |
| 0 | 0:通道3香氛已过期 |
| 1~30 | 1-30:通道3香氛即将过期的剩余天数 |
| 31 | 31:通道3未过期（包括香氛盒未安装、未认证） |

## int getFGA\_3\_FGAStatusSts() 香氛系统状态

|  |  |
| --- | --- |
| 0 | 0x0:香氛当前状态是唤醒的且没有开启香气 |
| 1 | 0x1:香氛当前状态是唤醒的且有开启香气 |
| 2 | 0x2:模式错误 |
| 3 | 0x3:预留 |

## int getFGA\_3\_FRAGTempSts() 香氛温度状态

|  |  |
| --- | --- |
| 0 | 0x0:温度正常 |
| 1 | 0x1:温度过高 |
| 2 | 0x2:温度过低 |
| 3 | 0x3:无 |

## int getFGA\_3\_FRAGPowerSupplySts() 香氛电压状态

|  |  |
| --- | --- |
| 0 | 0x0:电源正常 |
| 1 | 0x1:电源过压 |
| 2 | 0x2:电源欠压 |
| 3 | 0x3:无 |

## int getFGA\_3\_FRAGFanSts() 香氛风机状态

|  |  |
| --- | --- |
| 0 | 0x0:风机正常 |
| 1 | 0x1:风机堵塞异常 |

## int getFGA\_3\_FRAGUnKnownErr() 香氛未知错误

|  |  |
| --- | --- |
| 0 | 0x0:无其它未知异常 |
| 1 | 0x1:有其它未知异常（如：电机异常） |

## int getFGA\_2\_HW\_Major() 硬件主版本号

|  |  |
| --- | --- |
| 0x0-0x03 | 0x0-0x03:有效主版本号 |

## int getFGA\_2\_HW\_Minor() 硬件小版本号

|  |  |
| --- | --- |
| 0x0-0x0F | 0x0-0x0F:有效小版本号 |

## int getFGA\_2\_SW\_Major() 软件主版本号

|  |  |
| --- | --- |
| 0x0-0x03 | 0x0-0x03:有效主版本号 |

## int getFGA\_2\_SW\_Minor() 软件主版本号

|  |  |
| --- | --- |
| 0x0-0x0F | 0x0-0x0F:有效小版本号 |

## interface IFragranceChangeListener {

**void onFragranceChange(int type, int value);}香氛的状态变化的回调；**

**value为各type对应的值；**

**type如下：**

**通道1香氛类型**

**public static final int TYPE\_FGA\_1\_FG\_CHAN1TYP = 0;**

**通道2香氛类型**

**public static final int TYPE\_FGA\_1\_FG\_CHAN2TYP = 1;**

**通道3香氛类型**

**public static final int TYPE\_FGA\_1\_FG\_CHAN3TYP = 2;**

**通道1香氛罐残余百分比**

**public static final int TYPE\_FGA\_1\_FG\_LIFEREMAININGCHAN1 = 3;**

**通道2香氛罐残余百分比**

**public static final int TYPE\_FGA\_1\_FG\_LIFEREMAININGCHAN2 = 4;**

**通道3香氛罐残余百分比**

**public static final int TYPE\_FGA\_1\_FG\_LIFEREMAININGCHAN3 = 5;**

**当前香氛工作浓度信息**

**public static final int TYPE\_FGA\_1\_FGACURRENTDENSITY = 6;**

**当前香氛模块工作状态**

**public static final int TYPE\_FGA\_1\_FGACURRENTWORKCH = 7;**

**通道1香氛超期敬告**

**public static final int TYPE\_FGA\_3\_FGID1OVERDUE = 8;**

**通道2香氛超期敬告**

**public static final int TYPE\_FGA\_3\_FGID2OVERDUE = 9;**

**通道3香氛超期敬告**

**public static final int TYPE\_FGA\_3\_FGID3OVERDUE = 10;**

**香氛系统状态**

**public static final int TYPE\_FGA\_3\_FGASTATUSSTS = 11;**

**香氛温度状态**

**public static final int TYPE\_FGA\_3\_FRAGTEMPSTS = 12;**

**香氛电压状态**

**public static final int TYPE\_FGA\_3\_FRAGPOWERSUPPLYSTS = 13;**

**香氛风机状态**

**public static final int TYPE\_FGA\_3\_FRAGFANSTS = 14;**

**香氛未知错误**

**public static final int TYPE\_FGA\_3\_FRAGUNKNOWNERR = 15;**

**硬件主版本号**

**public static final int TYPE\_FGA\_2\_HW\_MAJOR = 16;**

**硬件小版本号**

**public static final int TYPE\_FGA\_2\_HW\_MINOR = 17;**

**软件主版本号**

**public static final int TYPE\_FGA\_2\_SW\_MAJOR = 18;**

**软件主版本号**

**public static final int TYPE\_FGA\_2\_SW\_MINOR = 19;**

## int getPsngrFrntDetct\_D\_Actl()副驾驶是否有人；

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0xFF | 0x0 | Faulty | 副驾坐人 | PsngrFrntDetct\_D\_Actl （msg0x4C:RCMStatusMessage\_HS3） |
| 0x1 | Occupied |
| 0x2 | Empty |
| 0x3 | Unknown |

## interface IPsngrFrntDetctChange { void onPsngrFrntDetctChange(int psngrFrntDetct);}

副驾驶是否有人的回调，**psngrFrntDetct见31.33的赋值；**

## void setSoundStage(int dspSoundStage) 设置声音的方位

public static final int *SOUND\_STAGE\_OFF* = DspSoundStage.*SOUND\_STAGE\_OFF*;  
public static final int *SOUND\_STAGE\_ALL\_PASSENGER* = DspSoundStage.*SOUND\_STAGE\_ALL\_PASSENGER*; 全车  
public static final int *SOUND\_STAGE\_DRIVER* = DspSoundStage.*SOUND\_STAGE\_DRIVER*; 驾驶位；

## int getSoundStage() 声音得方位获取

public static final int *SOUND\_STAGE\_OFF* = DspSoundStage.*SOUND\_STAGE\_OFF*;  
public static final int *SOUND\_STAGE\_ALL\_PASSENGER* = DspSoundStage.*SOUND\_STAGE\_ALL\_PASSENGER*;  
public static final int *SOUND\_STAGE\_DRIVER* = DspSoundStage.*SOUND\_STAGE\_DRIVER*;

## interface ISoundStageChange {

**void onSoundStageChange(int SoundStage);}声音得方位得回调**

## void setStreamVolume(int streamType, int progress) 设置音量

streamTyp；

public static final int *STREAMTYPE\_STREAM\_MUSIC* = AudioManager.*STREAM\_MUSIC*; 媒体音；  
public static final int *STREAMTYPE\_STREAM\_VOICE\_CALL* = AudioManager.*STREAM\_VOICE\_CALL*; 通话音  
public static final int *STREAMTYPE\_STREAM\_VR*=AudioManager.STREAM\_VR; vR  
public static final int *STREAM\_NAVIGATION*=AudioManager.STREAM\_NAVIGATION; 导航

progress的范围：0-30

## int getStreamVolume(int streamType) 获取音量

**streamType：**

public static final int *STREAMTYPE\_STREAM\_MUSIC* = AudioManager.*STREAM\_MUSIC*;  
public static final int *STREAMTYPE\_STREAM\_VOICE\_CALL* = AudioManager.*STREAM\_VOICE\_CALL*;  
public static final int *STREAMTYPE\_STREAM\_VR*=AudioManager.STREAM\_VR;  
public static final int *STREAM\_NAVIGATION*=AudioManager.STREAM\_NAVIGATION;

