AUTELMobileSDK 功能说明

文档变更记录

版本	变更说明	日期	作者	审核
1.1.0	创建文档	2023-02-27	毛威	

1.概述

本教程旨在帮助您基本了解AUTELMobileSDK 的相关功能以及相关接口的使用。AUTELMobileSDK包含了与 飞机各个模块通信的服务，主要包含通用模块、飞行任务模块、AI服务模块、相机模块、飞行控制模块、飞行参数模块、云台模块、视觉模块以及图传模块，同时也提供飞机连接状态的监听以及遥控器模块的相关功能。下面为AUTELMobileSDK讲解如何在开发中使用这些功能

2.Android Studio 配置工程

2.1 新建一个Android工程

2.2 导入aar包

将aar包复制到工程的libs目录下，并在build.gradle中添加以下代码

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

```
implementation(name: 'autel-sdk-release', ext: 'aar')
```

2.3 基础权限申请，在AndroidManifest.xml文件中添加以下元素

```
 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

3.快速开始

3.1 初始化SDK及参数说明

param1 上下文对象，使用Application context；

Param2 是否为Debug模式；

Param3 自定义存储组件，不传默认使用SharedPreferences；

Param3 自定义日志组件，不传默认使用系统Log且release无log；

```
SDKManager.get().init(applicationContext, AppInfoManager.isBuildTypeDebug(),
AppStorage(), AppLog())
```

3.2 核心操作类KeyManager相关方法介绍

3.2.1 getValue：一般是用来获取飞机上的某个属性，以获取相机设备信息为例：

```
val key = KeyTools.createKey(CameraKey.KeyCameraDeviceInfo)
DeviceManager.getDeviceManager().getFirstDroneDevice()?.getKeyManager()?.getValu
e(key,
  object : CommonCallbacks.CompletionCallbackWithParam<DeviceInfoBean> {
    override fun onSuccess(t: DeviceInfoBean?) {
    }
    override fun onFailure(error: IAutelCode, msg: String?) {
    }
  })
```

3.2.2 setValue：一般是给飞机设置某个属性，以设置相机工作模式为例：

```
val key = KeyTools.createKey(CameraKey.KeyCameraWorkMode)
DeviceManager.getDeviceManager().getFirstDroneDevice()?.getKeyManager()?.setValu
e(key,
  CameraWorkModeEnum.PHOTO,
  object : CommonCallbacks.CompletionCallback {
    override fun onSuccess() {
    }
    override fun onFailure(code: IAutelCode, msg: String?) {
    }
  })
```

3.2.3 performAction：一般是执行某个动作或者命令，以开始拍照为例：

```
val key = KeyTools.createKey(CameraKey.KeyStartTakePhoto)
DeviceManager.getDeviceManager().getFirstDroneDevice()?.getKeyManager()
  ?.performAction(key, null,
    object : CommonCallbacks.CompletionCallbackWithParam<Void> {
      override fun onSuccess(t: Void?) {
      }
      override fun onFailure(code: IAutelCode, msg: String?) {
      }
    })
```

3.2.4 listen:一般是用来接收飞机上报的一些信息，以相机专业参数信息上报为例：

```
val key = KeyTools.createKey(CameraKey.KeyProfessionalParamInfo)
val callback = object : CommonCallbacks.KeyListener<ProfessionalParamInfoBean> {
  override fun onValueChange(
    oldValue: ProfessionalParamInfoBean?,
    newValue: ProfessionalParamInfoBean
  ) {
  }
}
DeviceManager.getDeviceManager().getFirstDroneDevice()?.getKeyManager()
  ?.listen(key, callback)
```

3.2.5 cancelListen:取消前面注册的监听，以相机专业参数信息上报为例：

```
val key = KeyTools.createKey(CameraKey.KeyProfessionalParamInfo)
val callback = object : CommonCallbacks.KeyListener<ProfessionalParamInfoBean> {
  override fun onValueChange(
    oldValue: ProfessionalParamInfoBean?,
    newValue: ProfessionalParamInfoBean
  ) {
  }
}
DeviceManager.getDeviceManager().getFirstDroneDevice()?.getKeyManager()
  ?.cancelListen(key, callback)
```

3.2.6 removeAllListen:取消前面注册的全部监听

```
DeviceManager.getDeviceManager().getFirstDroneDevice()?.getKeyManager()?.removeA
llListen()
```

3.3 飞机连接状态和相机能力集监听

```
interface IAutelDroneListener {
    /**
     * @param connected  is connected
     * @param drone drone device
     * @Description: listener for drone device connecting status
     */
    fun onDroneChangedListener(connected: Boolean, drone: IBaseDevice)

    /**
     *相机能力集变更通知
     * @param fetched true 更新了能力集文件，false 飞机断链了
     */
    fun onCameraAbilityFetchListener(fetched: Boolean)
}
```

3.3.1 注册飞机连接状态和相机能力集的监听

```
SDKManager.get().getDeviceManager().addDroneListener(this)
```

3.3.2 获取当前飞机的连接状态

```
val connectStatus = DeviceManager.getDeviceManager().isConnected()
```

3.4 获取相机能力集相关信息

```
val cameraSupport
=DeviceManager.getFirstDroneDevice()?.getCameraAbilitySetManger()?.getCameraSupp
ort()
```

相机能力集是指：当前相机具备哪些能力，相机能力集是以ICameraSupport这个接口返回，该接口类提供了相机相关能力