## void setStreamVolume(int streamType, int progress)设置蓝牙音量（就是31.39，只是蓝牙耳机的type不一样，取值范围不一样，get就是31.40）

streamType；

public static final int *STREAM\_MUSIC\_EXT*=0x10003;//蓝牙耳机

progress**的取值范围：0-10**

# com.ford.vendor.FordSeatRelxPosManager

## void connect(ConnectionCallback connection)连接服务

## void disConnect() 断开服务

## void sendRelaxActivate(int activate)触发 DSM 调节座椅到relax位置

**Activate的值如下；**

|  |  |
| --- | --- |
| 0x0 | FALSE |
| 0x1 | TRUE |

## int getRelaxActivate() 获取座椅relax的位置

**取值如下：**

|  |  |
| --- | --- |
| 0x0 | Fail |
| 0x1 | Success |

## interface IRelaxActivateChange { 座椅relax位置的回调；

**void onReceiveActivateRelax(int result);}**

**result的值如下；**

|  |  |
| --- | --- |
| 0x0 | Fail |
| 0x1 | Success |

## void sendRelaxDeactivate(int deactivate)触发 DSM 调节座椅到relax之前的位置

**Deactivate的值如下；**

|  |  |
| --- | --- |
| 0x0 | FALSE |
| 0x1 | TRUE |

## int getRelaxDeactivate() 获取座椅relax之前的位置；

值如下：

|  |  |
| --- | --- |
| 0x0 | Fail |
| 0x1 | Success |

## public interface IRelaxDeactivateChange {

**void onReceiveActivateRelax(int result);}座椅relax之前的位置变化的回调**

**Result的值如下；**

|  |  |
| --- | --- |
| 0x0 | Fail |
| 0x1 | Success |

## void sendRelaxPosSet(int posset)保存用户自己设置的放松位置

**Posset如下：**

|  |  |  |
| --- | --- | --- |
| 0xFF | 0x0 | Inactive |
| 0x1 | Store |
| 0x2 | Active |
| 0x3 | NotUsed |

## int getRelaxPosSet() 获取用户自己设置的放松位置

|  |  |
| --- | --- |
| 0x0 | Fail |
| 0x1 | Success |

## interface IRelaxPosSetChange {

**void onReceiveRelaxPosSet(int result);}用户自己设置的放松位置变化的回调；**

**Result的值如下：**

|  |  |
| --- | --- |
| 0x0 | Fail |
| 0x1 | Success |

# com.ford.vendor.FordTSRCoDriverManager

## void connect(ConnectionCallback connection)连接服务

## void disConnect() 断开

## int getTrnlpcGear() 获取手动挡信息

**TrnIpcDsplyGear\_D\_Actl**

**Can返回的值如下**

|  |  |  |
| --- | --- | --- |
| 0xFF | 0x0 | Neutral |
| 0x1 | \_1st\_Gear |
| 0x2 | \_2nd\_Gear |
| 0x3 | \_3rd\_Gear |
| 0x4 | \_4th\_Gear |
| 0x5 | \_5th\_Gear |
| 0x6 | \_6th\_Gear |
| 0x7 | \_7th\_Gear |
| 0x8 | \_8th\_Gear |
| 0x9 | \_9th\_Gear |
| 0xA | \_10th\_Gear |
| 0xB | \_11th\_Gear |
| 0xC | \_12th\_Gear |
| 0xD | \_13th\_Gear |
| 0xE | \_14th\_Gear |
| 0xF | No\_Gear\_Selected |

## interface ITrnlpcGearChange {

**void onTrnlpcGearChanged(int result);}手动挡信息回调接口；**

**resut值为TrnIpcDsplyGear\_D\_Actl的返回值**

**见33.3**

## boolean getFogFrontLightStatus() 前雾灯开关

**FogLghtFrontOn\_B\_Stat**

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |

## interface IFogFrontLightStatusChange {

**void onFogLightStatusChanged(boolean result);}前雾灯开关**

**resut值为FogLghtFrontOn\_B\_Stat的返回值**

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |

## boolean getFogRearLightStatus() 后雾灯开关状态

**0x0 off**

**0x1 on**

**FogLghtRearOn\_B\_Stat**

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |

## interface IFogRearLightStatusChange {

**void onFogRearLightStatusChanged(boolean result);}后雾灯开关状态回调接口；FogLghtRearOn\_B\_Stat**

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |

## int getParkLampStatus() 车灯开关

**Parklamp\_Status**

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |
| 0x2 | Unknown |
| 0x3 | Invalid |

## interface IParkLampStatusChange {

**void onParkLampStatusChanged(boolean result);}车灯开关回调接口；**

**Parklamp\_Status**

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |
| 0x2 | Unknown |
| 0x3 | Invalid |

## boolean getHeadLightHiOnStatusFromAhb()远光灯开关

**HeadLghtHiOn\_B\_StatAhb**

**0x0 Off**

**0x1 On**

## interface IHeadLightHiOnStatusFromAhbChange {

**void onHeadLightHiOnStatusFromAhbChanged(boolean result);}远光灯开关回调接口；**

**HeadLghtHiOn\_B\_StatAhb**

**0x0 Off**

**0x1 On**

## boolean getHeadLightHiOnStatus() 远光灯开关

**HeadLghtHiOn\_B\_Stat**

**0x0 off**

**0x1 on**

## interface IHeadLightHiOnStatusChange {

**void onHeadLightHiOnStatusChanged(boolean result);}远光灯开关回调接口；**

**HeadLghtHiOn\_B\_Stat**

**0x0 off**

**0x1 on**

## boolean isAutoMaticHighBeam()是否是自动远光灯/自动前照灯

## int getTirePressSystemStatus() 胎压监测系统状态

**Tire\_Press\_System\_Stat**

## interface ITirePressSystemStatusChange {

**void onTirePressSystemStatusChanged(int result);}胎压监测系统状态回调接口；**

**Tire\_Press\_System\_Stat**

## int getTsrMsgTxt()获取TSR信息

**TsrMsgTxt\_D\_Rq**

## interface ITsrMsgTxtChange {

**void onTsrMsgTxtChanged(int result);}获取TSR信息回调接口；**

**TsrMsgTxt\_D\_Rq**

## int getTsrOswWarnMsgTxt() 获取Osw警告信息

**TsrOswWarnMsgTxt\_D\_Rq**

## interface ITsrOswWarnMsgTxtChange {

**void onTsrOswWarnMsgTxtChanged(int result);}获取Osw警告信息回调接口；**

**TsrOswWarnMsgTxt\_D\_Rq**

## int getTsrOvtStatusMsgTxt() 获取Ovt状态信息

**TsrOvtkStatMsgTxt\_D\_Rq**

## interface ITsrOvtStatusMsgTxtChange {

**void onTsrOvtStatusMsgTxtChanged(int result);}获取Ovt状态信息回调接口；**

**TsrOvtkStatMsgTxt\_D\_Rq**

## int getTsrOvtMsgTxt() 获取Ovt信息

**TsrOvtkMsgTxt\_D\_Rq**

## interface ITsrOvtMsgTxtChange {

**void onTsrOvtMsgTxtChanged(int result);}获取Ovt信息回调接口；**

**TsrOvtkMsgTxt\_D\_Rq**

## int getPersCtaSplitView()PersCtaSplitView

**PersCtaSplitView\_D\_Stat**

## interface IPersCtaSplitViewChange {

**void onPersCtaSplitViewChanged(int result);}PersCtaSplitView回调接口；**

**PersCtaSplitView\_D\_Stat**

## int getTsrStatusMsgTxt() 获取Tsr状态信息

**TsrStatMsgTxt\_D\_Rq**

## interface ITsrStatusMsgTxtChange {

**void onTsrStatusMsgTxtChanged(int result);}获取Tsr状态信息回调接口**

**TsrStatMsgTxt\_D\_Rq**

## int getTsrVL1RstrcMsgTxt() 获取TSR VL1限制信息

## interface ITsrVL1RstrcMsgTxtChange {

**void onTsrVL1RstrcMsgTxtChanged(int result);}获取TSR VL1限制信息回调接口；**

## int getTsrVL1StatusMsgTxt() 获取TSR VL1状态信息

**TsrVl1StatMsgTxt\_D\_Rq**

## interface ITsrVL1StatusMsgTxtChange {

**void onTsrVL1StatusMsgTxtChanged(int result);}获取TSR VL1状态信息回调接口；**

**TsrVl1StatMsgTxt\_D\_Rq**

## int getTsrVL2RstrcMsgTxt() 获取TSR VL2限制信息

**TsrVl2RstrcMsgTxt\_D\_Rq**

## interface ITsrVL2RstrcMsgTxtChange {

**void onTsrVL2RstrcMsgTxtChanged(int result);}获取TSR VL2限制信息回调接口；**

**TsrVl2RstrcMsgTxt\_D\_Rq**

## int getTsrVL2StatusMsgTxt() 获取TSR VL2状态信息

**TsrVl2StatMsgTxt\_D\_Rq**

## interface ITsrVL2StatusMsgTxtChange {

**void onTsrVL2StatusMsgTxtChanged(int result);}获取TSR VL2状态信息回调接口；**

**TsrVl2StatMsgTxt\_D\_Rq**

## int getTsrVLim1MsgTxt()获取TSR VLIM1信息

**TsrVLim1MsgTxt\_D\_Rq**

## interface ITsrVLim1MsgTxtChange {

**void onTsrVLim1MsgTxtChanged(int result);}获取TSR VLIM1信息回调接口；**

**TsrVLim1MsgTxt\_D\_Rq**

## int getTsrVLim2MsgTxt()获取TSR VLIM2信息

**TsrVLim2MsgTxt\_D\_Rq**

## interface ITsrVLim2MsgTxtChange {

**void onTsrVLim2MsgTxtChanged(int result);}获取TSR VLIM2信息的回调**

**TsrVLim2MsgTxt\_D\_Rq**

## int getTsrVIUnitMsgTxt() 获取TSR VI单位信息

**TsrVlUnitMsgTxt\_D\_Rq**

## interface ITsrVIUnitMsgTxtChange {

**void onTsrVIUnitMsgTxtChanged(int result);}获取TSR VI单位信息回调接口；**

**TsrVlUnitMsgTxt\_D\_Rq**

## int getTsrVI1PrmntMsgTxt()获取TSR VI1常驻信息

**TsrVl1PrmntMsgTxt\_D\_Rq**

## interface ITsrVI1PrmntMsgTxtChange {

**void onTsrVI1PrmntMsgTxtChanged(int result);}获取TSR VI1常驻信息回调接口；**

**TsrVl1PrmntMsgTxt\_D\_Rq**

## int getTsrOvtkMsgTxt2()获取TSR Ovtk信息

**TsrOvtkMsgTxt2\_D\_Rq**

## interface ITsrOvtkMsgTxt2Change {

**void onTsrOvtkMsgTxt2Changed(int result);}获取TSR Ovtk信息回调接口；**

**TsrOvtkMsgTxt2\_D\_Rq**

## int getTsrVL1RstrcMsgTxt2() 获取TSR TVL1限制信息

**TsrVl1RstrcMsgTxt2\_D\_Rq**

## interface ITsrVL1RstrcMsgTxt2Change {

**void onTsrVL1RstrcMsgTxt2Changed(int result);}获取TSR TVL1限制信息回调接口；**

**TsrVl1RstrcMsgTxt2\_D\_Rq**

## boolean getWwaWarn() WwaWarn

**WwaWarn\_B\_Rq**

## interface IWwaWarnChange {

**void onWwaWarnChanged(boolean result);}WwaWarn 回调接口；**

**WwaWarn\_B\_Rq**

## int getTsrVL2RstrcMsgTxt2()获取TSR VL2限制信息

**TsrVl2RstrcMsgTxt2\_D\_Rq**

## interface ITsrVL2RstrcMsgTxt2Change {

**void onTsrVL2RstrcMsgTxt2Changed(int result);}获取TSR VL2限制信息回调接口**

**TsrVl2RstrcMsgTxt2\_D\_Rq**

## int getTsrVL2PrmntMsgTxt()获取TSR VL2 常驻信息

**TsrVl2PrmntMsgTxt\_D\_Rq**

## interface ITsrVL2PrmntMsgTxtChange {

**void onTsrVL2PrmntMsgTxtChanged(int result);}获取TSR VL2 常驻信息 回调接口；**

**TsrVl2PrmntMsgTxt\_D\_Rq**

## int getLaSysOffStatus() LaSysOffStatus

**LaSysOffStat\_D\_Dsply**

## interface ILaSysOffStatusChange {

**void onLaSysOffStatusChanged(int result);}LaSysOffStatus 回调接口；**

**LaSysOffStat\_D\_Dsply**

## void setCarDoorOpen() 设置门为打开

## void setCarDoorClose() 设置门为关闭

## void setHeadlightOpen()设置大灯为打开

## void setHeadlightClose()设置大灯为关闭

## void getCarDoorSwitchState() 查询门得状态

## void getHeadlightSwitchState() 查询大灯得状态

## interface ICarDoorSwitchStateChange {

**void onCarDoorSwitchStateChange(int state);}**

**state:1 ：打开； 0：关闭；**

## interface IHeadlightSwitchStateChange {

**void onHeadlightSwitchStateChange(int state);}**

**state:1:打开； 0;关闭；**

## int getEngineState() 发动机状态

|  |  |
| --- | --- |
| 0 | EngOff |
| 1 | EngOn |
| 2 | EngAutoStopped |
| 3 | NotUsed |