```
interface ICameraSupport {

    /**
     *  @return 返回当前支持的有效视频分辨率和帧率列表
     */
    fun getResolutionAndFrameRate(lensType: LensTypeEnum, flightMode:
FightModeEnum, modeEnum: RecordModeEnum): List<VideoResolutionFrameBean>


    /**
     *  @return 返回支持HDR对应的相片分辨率列表
     */
    fun getHDRSupportPhoto(flightMode: FightModeEnum, photoFormat:
PhotoFormatEnum, modeEnum: CameraModeEnum): List<PhotoResolutionEnum>


    /**
     *  @return 返回当前有效的相机模式(CameraModeEnum).
     */
    fun getCameraModeRange(lensType : LensTypeEnum,
                           flightMode: FightModeEnum = FightModeEnum.Manual,
modeEnum: TakePhotoModeEnum = TakePhotoModeEnum.UNKNOWN):
ArrayList<CameraModeEnum>

    /**
     * @return 返回手动变焦的 min max step的数值范围   "Default": {"Min": 1,"Max":
50,"Step": 1 }
     */
    fun getManualFocus(): RangeStepIntValue

    /**
     * @return 返回相机变焦的尺寸
     */
    fun getPhotoZoom(lensType : LensTypeEnum): RangeStepValue

    /**
     * @return 返回视频变焦的尺寸
     */
    fun getVideoZoom(lensType : LensTypeEnum,
                     videoZoomType : VideoZoomTypeEnum =
VideoZoomTypeEnum.Default): RangeStepValue
```

```
    /**
     * @return 返回水印和时间戳的取值列表
     */
     fun getWatermarkTimestamp(lensType: LensTypeEnum, photoFormat:
PhotoFormatEnum): Int

    /**
     *  @return 返回当前有效的相机曝光模式(ExposureModeEnum).
     */
    fun getExposureModeRange(): ArrayList<ExposureModeEnum>

    /**
     *  @return 返回当前有效的曝光补偿范围(ExposureExposureCompensationEnum).
     */
    fun getExposureCompensationRange():
ArrayList<ExposureExposureCompensationEnum>

    /**
     *  @return 返回当前有效的相机ISO范围 (ImageISOEnum).
     */
    fun getImageISOList(isPhoto: Boolean = true,
                        pattern: Int = CameraPatternEnum.MANUAL_FLIGHT.value,
                        modeEnum: TakePhotoModeEnum =
TakePhotoModeEnum.UNKNOWN): ArrayList<ImageISOEnum>


    /**
     *  @return 返回当前有效的相机ISO模式列表 (ISOModeEnum).
     */
    fun getPhotoISOModeRange(): List<ISOModeEnum>

    /**
     *  @return 返回当前有效的相机ISO模式列表 (ISOModeEnum).
     */
    fun getVideoISOModeRange(): List<ISOModeEnum>

    /**
     *  @return 返回当前有效的相机快门速度范围(ShutterSpeedEnum).
     */
    fun getShutterList(isPhoto: Boolean, fps: Int,
                       modeEnum: TakePhotoModeEnum = TakePhotoModeEnum.UNKNOWN):
ArrayList<ShutterSpeedEnum>


    /**
     *  @return 返回当前相机光圈可设置的范围
     */
    fun getApertureRange(): ArrayList<LrisEnum>

    /**
     *  @return 返回当前相机视频格式可选择的范围
     */
    fun getVideoFileFormatRange(lensType : LensTypeEnum): List<VideoFormatEnum>

    /**
     *  @return 返回当前相机录像时拍照间隔的可选择的范围
     */
    fun getPicInVideoIntervalRange(lensType : LensTypeEnum): List<VideoPivEnum>
```

```
    /**
     *  @return 返回当前相机视频标准可选择的范围
     */
    fun getVideoStandardRange(lensType : LensTypeEnum): List<VideoStandardEnum>

    /**
     *  @return 返回当前相机拍照图片格式可选择范围.
     */
    fun getPhotoFileFormatRange(lensType : LensTypeEnum, modeEnum:
TakePhotoModeEnum): List<PhotoFormatEnum>

    /**
     *  @return 返回当前相机快拍张数可选择范围
     */
    fun getPhotoBurstCountRange(lensType: LensTypeEnum): List<Int>

    /**
     *  @return 返回当前相机AEB拍摄张数可选择范围.
     */
    fun getPhotoAEBCaptureCountRange(): List<Int>

    /**
     *  @return 返回当前相机定时拍摄时间可选择范围.
     */
    fun getPhotoIntervalParamRange(lensType: LensTypeEnum = LensTypeEnum.Zoom):
List<Int>


    /**
     *  @return 返回当前相机白平衡可选择范围.
     */
    fun getWhiteBalanceList(): ArrayList<WhiteBalanceEnum>


    /**
     *  @return 返回当前相机白平衡自定义色温值范围.
     */
    fun getCustomColorTemperatureRange(): RangeStepIntValue


    /**
     *  @return 返回当前相机透雾模式范围.
     */
    fun getDehazeModeRange(): List<DefogModeEnum>


    /**
     *  @return 返回当前相机透雾使能状态可选择范围.
     */
    fun getDehazeSettingSwitchRange(): List<DefogEnum>

    /**
     *  @return 返回当前相机透雾使能状态可选择范围.
     */
    fun getDehazeSettingSwitchMergeRange(): List<DefogEnum>

    /**
```

```
     *  @return 返回当前相机抗闪烁模式可选择范围.
     */
    fun getAntiFlickerRange(): List<Int>

    /**
     *  @return TransferMode 是图传的清晰度: 1 是流畅 720p, 2是高清 1080p, 3是超高清
2.7K
     */
    fun getTransferMode(): List<VideoTransMissionModeEnum>

    /**
     *  @return 返回当前相机图像分辨率可选择范围.
     */
    fun getPhotoResolution(lensType : LensTypeEnum = LensTypeEnum.Zoom,
                           flightMode: FightModeEnum,
                           modeEnum: TakePhotoModeEnum):
List<PhotoResolutionEnum>


    /**
     *  @return 返回当前相机图像分辨率可选择范围.无对应枚举 返回UNKNOWN后 重新获取相册分辨
率列表
     */
    fun getPhotoResolutionTwice(lensType : LensTypeEnum = LensTypeEnum.Zoom,
                           flightMode: FightModeEnum,
                           modeEnum: TakePhotoModeEnum): List<PhotoResolution>


    /**
     *  @return 返回当前相机锐度可选择范围.
     */
    fun getSharpnessRange(modeEnum: TakePhotoModeEnum): List<Int>

    /**
     *  @return 返回当前相机对比度可选择范围.
     */
    fun getContrastRange(modeEnum: TakePhotoModeEnum): List<Int>

    /**
     *  @return 返回当前相机饱和度可选择范围.
     */
    fun getSaturationRange(modeEnum: TakePhotoModeEnum): List<Int>

    /**
     *  @return 返回当前相机对焦模式可选择范围.
     */
    fun getLensFocusModeRange(modeEnum: CameraModeEnum): List<Int>

    /**
     *  @return 返回当前相机支持的热成像伪彩信息.
     */
    fun supportedIrColor(): List<ThermalColorEnum>


    /**
     *  @return 返回热成像图测温模式.
     */
    fun getThermalIRTempMode(): List<IRTempModeEnum>
```

```
    /**
     *  @return 返回热成像图像模式.
     */
    fun getThermalIrImageMode(): IrImageMode?


    /**
     *  @return 返回热成像图像增强.
     */
    fun getThermalIrImageEnhance(): RangeStepIntValue?


    /**
     *  @return 返回热成像图像去噪.
     */
    fun getThermalIrNr(): MutableList<Int>

    /**
     *  @return 返回热成像图像去噪.
     */
    fun getIrGain(): IrGain?


    /**
     *  @return 返回热成像等温线.
     */
    fun getIrIsoThermMode(): List<Int>


    /**
     *  @return 返回热成像温度告警.
     */
    fun getIrTempAlarm(): IRHotColdValue?


    /**
     *  @return 返回热成像辐射.
     */
    fun getIrNrEmit(): RangeStepIntValue?
    /**
     *  @return 返回当前相机支持的视频文件压缩标准.
     */
    fun getVideoFileCompressionStandard(): List<VideoCompressStandardEnum>

    /**
     *  @return 返回当前相机支持的存储类型.
     */
    fun getStorageType(lensType: LensTypeEnum): List<StorageTypeEnum>

    fun getVersion():String

}
```

3.5 飞机状态信息缓存

```
val status =
DeviceManager.getDeviceManager().getFirstDroneDevice()?.getStateMachine()
```

飞机状态信息缓存主要是用来保存飞机的一些重要信息，这些信息也是会不断刷新的，主要是保持在DroneStateMachineBean中

```
data class DroneStateMachineBean(
    /** 飞机ID */
    val deviceId: Int,
    /** 系统初始化数据 */
    var systemInitData: SystemInfoData? = null,
    /** 飞机组件集合 */
    var componentList: List<DroneVersionItemBean> = mutableListOf(),
    /** 相机状态Map,key是相机的id */
    val cameraStateMachineList: MutableMap<Int, CameraStateMachineBean> =
mutableMapOf(),
    /** 云台状态Map,key是云台的id */
    val gimbalStateMachineList: MutableMap<Int, GimbalStateMachineBean> =
mutableMapOf(),
    /** 飞机通用参数上报（5HZ） */
    var droneSystemStateHFNtfyBean: DroneSystemStateHFNtfyBean? = null,
    /** 飞机通用参数上报 (2HZ) */
    var droneSystemStateLFNtfyBean: DroneSystemStateLFNtfyBean? = null,
    /** 飞机工作状态信息 */
    var flightControlStatusInfo: FlightControlStatusInfo? = null,
    /** 飞机告警 */
    var droneWarningStateNtfyBean: DroneWarningStateNtfyBean? = null,

    )
```

3.6 AutelKeyInfo和AutelActionKeyInfo相关介绍

3.6.1 AutelKeyInfo包含了单个接口属性和能力，下面以光圈大小的key为例：

参数说明：component.value指这个key是属于哪个模块，可能是相机、飞控、云台等模块；

CameraKeyConstants.ApertureSize指这个key的唯一标识字段；

AutelDoubleConvert指这个数据转换类，用于protobuf和Java之间的对象转换，数据类型对应的是Double；

canGet(true).canSet(true)是指这个key具体哪些能力，主要是四种能力canGet(true)表示具体getValue的能力，跟前面提到的KeyManager是对应的，

canSet(true)表示具备setValue的能力，canPerformAction(true)表示具备performAction的能力，canListen(true)表示具体设置监听和取消监听的能力