## interface IEngineStateChange {

**void onEngineState(int state);}**

**同33.62发动机状态得回调**

## int getClusterSpeed() 获取仪表展示速度

## interface IClusterSpeed {

**void onClusterSpeedChanged(int result);}仪表展示速度得回调**

## int getAhbcStat() 自动远光灯得获取

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |

## interface IAhbcStatChanged {

**void onAhbcStatChanged(int result);}自动远光灯得回调**

## int getAhbStat() 获取自动远光灯得状态

|  |  |
| --- | --- |
| 0x0 | OFF |
| 0x1 | ON |

## interface IAhbStatChanged {

**void onAhbStatChanged(int result);}自动远光灯状态得回调**

## void setCtrStkFeat2ConfigActl(int type) 设置的状态

Type: 0:OFF 1:ON

## void getFeatConfigIpc2Actl()查询状态

## interface IFeatConfigIpc2ActlChange {

**void onFeatConfigIpc2Actl(int result);}**

**状态值 result:**

**0：OFF 1:ON**

## void setCtrStkFeat2ConfigActlCommon(int featureid, int type)

**type：config**

## void getFeatConfigIpc2ActlCommon(int featureid)

## interface IFeatConfigIpc2ActlCommonChange {

**void onFeatConfigIpc2ActlCommon(int featureid, int result);}**

**result:对应featureid的config**

## int getbsBattSOC()

## interface IBsBattSOCChange{

**void onBsBattSOCChange(int result);}**

# com.ford.vendor.FordSunroofManager

## void connect(ConnectionCallback connection)连接服务

## void disConnect() 断开服务

## void setSunroofControl(int TurnAsstSwtch\_D\_Stat)设置天窗的状态

## void setSunshadeControl(int TurnAsstSwtch\_D\_Stat) 设置遮阳板的状态

## int getSunroofDSPLStatusAntiPinch()天窗的状态

## int getSunroofDSPLStatusPosition() 天窗的移动位置

## int getSunroofDSPLStatusMovement() 天窗的移动状态

## int getSunroofDSPLStatusAntiPinchSS()遮阳板的状态

## int getSunroofDSPLStatusPositionSS() 遮阳板的移动位置

## int getSunroofDSPLStatusMovementSS() 遮阳板的移动状态

## interface ISunroofStateChange {

**void onSunroofStateChange(int type, int state);}天窗的回调：**

**State为type对应的回调；**

**Type对应的值如下：**

**public static final int TYPE\_STATE\_SUNROOFDSPLSTATUSANTIPINCH = 0;**

**public static final int TYPE\_STATE\_SUNROOFDSPLSTATUSPOSITION = 1;**

**public static final int TYPE\_STATE\_SUNROOFDSPLSTATUSMOVEMENT = 2;**

**public static final int TYPE\_STATE\_SUNROOFDSPLSTATUSANTIPINCHSS = 3;**

**public static final int TYPE\_STATE\_SUNROOFDSPLSTATUSPOSITIONSS = 4;**

**public static final int TYPE\_STATE\_SUNROOFDSPLSTATUSMOVEMENTSS = 5;**

# com.ford.vendor.FordLiftgateManager

## void connect(ConnectionCallback connection) 连接服务

## void disConnect() 断开服务

## void setLiftgateRelaseCounter(int Counter) 打开后备箱, 参数从1到6循环

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Param2:LiftgateReleaseCounter\_Rq** | | | | | |
| **Type** | **Range** | | **Value** | **功能设置** | **Can信号名** |
| **U8** | **0** | **0xFF** | **0-7** | **null** | **BecRleas\_No\_RqMnu (msg:0x32B APIM\_Data** |

## int getLiftgateAjarSt() 后备箱状态

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Param7:LiftgateAjar\_St | | | | | |
| Type | Range | | Value | 功能设置 | Can信号名 |
| U8 | 0 | 0xFF | 0x0 | Closed | DrStatInnrTgate\_B\_Actl (msg:0x3B2 BodyInfo\_3\_HS3 |
| 0x1 | Ajar |

## interface ILiftgateAjarSt {

**void onLiftgateAjarChange(int state);}后备箱状态的接口；**

**同35.4**

# com.ford.vendor. FordSoaManager

## void connect(ConnectionCallback connection) 连接服务

## String getTcuEsn() 获取tcuesn

## String getTcuIccId() 获取tcuIccid

## interface ITcuBezelDiagnosticDataChange {

**void onTcuBezelDiagnosticDataChange(TcuBezelDiagnosticData data);}**

**TcuEsn IccId得回调**

# com.ford.vendor.FordAmbientLightManager

## void connect(ConnectionCallback connection)连接服务

## void disConnect()断开服务

## void setAmbL\_ALM\_Set(int ambL\_ALM\_Set) 氛围灯开关

|  |  |
| --- | --- |
| 0x00 | invalid |
| 0x01 | close |
| 0x02 | open |

## void setAmbL\_Color\_Mode(int ambL\_Color\_Mode) 氛围灯模式

|  |  |
| --- | --- |
| 0x00 | invalid |
| 0x01 | static |
| 0x02 | dynamic |
| 0x03 | customize |
| 0x04 | music |

## void setAmbL\_DrvMde\_D\_Rq(int ambL\_DrvMde\_D\_Rq)驾驶模式设置

|  |  |
| --- | --- |
| 0x00 | manual |
| 0x01 | auto |

## void setAmbL\_Dynamic\_Color(int ambL\_Dynamic\_Color)动态模式的颜色设置

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## void setAmbL\_Main\_Intensity\_Set(int ambL\_Main\_Intensity\_Set)主亮度设置；

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## void setAmbL\_Music\_Frq\_Level(int ambL\_Music\_Frq\_Level)音乐频率；

|  |  |
| --- | --- |
| 0x0 | 0-63hz |
| 0x1 | 64-127hz |
| 0x2 | 128-255hz |
| 0x3 | 256-511hz |
| 0x4 | 512-1023hz |
| 0x5 | 1023-2047hz |
| 0x6 | 2048-4095hz |
| 0x7 | 4096-8191hz |
| 0x8 | 8192-16383hz |

## void setAmbL\_Music\_Range\_Level(int ambL\_Music\_Range\_Level)音乐范围：

|  |  |
| --- | --- |
| 0x0 | 0-15 |
| 0x1 | 16-31 |
| 0x2 | 32-47 |
| 0x3 | 48-63 |
| 0x4 | 64-79 |
| 0x5 | 80-95 |
| 0x6 | 96-111 |
| 0x7 | 112-127 |

## void setAmbL\_Static\_ColorValue\_Set(int ambL\_Static\_ColorValue\_Set) 静态颜色设置：

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## void setAmbL\_CustomizeType\_Set(int ambL\_CustomizeType\_Set)个性化设置：

|  |  |
| --- | --- |
| 0x00 | Invalid |
| 0x01 | Customize1 |
| 0x02 | Customize2 |
| 0x03 | Customize3 |

## void setAmbL\_Door\_Color\_Value(int ambL\_Door\_Color\_Value)DOOR的氛围灯颜色：

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## void setAmbL\_Door\_Intensity\_Value(int ambL\_Door\_Intensity\_Value)DOOR的氛围灯亮度

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## void setAmbL\_Door\_Swtich(int ambL\_Door\_Swtich) Door的氛围灯开关：

|  |  |
| --- | --- |
| 0x00 | Close |
| 0x01 | Open |

## void setAmbL\_Foot\_Color\_Value(int ambL\_Foot\_Color\_Value)FOOT的氛围灯颜色：

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## void setAmbL\_Foot\_Intensity\_Value(int ambL\_Foot\_Intensity\_Value) Foot的氛围灯亮度；

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## void setAmbL\_Foot\_Swtich(int ambL\_Foot\_Swtich) Foot的氛围灯开关；

|  |  |
| --- | --- |
| 0x00 | Close |
| 0x01 | Open |

## void setAmbL\_IP\_Color\_Value(int ambL\_IP\_Color\_Value) IP的氛围灯颜色：

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## void setAmbL\_IP\_Intensity\_Value(int ambL\_IP\_Intensity\_Value)IP的氛围灯亮度;

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## void setAmbL\_IP\_Swtich(int ambL\_IP\_Swtich) IP的氛围灯开关：

|  |  |
| --- | --- |
| 0x00 | Close |
| 0x01 | Open |

## void setRearLight\_Animation\_Rq(int rearLight\_Animation\_Rq) 尾灯动画设置：

|  |  |
| --- | --- |
| 0x0 | Null |
| 0x1 | Mid\_1 |
| 0x2 | Mid\_2 |
| 0x3 | Mid\_3 |
| 0x4 | High\_1 |
| 0x5 | High\_2 |
| 0x6 | High\_3 |
| 0x7 | Reserved |

## int getAmbL\_ALM\_Set() 氛围灯开关

|  |  |
| --- | --- |
| 0x00 | invalid |
| 0x01 | close |
| 0x02 | open |

## int getAmbL\_Color\_Mode() 氛围灯模式：

|  |  |
| --- | --- |
| 0x00 | invalid |
| 0x01 | static |
| 0x02 | dynamic |
| 0x03 | customize |
| 0x04 | music |

## int getAmbL\_DrvMde\_D\_Rq() 驾驶模式设置：

|  |  |
| --- | --- |
| 0x00 | manual |
| 0x01 | auto |

## int getAmbL\_Dynamic\_Color() 动态模式的颜色;

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## int getAmbL\_Main\_Intensity\_Set() 主亮度设置