```
/** 光圈大小 */
val KeyApertureSize: AutelKeyInfo<Double> = AutelKeyInfo(component.value,
CameraKeyConstants.ApertureSize, AutelDoubleConvert())
    .canGet(true).canSet(true)
```

3.6.2 AutelActionKeyInfo是AutelKeyInfo的子类，包含了AutelKeyInfo的能力，不同之处是这个类有两个泛型，分别表示请求参数类型和响应数据类型，以开始录像为例：

```
/** 开始录像 */
val KeyStartRecord: AutelActionKeyInfo<Void, Void> =
    AutelActionKeyInfo(component.value, CameraKeyConstants.StartRecord,
AutelEmptyConvert(), AutelEmptyConvert())
        .canPerformAction(true)
```

3.7 飞机各个模块的Key相关介绍

3.7.1 通用模块

```
object CommonKey {
    private val component = ComponentType.COMMON

    /** 飞机保活心跳 */
    val KeyHeartBeatPhone: AutelKeyInfo<Void> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.LISTENER_HEARTBEAT_PHONE,
        AutelEmptyConvert()
    ).canListen(true)

    /** APP保活心跳 */
    val KeyHeartBeatApp: AutelKeyInfo<Void> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.LISTENER_HEARTBEAT_APP,
        AutelEmptyConvert()
    ).canListen(true)

    /** 设置系统时间 */
    val KeySetSystemDataTime: AutelActionKeyInfo<SystemTimeInfoBean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.SET_SYSTEM_DATA_TIME_MSG,
        SystemTimeConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 获取系统初始化数据 */
    val keyGetSystemInitData: AutelActionKeyInfo<Void, SystemInfoData> =
AutelActionKeyInfo(//获取系统初始化数据
        component.value,
        MessageTypeConstant.GET_SYSTEM_INIT_DATA_MSG,
        AutelEmptyConvert(),
        SystemInitDataConvert()
    ).canPerformAction(true)

    /** 获取飞机设备信息 */
    val KeyGetDroneDevicesInfo: AutelActionKeyInfo<Void,
List<DroneVersionItemBean>> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.GET_DRONE_DEVICES_INFO_MSG,
        AutelEmptyConvert(),
        SystemProfileInfoConvert(),
    ).canPerformAction(true)

    /** 飞机通用参数上报（5HZ） */
```

```
    val keyDroneSystemStatusHFNtfy: AutelKeyInfo<DroneSystemStateHFNtfyBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_SYSTEM_STATUS_HF_NTFY,
        DroneSystemStateHFNtfyConverter()
    ).canListen(true)

    /** 飞机通用参数上报（2HZ） */
    val KeyDroneSystemStatusLFNtfy: AutelKeyInfo<DroneSystemStateLFNtfyBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_SYSTEM_STATUS_LF_NTFY,
        FlightControlStateLFConvert()
    ).canListen(true)

    /** 飞机工作状态信息上报 */
    val KeyDroneWorkStatusInfoReport: AutelKeyInfo<FlightControlStatusInfo> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_WORK_STATUS_INFO_NTFY,
        WorkStatusInfoConvert()
    ).canListen(true)

    /** 飞机告警上报 */
    val KeyDroneWarningMFNtfy: AutelKeyInfo<DroneWarningStateNtfyBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_WARNING_MF_NTFY,
        FlightControlWarningStateConvert()
    ).canListen(true)

    /** 遥控器定频上报(需要先开启遥控器定频上报) */
    val KeyRCHardwareState: AutelKeyInfo<RCHardwareStateNtfyBean> =
AutelKeyInfo(
        ComponentType.REMOTE_CONTROLLER.value,
        MessageTypeConstant.RC_HARDWARE_STATE_NTFY,
        RCHardwareStateConvert()
    ).canListen(true)

    /** 遥控器按钮触发上报 */
    val KeyRCHardwareInfo: AutelKeyInfo<HardwareButtonInfoBean> = AutelKeyInfo(
        ComponentType.REMOTE_CONTROLLER.value,
        MessageTypeConstant.RC_HARDWARE_BUTTON_INFO_NTFY,
        RCHardwareButtonInfoConvert()
    ).canListen(true)

    /** 遥控器状态上报 */
    val KeyRCState: AutelKeyInfo<RCStateNtfyBean> = AutelKeyInfo(
        ComponentType.REMOTE_CONTROLLER.value,
        MessageTypeConstant.RC_STATE_NTFY,
        RemoteStateConvert()
    ).canListen(true)

    /** 遥控器校准上报 */
    val KeyRCRockerCalibrationState:
AutelKeyInfo<RockerCalibrationStateNtfyBean> = AutelKeyInfo(
        ComponentType.REMOTE_CONTROLLER.value,
        MessageTypeConstant.RC_ROCKER_CALIBRATION_STATE_NTFY,
```

```
        RockerCalibrationStateConvert()
    ).canListen(true)

    /** 控制单个飞机LED灯 */
    val KeyControlLed: AutelActionKeyInfo<DroneLedStatusBean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_CONTROL_LED_MSG,
        DroneLedStatusConverter(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 查询飞机所有LED灯状态 */
    val KeyQueryLedStatus: AutelActionKeyInfo<Void, DroneAllLedStatusBean> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_QUERY_LED_STATUS_MSG,
        AutelEmptyConvert(),
        DroneAllLedStatusConverter()
    ).canPerformAction(true)

    /** 通用校准指令 */
    val KeyDroneCalibrationCommand: AutelActionKeyInfo<CalibrationCommandBean,
Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_CALIBRATION_COMMAND_MSG,
        CalibrationCommandConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 校准事件通知 */
    val KeyDroneCalibrationEventNtfy: AutelKeyInfo<CalibrationEventBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_CALIBRATION_EVENT_NTFY,
        CalibrationEventConvert(),
    ).canListen(true)

    /** 校准进度通知 */
    val KeyDroneCalibrationScheduleNtfy: AutelKeyInfo<CalibrationScheduleBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_CALIBRATION_SCHEDULE_NTFY,
        CalibrationScheduleConvert(),
    ).canListen(true)

    /** 控制LED夜航灯 */
    val KeyDroneControlNightNavigationLed: AutelActionKeyInfo<Boolean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_CONTROL_NIGHT_NAVIGATION_LED_MSG,
        AutelIntToBoolConverter(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 飞机设备信息通知 */
    val KeyDroneVersionNtfy: AutelKeyInfo<List<DroneVersionItemBean>> =
AutelKeyInfo(
```

```
        component.value,
        MessageTypeConstant.DRONE_VERSION_NTFY,
        SystemProfileInfoConvert(),
    ).canListen(true)

    /** 飞机事件通知 */
    val KeyDroneEventNtfy: AutelKeyInfo<EventInfoBean> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_EVENT_NTFY,
        DroneEventConvert()
    ).canListen(true)

    /** 设备临时连接上报 飞机未连接时可以收到 */
    val KeyDroneTempConnectNtfy: AutelKeyInfo<DeviceTempConnectBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_TEMP_CONNECT_NTFY,
        DeviceTempConnectConvert()
    ).canListen(true)

    /** 飞机GPS UTC时间同步 */
    val KeyDroneUtcTimeSyncNtfy: AutelKeyInfo<DroneUTCTimeSyncBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_UTC_TIME_SYNC_NTFY,
        DroneUTCTimeSyncConvert()
    ).canListen(true)

    /** 设置飞机国家码 */
    val KeyDroneSetCountryCode: AutelActionKeyInfo<String, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_SET_COUNTRY_CODE_MSG,
        AutelStringConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** RCType类型通知 */
    val KeyRCBandInfoTypeNtfy: AutelKeyInfo<RCBandInfoTypeBean> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.RC_BAND_INFO_TYPE_NTFY,
        RCBandInfoTypeConvert(),
    ).canListen(true)

    /** 清除禁飞区文件 */
    val KeyDroneCleanNoflyZone: AutelActionKeyInfo<CleanNoFlyZoneEnum, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_CLEAN_NOFLY_ZONE_COMMAND_MSG,
        CleanNoFlyZoneConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 飞机告警通知 */
    val KeyDroneWarning: AutelKeyInfo<List<WarningAtom>> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_WARNING_NTFY,
        WarningInfoConvert(),
```

```
    ).canListen(true)

    /** 飞机实时告警通知 */
    val KeyDroneRuntimeWarning: AutelKeyInfo<WarningAtom> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.DRONE_RUNTIME_WARNING_NTFY,
        WarningAtomConvert(),
    ).canListen(true)

}
```

3.7.2 飞行任务模块

```
object FlightMissionKey{

    val component = ComponentType.MISSION

    // ----------------- 航点任务 -----------------
    /** 进入航点任务 */
    val KeyEnter: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_ENTER_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 退出航点任务 */
    val KeyExit: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_EXIT_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 开始航点任务 */
    val KeyStart: AutelActionKeyInfo<MissionWaypointGUIDBean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_START_MSG,
        WaypointGUIDConverter(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 暂停航点任务 */
    val KeyPause: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_PAUSE_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 恢复航点任务 */
    val KeyResume: AutelActionKeyInfo<MissionWaypointGUIDBean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_CONTINUE_MSG,
        WaypointGUIDConverter(),
```

```
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 停止航点任务 */
    val KeyStop: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_STOP_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 查询航点任务断点信息 */
    val KeyBreakRequest: AutelActionKeyInfo<MissionWaypointGUIDBean,
MissionWaypointBreakRspBean> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_BREAK_REQUEST_MSG,
        WaypointGUIDConverter(),
        WaypointBreakRspConverter()
    ).canPerformAction(true)

    /** 航点状态信息上报 */
    val KeyStatusReportNtfy: AutelKeyInfo<MissionWaypointStatusReportNtfyBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_WAYPOINT_STATUS_REPORT_NTFY,
        WaypointStatusReportNtfyConverter()
    ).canListen(true)

    /** 进入兴趣点飞行任务 */
    val KeyIPMEnter: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_INTEREST_POINT_ENTER_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 退出兴趣点飞行任务 */
    val KeyIPExit: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_INTEREST_POINT_EXIT_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 开始兴趣点飞行任务 */
    val KeyIPMStart: AutelActionKeyInfo<MissionInterestPointStartMsgBean, Void>
= AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_INTEREST_POINT_START_MSG,
        MissionInterestPointStartMsgConverter(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 停止兴趣点飞行任务 */
    val KeyIPMStop: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_INTEREST_POINT_STOP_MSG,
        AutelEmptyConvert(),
```