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## int getAmbL\_Music\_Frq\_Level()音乐频率：

|  |  |
| --- | --- |
| 0x0 | 0-63hz |
| 0x1 | 64-127hz |
| 0x2 | 128-255hz |
| 0x3 | 256-511hz |
| 0x4 | 512-1023hz |
| 0x5 | 1023-2047hz |
| 0x6 | 2048-4095hz |
| 0x7 | 4096-8191hz |
| 0x8 | 8192-16383hz |

## int getAmbL\_Music\_Range\_Level()音乐范围；

|  |  |
| --- | --- |
| 0x0 | 0-15 |
| 0x1 | 16-31 |
| 0x2 | 32-47 |
| 0x3 | 48-63 |
| 0x4 | 64-79 |
| 0x5 | 80-95 |
| 0x6 | 96-111 |
| 0x7 | 112-127 |

## int getAmbL\_Static\_ColorValue\_Set() 静态颜色：

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## int getAmbL\_CustomizeType\_Set() 个性化设置：

|  |  |
| --- | --- |
| 0x00 | Invalid |
| 0x01 | Customize1 |
| 0x02 | Customize2 |
| 0x03 | Customize3 |

## int getAmbL\_Door\_Color\_Value() Door的氛围灯颜色；

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## int getAmbL\_Door\_Intensity\_Value() Door的氛围灯亮度：

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## int getAmbL\_Door\_Swtich()Door的氛围灯开关：

|  |  |
| --- | --- |
| 0x00 | Close |
| 0x01 | Open |

## int getAmbL\_Foot\_Color\_Value() Foot的氛围灯颜色；

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## int getAmbL\_Foot\_Intensity\_Value() Foot的氛围灯亮度；

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## int getAmbL\_Foot\_Swtich() Foot的氛围灯开关：

|  |  |
| --- | --- |
| 0x00 | Close |
| 0x01 | Open |

## int getAmbL\_IP\_Color\_Value()IP的氛围灯颜色；

|  |  |
| --- | --- |
| 0x0-0x7F | 0-127 |

## int getAmbL\_IP\_Intensity\_Value()IP的氛围灯亮度

|  |  |
| --- | --- |
| 0x0-0x64 | 0-100(%) |

## int getAmbL\_IP\_Swtich() IP的氛围灯开关；

|  |  |
| --- | --- |
| 0x00 | Close |
| 0x01 | Open |

## int getRearLight\_Animation\_Rq() 尾灯动画设置：

|  |  |
| --- | --- |
| 0x0 | Null |
| 0x1 | Mid\_1 |
| 0x2 | Mid\_2 |
| 0x3 | Mid\_3 |
| 0x4 | High\_1 |
| 0x5 | High\_2 |
| 0x6 | High\_3 |
| 0x7 | Reserved |

## interface IAmbientLightChange {

**void onAmbientLightChange(int type, int value);}**

**氛围灯的回调；**

**type:如下： value：对应的type的值；**

**//氛围灯开关**

**public static final int TYPE\_AMBL\_ALM\_SET = 0;**

**//氛围灯模式**

**public static final int TYPE\_AMBL\_COLOR\_MODE = 1;**

**//驾驶模式设置**

**public static final int TYPE\_AMBL\_DRVMDE\_D\_RQ = 2;**

**//动态模式的颜色设置**

**public static final int TYPE\_AMBL\_DYNAMIC\_COLOR = 3;**

**//主亮度设置**

**public static final int TYPE\_AMBL\_MAIN\_INTENSITY\_SET = 4;**

**//音乐频率**

**public static final int TYPE\_AMBL\_MUSIC\_FRQ\_LEVEL = 5;**

**//音乐范围**

**public static final int TYPE\_AMBL\_MUSIC\_RANGE\_LEVEL = 6;**

**//静态颜色设置**

**public static final int TYPE\_AMBL\_STATIC\_COLORVALUE\_SET = 7;**

**//个性化设置**

**public static final int TYPE\_AMBL\_CUSTOMIZETYPE\_SET = 8;**

**//Door的氛围灯颜色**

**public static final int TYPE\_AMBL\_DOOR\_COLOR\_VALUE = 9;**

**//Door的氛围灯亮度**

**public static final int TYPE\_AMBL\_DOOR\_INTENSITY\_VALUE = 10;**

**// Door的氛围灯开关**

**public static final int TYPE\_AMBL\_DOOR\_SWTICH = 11;**

**//Foot的氛围灯颜色**

**public static final int TYPE\_AMBL\_FOOT\_COLOR\_VALUE = 12;**

**//Foot的氛围灯亮度**

**public static final int TYPE\_AMBL\_FOOT\_INTENSITY\_VALUE = 13;**

**//Foot的氛围灯开关**

**public static final int TYPE\_AMBL\_FOOT\_SWTICH = 14;**

**//IP的氛围灯颜色**

**public static final int TYPE\_AMBL\_IP\_COLOR\_VALUE = 15;**

**// IP的氛围灯亮度**

**public static final int TYPE\_AMBL\_IP\_INTENSITY\_VALUE = 16;**

**//IP的氛围灯开关**

**public static final int TYPE\_AMBL\_IP\_SWTICH = 17;**

**//尾灯动画设置**

**public static final int TYPE\_REARLIGHT\_ANIMATION\_RQ = 18;**

# com.ford.vendor.FordARPOCManager

## void connect(ConnectionCallback connection)

## void disConnect()

## int getLaActvStats\_D2\_Dsply()

|  |  |
| --- | --- |
| 0x0 | NoLeft\_NoRight |
| 0x1 | AvailableLeft\_NoRight |
| 0x2 | SuppressLeft\_NoRight |
| 0x3 | WarnLeft\_NoRight |
| 0x4 | InterveneLeft\_NoRight |
| 0x5 | NoLeft\_AvailableRight |
| 0x6 | AvailableLeft\_AvailRight |
| 0x7 | SuppressLeft\_AvailRight |
| 0x8 | WarnLeft\_AvailRight |
| 0x9 | InterveneLeft\_AvailRight |
| 0xA | NoLeft\_SuppressRight |
| 0xB | AvailLeft\_SuppressRight |
| 0xC | SuppressLeft\_SuppressRight |
| 0xD | WarnLeft\_SuppressRight |
| 0xE | InterveneLeft\_SuppressRght |
| 0xF | NoLeft\_WarnRight |
| 0x10 | AvailableLeft\_WarnRight |
| 0x11 | SuppressLeft\_WarnRight |
| 0x12 | WarnLeft\_WarnRight |
| 0x13 | InterveneLeft\_WarnRight |
| 0x14 | NoLeft\_InterveneRight |
| 0x15 | AvailLeft\_InterveneRhtt |
| 0x16 | SuppressLeft\_InterveneRght |
| 0x17 | WarnLeft\_InterveneRight |
| 0x18 | InerveneLeft\_InterveneRght |
| 0x19 | Unused1 |
| 0x1A | Unused2 |
| 0x1B | Unused3 |
| 0x1C | Unused4 |
| 0x1D | ECE\_OFF\_RTT |
| 0x1E | LA\_Off |
| 0x1F | Unused7 |

## int getCcStat\_D\_Actl()

|  |  |
| --- | --- |
| 0x0 | Off |
| 0x1 | Denied |
| 0x2 | Standby\_Denied |
| 0x3 | Standby |
| 0x4 | Active\_Que\_Assist |
| 0x5 | Active |
| 0x6 | Undefined\_1 |
| 0x7 | Undefined\_2 |