```
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 兴趣点飞行状态信息上报 */
    val KeyIPMStatusReport:
AutelKeyInfo<MissionInterestPointStatusReportNtfyBean> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_INTEREST_POINT_STATUS_REPORT_NTFY,
        MissionInterestPointStatusReportNtfyConverter()
    ).canListen(true)

    /** 兴趣点图传打点 */
    val KeyIPMCreatePoint:
AutelActionKeyInfo<MissionInterestPointCreatePointMsgBean,
MissionInterestPointCreatePointRspBean> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_INTEREST_POINT_CREAT_POINT_MSG,
        MissionInterestPointCreatePointMsgConverter(),
        MissionInterestPointCreatePointRspConverter()
    ).canPerformAction(true)

    /** 任务一键急停 */
    val KeyMissionOneClickStop: AutelActionKeyInfo<Void, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.MISSION_ONE_CLICK_STOP,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

}
```

3.7.3 AIService模块

```
object AIServiceKey {

    private val component = ComponentType.AI_SERVICE

    /** 飞机端检测目标并将检测框发送至App */
    val KeySecurityDetectTarget: AutelKeyInfo<DetectTrackAreaNotifyBean> =
AutelKeyInfo(
        component.value,
        MessageTypeConstant.AISERVICE_DETECT_TARGET_RECT_NTFY,
        SecurityObjectNotifyConvert()
    ).canListen(true)

    /** 用户在App上选定目标后，飞机端锁定目标后将跟踪框发送至App */
    val KeyTrackingTargetRect: AutelKeyInfo<TrackAreaNotifyBean> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.AISERVICE_TRACKING_TARGET_RECT_NTFY,
        TrackAreaNotifyConvert()
    ).canListen(true)

    /** 目标检测类型设置 */
    val KeyTrackingTargetTypeSet: AutelActionKeyInfo<List<DetectTargetEnum>,
Void> = AutelActionKeyInfo(
        component.value,
```

```
        MessageTypeConstant.AISERVICE_TARGET_TYPE_SET_MSG,
        SecurityDetectTargetTypeSetConvert(), AutelEmptyConvert()
    ).canPerformAction(true)

}
```

3.7.4 相机模块

```
object CameraKey {
    private val component = ComponentType.CAMERA

    /** 相机设备信息 */
    val KeyCameraDeviceInfo: AutelKeyInfo<DeviceInfoBean> =
AutelKeyInfo(component.value, CameraKeyConstants.DeviceInfo,
DeviceInfoConvert())
        .canGet(true)

    /** 存储设备类型 */
    val KeyStorageType: AutelKeyInfo<StorageTypeEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.StorageType,
StorageTypeConvert())
        .canGet(true).canSet(true)

    /** SD卡状态 */
    val KeyStorageStatus: AutelKeyInfo<CardStatusBean> =
AutelKeyInfo(component.value, CameraKeyConstants.StorageStatus,
StorageStatusConvert())
        .canGet(true).canListen(true)

    /** MMC机载闪存状态 */
    val KeyMMCStatus: AutelKeyInfo<CardStatusBean> =
AutelKeyInfo(component.value, CameraKeyConstants.MMCStatus,
StorageStatusConvert())
        .canGet(true)

    /** 视频编码器配置 */
    val KeyVideoEncoderConfig: AutelKeyInfo<VideoEncoderConfigBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.VideoEncoderConfig,
VideoEncoderConfigConvert())
            .canGet(true).canSet(true)

    /** 视频源配置 */
    val KeyVideoSourceConfig: AutelKeyInfo<VideoSourceConfigBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.VideoSourceConfig,
VideoSourceConfigConvert())
            .canGet(true).canSet(true)

    /** 相机档位 */
    val KeyCameraGear: AutelKeyInfo<CameraGearEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.CameraGear,
CameraGearConvert())
        .canGet(true).canSet(true)

    /** 相机模式 */
    val KeyCameraMode: AutelKeyInfo<CameraWorkModeInfoBean> =
AutelKeyInfo(component.value, CameraKeyConstants.CameraMode,
CameraModeConvert())
```

```
        .canGet(true).canSet(true)

    /** 显示模式 */
    val KeyDisplayMode: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.DisplayMode,
AutelEmptyConvert(), AutelEmptyConvert()).canPerformAction(true)
            .canListen(true)

    /** 拍照参数 */
    val KeyTakePhotoParameters: AutelKeyInfo<TakePhotoParametersBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.TakePhotoParameters,
TakePhotoParamConvert())
            .canGet(true).canSet(true)

    /** 录像参数 */
    val KeyRecordParameters: AutelKeyInfo<RecordParametersBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.RecordParameters,
RecordParamConvert())
            .canGet(true).canSet(true)

    /** 测光点 */
    val KeyMeteringPoint: AutelKeyInfo<MeteringPointBean> =
AutelKeyInfo(component.value, CameraKeyConstants.MeteringPoint,
MeteringPointConvert())
        .canGet(true).canSet(true)

    /** 图像风格 */
    val KeyImageStyle: AutelKeyInfo<ImageStyleBean> =
AutelKeyInfo(component.value, CameraKeyConstants.ImageStyle,
ImageStyleConvert())
        .canGet(true).canSet(true)