## int getAccTrgDist2\_D\_Dsply()

|  |  |
| --- | --- |
| 0x0 | DIST\_OFF |
| 0x1 | DIST\_STANDBY |
| 0x2 | DIST\_ACTIVE\_No\_Target |
| 0x3 | DIST\_ACTIVE\_1\_Closest |
| 0x4 | DIST\_ACTIVE\_2 |
| 0x5 | DIST\_ACTIVE\_3 |
| 0x6 | DIST\_ACTIVE\_4 |
| 0x7 | DIST\_ACTIVE\_5 |
| 0x8 | DIST\_ACTIVE\_6 |
| 0x9 | DIST\_ACTIVE\_7 |
| 0xA | DIST\_ACTIVE\_8 |
| 0xB | DIST\_ACTIVE\_9 |
| 0xC | DIST\_ACTIVE\_10 |
| 0xD | DIST\_ACTIVE\_11 |
| 0xE | DIST\_ACTIVE\_12 |
| 0xF | DIST\_ACTIVE\_13\_Farthest |

## interface IARPocChange {

**void onARPocChange(int type, int value);}**

**type:如下： value是type，对应的值；**

**public static final int TYPE\_LAACTVSTATS\_D2\_DSPLY = 0;**

**// LaActvStats\_D2\_Dsply的回调**

**public static final int TYPE\_CCSTAT\_D\_ACTL = 1;**

**//CcStat\_D\_Actl的回调**

**public static final int TYPE\_ACCTRGDIST2\_D\_DSPLY = 2;**

**//AccTrgDist2\_D\_Dsply对应的回调；**

# com.ford.car.FordNaviautoSpeedManager

## void connect(ConnectionCallback connection)//连接服务

## float getCarSpeed()//获取速度

## interface SensorChangeListener {//速度的回调

**void onSensorChange(float speed);}**

# com.ford.car.FordCommonUtil

## public static boolean isU625()

## public static boolean isU625Timberline()

## public static boolean isCD542H()

## public static boolean isCD542ICAH()

## public static boolean isCD542L()

## public static boolean isCD542ICAL()

## public static boolean isCD764()

## public static boolean isCDX706H()

## public static boolean isCDX706L()

## public static boolean isCX483()

## public static boolean isP702()

## public static boolean isMY23P702()

## public static boolean isMY24P702()

## public static boolean isCX727()

## public static boolean isU554()

## public static boolean isS650()

## public static boolean isCX727ICA()

## public static boolean isU554MY23()

## public static boolean isU554MY24()

## public static boolean isCX727LFP()

## public static boolean isCD764ICA()

## public static boolean is483PTICA()

## public static boolean isU725C()

## public static CarSWVariantEnum getCarSWVariant()

8155对应的Enum:

*CARTYPE\_CD764ICA*,*CARTYPE\_CX727LFP*,*CARTYPE\_CX483PT\_ICA*,  
*CARTYPE\_U725*

820A对应的Enum:

*CARTYPE\_DEFAULT*,*CARTYPE\_CD542L*,*CARTYPE\_P702*,*CARTYPE\_MY23P702*,  
*CARTYPE\_MY24P702*,*CARTYPE\_CD542H*,*CARTYPE\_U625*,

*CARTYPE\_U625Timber*,*CARTYPE\_CX727*,*CARTYPE\_U554*,  
*CARTYPE\_CD764*,*CARTYPE\_CDX706H*,*CARTYPE\_CD542ICAH*,  
*CARTYPE\_CDX706L*,*CARTYPE\_CD542ICAL*,*CARTYPE\_S650*,  
*CARTYPE\_CX483*,*CARTYPE\_CX727ICA*,*CARTYPE\_U554MY23*, *CARTYPE\_U554MY24*,

# com.ford.vendor.FordIodMessageManager

## int getVehElEffTrip\_No\_Dsply()

**当前行程能耗**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x7F | 0x0 | -100 | 单位：wattHr/km Resolution: 10 Offset: -100 | (Msg: 0x39D TripInfoData3\_HS3) |
| 0x7D | 1150 |
| 0x7E | NoDataExists |
| 0x7F | Faulty |
|  |  |
|  |  |
|  |  |
|  |  |

## void queryEnergyConsumptionUnit()

**查询能耗单位**

## int getEnergyConsumptionUnit()

**获取能耗单位**

**0 ：mi/kWh**

**2 ：kWh/100km**

**3 ：km/kWh**

## int getElDistTripUnit\_d\_stat()

**行程单位**

|  |  |  |
| --- | --- | --- |
| 0 | Inactive | ElDistTripUnit\_D\_Stat indicate the unit to display for the Electric Trip Distance (msg: 0x327 InstFuelEconomy\_3) |
| 1 | Kilometers |
| 2 | Miles |
| 3 | Reserved |
| 0x4-0xFF | not update |

## double getElDistTripCur\_No\_Dsply()

**当前行程 里程**

|  |  |  |  |
| --- | --- | --- | --- |
| 0x1FFFF | NA | 低字节 | ElDistTripCur\_No\_Dsply  Resolution: 0.1  (msg: 0x327 InstFuelEconomy\_3) |
| NA | 中字节 |
| NA | 高字节 |
| 0x20000-0xFFFFFF | not update |

## interface IODMessageChangeListener{

**void onIODMessageChange(int type,int value)}**

**type:**

**public static final int EVENT\_TYPE\_VEHELEFFTRIP\_NO\_DSPLY = 0;//当前行程能耗**

**public static final int EVENT\_TYPE\_ENERGY\_CONSUMPTION\_UNIT = 1;//能耗单位**

**public static final int EVENT\_TYPE\_ELDISTTRIPUNIT\_D\_STAT=2;//行程单位**

## interface IElDistTripCur\_No\_DsplyListener{

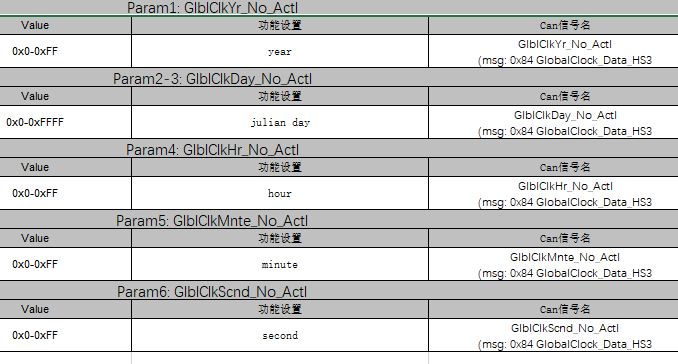
**void onElDistTripCur\_No\_DsplyChange(double value);}**

**当前行程里程的回调**

# com.ford.vendor.FordModeCardManager

## int[] getGlblClk() 时间

int[]：长度为5：分别代表：年/天/时/分/秒



## int getGearPos\_D\_Trg() 挡位方向

挡位方向：取值如下：



## void setCtrStkFeatConfig(int type) 设置DLR的开关

设置DLR的开关

CtrStkDsplyOp\_D\_Rq=0x2 set

CtrStkFeatNoActl=0x0415

CtrStkFeatConfigActl=0x0/0x1

CtrStkPersIndex\_D\_Actl

Type：对应的就是CtrStkFeatConfigActl

## void getFeatConfig() 查询DRL的开关状态

//查询DRL的开关状态

## int[] getVedsBelt安全带的状态

安全带的状态：0x0:No Event 0x1:Belted 0x2:NotBelted

0x3:NotSupported 0x5:Fault

对应关系如下：ints[0]驾驶员 ints[1]乘客侧

ints[2]二排驾驶侧 ints[3]二排中间 ints[4]二排乘客侧

## int getPsngrFrntDetct\_D\_Actl 副驾是否坐人

副驾是否坐人：0x0:Faulty 0x1:Occupied 0x2:Empty

0x3 Unknown

## void operateTheFrontTrunk(boolean open) 打开前备箱

打开前备箱：**在 boolean getFrunkPressPermission()=true的时候在打开**

Open:打开 !open:关闭

## int getHoodStatus() 前备箱的开关状态

前备箱的开关状态；1是打开 非0是关闭

## void operateTheRearTrunk(boolean open)打开后备箱

open:打开 !open:关闭

## int getDrStatTgate\_B\_Actl() 后备箱的开关状态

开关状态（ 0：Closed 1：Ajar）

## int getVeh\_Lock\_Status()//锁车

对应的值如下：



## float getCabnAmb\_Te\_Actl//获取车内温度

## NavSignalsBean getNavSignalsBean()//导航信号 bean 类

private int DistUnits;//距离单位 0：米 1：千米

private int TotalDistTraveled;//总距离

private int TotalTime;//总时间

private String Destination;//目的地名称

private int SpeedLimit;//限速值

private String CurentStreetName;//道路名称

private int UnitOfLength;//长度单位 0：米 1：千米

private int DistanceToNextManeuver;//下一个机动点距离

private String StreetName\_St;//下一道路名称

private int Nav\_DistanToDestUnit\_St;//剩余长度单位 0：米 1：千米

private int Nav\_DistancToDestVal\_St;//剩余长度

private int Nav\_RemainTTDestMin\_St;//剩余分钟

private boolean RouteActive\_St;//导航激活状态指示 boolean false： naviEnd true： naviStart

## public interface INavSignalsListener {//导航信号的回调

**void onNavSignalsChange(NavSignalsBean bean);}**

## interface IModeCardListener {

**void onModeCardChange(int eventType, int[] intValues, float[] floatValues);}**

**对应的type如下: type对应的值在intValues中；**

**public static final int EVENTTYPE\_GLBLCLK = 0;//年月日分秒**

**public static final int EVENTTYPE\_GEARPOS\_D\_TRG = 1;//挡位方向**

**public static final int EVENTTYPE\_CTRSTKFEATCONFIG = 2;//灯光开关**

**public static final int EVENTTYPE\_VEDS = 3;//驾驶位置**

**public static final int EVENTTYPE\_PSNGRFRNTDETCT = 4;//副驾是否有人**

**public static final int EVENTTYPE\_FRONT\_TRUNK = 5;//前备箱**

**public static final int EVENTTYPE\_REAR\_TRUNK = 6;//后备箱**

**public static final int EVENTTYPE\_VEH\_LOCK\_STATUS = 7;//锁车**

**public static final int EVENTTYPE\_CABNAMB\_TE\_ACTL = 8;//车内温度**

**public static final int EVENTTYPE\_DAY\_NIGHT\_STATUS = 9;//光亮**

**public static final int *EVENTTYPE\_WIPRFRONT\_D\_STAT* = 10;//雨量传感器**

**public static final int EVENTTYPE\_PRKBRKSTATUS = 11;//电子手刹**

**public static final int EVENTTYPE\_AEIS = 12;//30分钟怠速开关**

**public static final int EVENTTYPE\_HOTSPOTENABLEMENT = 13;//热点开关**

**public static final int EVENTTYPE\_BALANCEFADER = 14;//BalanceFader 平衡衰减**

## int getDay\_Night\_Status() 光亮

|  |  |  |
| --- | --- | --- |
| 0 | null | Day\_Night\_Status（BodyInfo\_3\_HS3, Rx） |
| 1 | day |
| 2 | night |

## void setTheOrderTimeFromFord(int time) Ford设置时间

## interface ITheReturnResultToFordListener{//返回回调给Ford

**void onReturnsResultToFordChange();}**

## int getWiprFront\_D\_Stat() 雨量传感器

|  |  |  |
| --- | --- | --- |
| 0x0 | OFF | WiprFront\_D\_Stat (Msg: 0x76 BrakeSnData\_HS3) |
| 0x01 | AUTO\_OFF |
| 0x02 | OFF\_MOVING |
| 0x03 | MAN\_INT\_OFF |
| 0x04 | MAN\_INT\_ON |
| 0x05 | MAN\_LOW |
| 0x06 | MAN\_HIGH |
| 0x07 | MIST\_FLICK |
| 0x08 | WASH |
| 0x09 | AUTO\_LOW |
| 0x0A | AUTO\_HIGH |
| 0x0B | COURTESYWIPE |
| 0x0C | AUTO\_ADJUST |
| 0x0D | RESERVED |
| 0x0E | STALLED |
| 0x0F | NO\_DATA\_EXISTS |

## void setAEISFeatConfig(int type) 设置30分钟怠速开关

Type:

0x00- OFF

0x01- ON

## void getAEISFeatConfig() 查询30分钟怠速开关

## int getPrkBrkStatus() 电子手刹状态（手动挡配置）

|  |  |  |
| --- | --- | --- |
| 0 | Not\_Supported | PrkBrkStatus (msg: 0x213 DesiredTorqBrk\_HS3) |
| 1 | Rear\_Caliper\_Closed |
| 2 | Rear\_Caliper\_Transition |
| 3 | RWU\_By\_EPB\_Active |
| 4 | Rear\_Caliper\_Open |
| 5 | EPB\_Limphome\_Active |
| 6 | ECD\_by\_Brake\_ECU\_Active |
| 7 | GeneralFault\_MaintenceMode |
| 0xFF | not update |