    /** 白平衡参数 */
    val KeyWhiteBalance: AutelKeyInfo<WhiteBalanceBean> =
AutelKeyInfo(component.value, CameraKeyConstants.WhiteBalance,
WhiteBalanceConvert())
        .canGet(true).canSet(true)

    /** 图像颜色参数 */
    val KeyImageColor: AutelKeyInfo<ImageColorEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.ImageColor,
ImageColorConvert())
        .canGet(true).canSet(true)

    /** 图像曝光参数 */
    val KeyImageExposure: AutelKeyInfo<Double> = AutelKeyInfo(component.value,
CameraKeyConstants.ImageExposure, AutelDoubleConvert())
        .canGet(true).canSet(true)

    /** 图像感光度 */
    val KeyImageIso: AutelKeyInfo<Int> = AutelKeyInfo(component.value,
CameraKeyConstants.ImageIso, AutelIntConvert())
        .canGet(true).canSet(true)

    /** AELock */
    val KeyAELock: AutelKeyInfo<Boolean> = AutelKeyInfo(component.value,
CameraKeyConstants.AELock, AutelBooleanConvert())
        .canGet(true).canSet(true)
```

```
    /** 快门速度 */
    val KeyShutterSpeed: AutelKeyInfo<ShutterSpeedBean> =
AutelKeyInfo(component.value, CameraKeyConstants.ShutterSpeed,
ShutterSpeedConvert())
        .canGet(true).canSet(true)

    /** 快门模式 */
    val KeyShutterMode: AutelKeyInfo<ShutterModeEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.ShutterMode,
ShutterModeConvert())
        .canGet(true).canSet(true)

    /**  聚焦信息模式 */
    val KeyFocusInfoMode: AutelKeyInfo<FocusModeEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.FocusInfoMode,
FocusModeConvert())
        .canGet(true).canSet(true)

    /**  自动聚焦模式 */
    val KeyAFMeterMode: AutelKeyInfo<AFLensFocusModeEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.AFMeterMode,
AFMeterModeConvert())
        .canGet(true).canSet(true)

    /**  点聚焦坐标组 */
    val KeyCameraFocusSpotArea: AutelKeyInfo<MeteringPointBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.CameraFocusSpotArea,
MeteringPointConvert())
            .canGet(true).canSet(true)

    /**  手动变焦物距
     *   物距【0-50】
     */
    val KeyCameraMFObjectDistance: AutelKeyInfo<Int> =
AutelKeyInfo(component.value, CameraKeyConstants.CameraMFObjectDistance,
AutelIntConvert())
        .canGet(true).canSet(true)

    /**  AF辅助对焦使能 */
    val KeyCameraAFAssistFocusEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CameraAFAssistFocusEnable, AutelBooleanConvert())
            .canGet(true).canSet(true)

    /**  MF辅助对焦使能 */
    val KeyCameraMFAssistFocusEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CameraMFAssistFocusEnable, AutelBooleanConvert())
            .canGet(true).canSet(true)

    /** 光圈大小 */
    val KeyApertureSize: AutelKeyInfo<Double> = AutelKeyInfo(component.value,
CameraKeyConstants.ApertureSize, AutelDoubleConvert())
        .canGet(true).canSet(true)

    /** 光圈模式 */
```

```
    val KeyApertureMode: AutelKeyInfo<ApertureModeEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.ApertureMode,
ApertureModeConvert())
        .canGet(true).canSet(true)

    /** 数码/热成像变焦信息 */
    val KeyZoomFactor: AutelKeyInfo<Int> = AutelKeyInfo(component.value,
CameraKeyConstants.ZoomFactor, AutelIntConvert())
        .canGet(true).canSet(true)

    /** 相机界面模式 */
    val KeyPatternMode: AutelKeyInfo<PatternModeEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.PatternMode,
PatternModeConvert())
        .canGet(true).canSet(true)

    /** PIV录像状态 */
    val KeyRecordPiv: AutelKeyInfo<CameraRecordPivInfoBean> =
AutelKeyInfo(component.value, CameraKeyConstants.RecordPiv, RecordPivConvert())
        .canGet(true).canSet(true)

    /** HDR配置 */
    val KeyHDR: AutelKeyInfo<Boolean> = AutelKeyInfo(component.value,
CameraKeyConstants.HDR, AutelBooleanConvert())
        .canGet(true).canSet(true)

    /** 透雾功能配置 */
    val KeyDefog: AutelKeyInfo<DefogBean> = AutelKeyInfo(component.value,
CameraKeyConstants.Defog, DefogConvert())
        .canGet(true).canSet(true)

    /** ROI配置 */
    val KeyROI: AutelKeyInfo<ROIBean> = AutelKeyInfo(component.value,
CameraKeyConstants.ROI, ROIConvert())
        .canGet(true).canSet(true)

    /** 感光度模式 */
    val KeyISOMode: AutelKeyInfo<ISOModeEnum> = AutelKeyInfo(component.value,
CameraKeyConstants.ISOMode, ISOModeConvert())
        .canGet(true).canSet(true)

    /** 录像分段打包大小 */
    val KeyRecordPacket: AutelKeyInfo<RecordPacketBean> =
AutelKeyInfo(component.value, CameraKeyConstants.RecordPacket,
RecordPacketConvert())
        .canGet(true).canSet(true)