## int getTcuAvailableStat()获取TCU的可用状态; 2 是enable，其他当disable处理

## int getHotspotEnableStat() 获取热点的开关1 关 2开

## void switchHotspot(int status) 设置热点的开关 //1 关 2开

## void setBalanceFader(int balance, int fader)//设置平衡衰减

int balance：平衡

int fader：衰减

## int[] getBalanceFaderValue()

int[0]= balance 平衡

int[1]= fader 衰减

## public interface IRouteActive\_StListener {//导航激活状态指示

**void onRouteActive\_StChange(boolean isActive);}**

## public interface IDistanceToNextManeuverListener {//下一个机动点距离

**void onDistanceToNextManeuverChange(int isActive);}**

## public interface IStreetName\_StListener {//下一道路名称

**void onStreetName\_StChange(String s);}**

## public interface ISpeedLimitListener {//限速值

**void onSpeedLimitChange(int s);}**

## public interface ICurentStreetNameListener {//道路名称

**void onCurentStreetNameChange(String s);}**

## public interface INav\_RemainTTDestMin\_StListener {//剩余分钟

**void onNav\_RemainTTDestMin\_StChange(int s);}**

## public interface INav\_DistancToDestVal\_StListener {//剩余长度

**void onNav\_DistancToDestVal\_StChange(int s);}**

## public interface IDestinationListener {//目的地名称

**void onDestinationChange(String s);}**

# com.ford.vendor.FordWiFiManager

## int initialize()初始化

## void WiFiON() 打开wifi

## void WiFiOFF()关闭wifi

## interface IWifiCallback{ wifi开关的回调

**void onWifiSwitchStatus(int status);}** 1是打开 0是关闭

## void getWifiStatus()//获取wifi开关状态

## void scanAPs()//扫描

## boolean getVehicleConnActivityState() //ccs连接状态

## void connectAP(HALWlanProfile halWlanProfile) //主动连接wifi

## void disconnectAP() //主动断开wifi

## void forgetAP(ApInfo apInfo) //忽略wifi

## void enableWifiSta() //开启wir wifi底层进程

## void disableWifiSta() //关闭wir wifi底层进程

## interface IWifiRespListener {

void scanAPsResp(int i, HALScanResults halScanResults);//扫描结果回调

void getWifiStateResp(int i, boolean b);//开关状态回调

void connectAPResp(int i, ApInfo apInfo);//主动连接回调

void disconnectAPResp(int i);//主动断开回调

void forgetAPResp(int i);//忽略回调

void disableSyncWifiStaModeResp(int i);//关闭wir wifi底层进程回调

void enableSyncWifiStaModeResp(int i);//开启wir wifi底层进程回调

void onWifiConnected(HALWlanProfile halWlanProfile);//wifi连接回调

void onWifiDisconnected();}//wifi断开回调

# com.ford.vendor.FordConnectivityManager

## long getUidIfaceRxBytes(String packageName, int networkType)

统计app接收的字节数（即接收流量），返回值为查询的流量大小

## long getUidIfaceTxBytes(String packageName, int networkType)

统计app发送的字节数（即发送流量), 返回值为查询的流量大小

## void disableAccessNetwork(String packageName, int networkType)

禁止app访问某个网络类型 无返回值

## void enableAccessNetwork(String packageName, int networkType)

启用app访问某个网络类型 无返回值

**参数说明：**

**String packageName：代表要操作的app的包名**

**int networkType: 代表要操作的网口的类型，即apn1，apn2，tcu wifi，sync wifi等。**

**具体变量值定义如下：**

**public static final int NETWORK\_TYPE\_APN\_1 = 1;**

**public static final int NETWORK\_TYPE\_APN\_2 = 2;**

**public static final int NETWORK\_TYPE\_TCU\_WIFI = 3;**

**public static final int NETWORK\_TYPE\_SYNC\_WIFI = 4;**

# com.ford.vendor.FordAudioSettingManager

## 方位：不区分内外置 31.39/31.40/31.41

**内外置判断：28.11**

## SoundSurround

**A:内置：31.4/31.5/31.6**

**B:外置：set接口同31.4**

**C:int getSurndSndUpmix2\_D\_Stat() get接口**

**D:interface ISurndSndUpmix2\_D\_StatListener{ 回调接口**

**void onSurndSndUpmix2\_D\_StatChange(int state);}**

|  |  |
| --- | --- |
| 0x00 | Inactive |
| 0x01 | Stereo |
| 0x02 | Surround |
| 0x03 | On\_Stage |
| 0x04 | Audience |

## 音效模式：

**A:内置：31.1/31.2/31.3**

**B:外置：set接口同31.1**

**C:int getDSP\_Sur\_Sound\_St() get接口**

**D:interface IDSP\_Sur\_Sound\_StListener{ 回调接口**

**void onDSP\_Sur\_Sound\_StListener(int state);}**

|  |  |
| --- | --- |
| 0x00 | Inactive |
| 0x01 | Stereo |
| 0x02 | Surround |

## 3d:不区分内外置

**A:int getSndImmrsnLvl\_D\_Stat() get接口**

*0:立体声  
immersion > 0 && immersion <= 64:观众  
immersion > 64 && immersion <= 127：舞台效果*

**B:interface ISndImmrsnLvl\_D\_StatListener { 回调接口**

**void onSndImmrsnLvl\_D\_StatChange(int state);}**

**C:void setEngExhMdeHrEnd\_D\_RqParam36(int Param36) set接口**

# com.ford.vendor.FordOffRoadManager

## int getAwd2wdLamp\_D\_RqDsply()

0x0：Off 0x1：On 0x2：Flash 0x3：Not\_Used

## int getAwdAutoLamp\_D\_RqDsply()

0x0：Off 0x1：On 0x2：Flash 0x3：Not\_Used

## int getAwdHiLamp\_D\_RqDsply()

0x0：Off 0x1：On 0x2：Flash 0x3：Not\_Used

## int getAwdLoLamp\_D\_RqDsply()

0x0：Off 0x1：On 0x2：Flash 0x3：Not\_Used

## int getStePinCompAnEst\_D\_Qf()

0x0:Faulty 0x1:No\_Data\_Exists 0x2:Degraded 0x3:OK

## int getVeh\_V\_ActlEng()

0x0：Off 0x1：Standby 0x2：AvailableLeft

0x3：AvailableRight 0x4：AvailableLeftRight

0x5：PreparingLeft 0x6：PreparingRight

0x7：LcActiveLeft 0x8：LcActiveRight

0x9：NotUsed\_1 0xA：NotUsed\_2

0xB：NotUsed\_3 0xC：NotUsed\_4

0xD：NotUsed\_5 0xE：NotUsed\_6

0xF：NotUsed\_7

## int getVehVActlEng\_D\_Qf()

0x0:Faulty 0x1:No\_Data\_Exists

0x2:Degraded 0x3:OK

## int getStePinComp\_An\_Est()

0x0:Faulty 0x1:No\_Data\_Exists

0x2:Degraded 0x3:OK

## int getStabBarCnnCtlamp\_D\_Rq()

0x0:Faulty 0x1:No\_Data\_Exists

0x2:Degraded 0x3:OK

## int getFrontDiffLckLamp\_D\_Rq()

0x0:Off 0x1:On

0x2:Flash 0x3:Triggered

## int getRearDiffLckLamp\_D\_Rq()

0x0:Off 0x1:On

0x2:Flash 0x3:Not\_Used

## int getOrtaMde\_D\_Ind()

0x0:Off 0x1:StandbyLeft

0x2:StandbyRight 0x3:ActiveLeft

0x4:ActiveRight 0x5:NotUsed\_1

0x6:NotUsed\_2 0x7:Faulty

## int getVehPtch\_An\_Dsply()

0x7E:NoDataExists

0x7F:Faulty

## int getVehRol\_An\_Dsply()

0x7E:NoDataExists

0x7F:Faulty

## interface IOffRoadResponseListener {

void onOffRoadResponseChange(int eventType, int value);}

回调对应can信号的值见get接口；

eventType:序号

public static final int EVENT\_TYPE\_AWD2WDLAMP\_D\_RQDSPLY = 0;（Awd2wdLamp\_D\_RqDsply）

public static final int EVENT\_TYPE\_AWDAUTOLAMP\_D\_RQDSPLY = 1;（AwdAutoLamp\_D\_RqDsply）

public static final int EVENT\_TYPE\_AWDHILAMP\_D\_RQDSPLY = 2; （AwdHiLamp\_D\_RqDsply）

public static final int EVENT\_TYPE\_AWDLOLAMP\_D\_RQDSPLY = 3; （AwdLoLamp\_D\_RqDsply）

public static final int EVENT\_TYPE\_STEPINCOMPANEST\_D\_QF = 4; （StePinCompAnEst\_D\_Qf）

public static final int EVENT\_TYPE\_VEH\_V\_ACTLENG = 5; （Veh\_V\_ActlEng）

public static final int EVENT\_TYPE\_VEHVACTLENG\_D\_QF = 6; （VehVActlEng\_D\_Qf）

public static final int EVENT\_TYPE\_STEPINCOMP\_AN\_EST = 7; （StePinComp\_An\_Est）

public static final int EVENT\_TYPE\_STABBARCNNCTLAMP\_D\_RQ = 8; （StabBarCnnCtlamp\_D\_Rq）

public static final int EVENT\_TYPE\_FRONTDIFFLCKLAMP\_D\_RQ = 9; （FrontDiffLckLamp\_D\_Rq）

public static final int EVENT\_TYPE\_REARDIFFLCKLAMP\_D\_RQ = 10; （RearDiffLckLamp\_D\_Rq）

public static final int EVENT\_TYPE\_ORTAMDE\_D\_IND = 11; （OrtaMde\_D\_Ind）

public static final int EVENT\_TYPE\_VEHPTCH\_AN\_DSPLY = 12; （VehPtch\_An\_Dsply）

public static final int EVENT\_TYPE\_VEHROL\_AN\_DSPLY = 13; （VehRol\_An\_Dsply）

value：对应type对应的can信号的值（序号后面对应的是can信号）