    /** 水印 */
    val KeyWatermark: AutelKeyInfo<WatermarkBean> =
AutelKeyInfo(component.value, CameraKeyConstants.Watermark, WatermarkConvert())
        .canGet(true).canSet(true)

    /** 热成像伪彩信息 */
    val KeyThermalColor: AutelKeyInfo<ThermalColorEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.ThermalColor,
ThermalColorConvert())
        .canGet(true).canSet(true)
```

```
    /** 热成像图像模式 */
    val KeyThermalMode: AutelKeyInfo<ThermalImageBean> =
AutelKeyInfo(component.value, CameraKeyConstants.ThermalMode,
ThermalImageConvert())
        .canGet(true).canSet(true)

    /** 热成像图像增强 */
    val KeyThermalEnhance: AutelKeyInfo<ThermalEnhanceBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.ThermalEnhance,
ThermalEnhanceConvert())
            .canGet(true).canSet(true)

    /** 热成像图像去噪 */
    val KeyThermalDenoising: AutelKeyInfo<Boolean> =
AutelKeyInfo(component.value, CameraKeyConstants.ThermalDenoising,
AutelBooleanConvert())
        .canGet(true).canSet(true)

    /** 热成像图像增益 */
    val KeyThermalGain: AutelKeyInfo<ThermalGainEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.ThermalGain,
ThermalGainConvert())
        .canGet(true).canSet(true)

    /** 热成像等温线 */
    val KeyThermalIsotherm: AutelKeyInfo<ThermalIsothermBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.ThermalIsotherm,
ThermalIsothermConvert())
            .canGet(true).canSet(true)

    /** 热成像温度属性 */
    val KeyThermalTemperature: AutelKeyInfo<ThermalTempAttrBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.ThermalTemperature,
ThermalTempAttrConvert())
            .canGet(true).canSet(true)

    /** 热成像温度告警属性 */
    val KeyThermalTemperatureAlarm: AutelKeyInfo<ThermalTempAlarmBean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.ThermalTemperatureAlarm, ThermalTempAlarmConvert())
            .canGet(true).canSet(true)

    /** 热成像辐射率 */
    val KeyThermalRadiance: AutelKeyInfo<Int> = AutelKeyInfo(component.value,
CameraKeyConstants.ThermalRadiance, AutelIntConvert())
        .canGet(true).canSet(true)

    /** 相机拍照分辨率信息 */
    val KeyTakePhotoResolution: AutelKeyInfo<PhotoResolutionEnum> =
        AutelKeyInfo(component.value, CameraKeyConstants.TakePhotoResolution,
TakePhotoResolutionConvert())
            .canGet(true).canSet(true)

    /** 相机录像分辨率信息 */
    val KeyRecordResolution: AutelKeyInfo<VideoResolutionBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.RecordResolution,
RecordResolutionConvert())
            .canGet(true).canSet(true)
```

```
    /** 相机拍照图片类型 */
    val KeyPhotoFileFormat: AutelKeyInfo<PhotoFormatEnum> =
        AutelKeyInfo(component.value, CameraKeyConstants.PhotoFileFormat,
PhotoFileFormatConvert())
            .canGet(true).canSet(true)

    /** 相机AEB拍照Count */
    val KeyTakePhotoAebCount: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.TakePhotoAebCount,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 相机Bust拍照Count */
    val KeyTakePhotoBustCount: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.TakePhotoBustCount,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 相机Timelapse拍照间隔 */
    val KeyTakePhotoTimeLapse: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.TakePhotoTimeLapse,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 录像文件类型 */
    val KeyRecordFileFormat: AutelKeyInfo<VideoFormatEnum> =
        AutelKeyInfo(component.value, CameraKeyConstants.RecordFileFormat,
VideoFileFormatConvert())
            .canGet(true).canSet(true)

    /** 录像文件编码类型 */
    val KeyRecordFileEncodeFormat: AutelKeyInfo<VideoCompressStandardEnum> =
        AutelKeyInfo(component.value, CameraKeyConstants.RecordFileEncodeFormat,
VideoEncodeFormatConvert())
            .canGet(true).canSet(true)

    /** 视频字幕开关 */
    val KeyCameraSubtitleKey: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value, CameraKeyConstants.CameraSubtitleKey,
AutelBooleanConvert())
            .canGet(true).canSet(true)

    /** 获取相机基本参数 */
    val KeyGetBaseParamMsg: AutelKeyInfo<CameraBaseParamBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.GetBaseParamMsg,
CameraBaseParamConvert())
            .canGet(true).canSet(true)

    /** 抗闪烁模式 */
    val KeyCameraAntiflicker: AutelKeyInfo<AntiflickerEnum> =
        AutelKeyInfo(component.value, CameraKeyConstants.CameraAntiflicker,
AntiflickerConvert())
            .canGet(true).canSet(true)

    /** 拍照录像时自动关闭机臂灯 */
    val KeyCameraTurnOffArmLight: AutelKeyInfo<Boolean> = AutelKeyInfo(
```

```
        component.value, CameraKeyConstants.CameraTurnOffArmLight,
AutelBooleanConvert()
    ).canGet(true).canSet(true)

    /** 联动变焦开关 */
    val KeyCameraLinkageZoom: AutelKeyInfo<Boolean> = AutelKeyInfo(
        component.value, CameraKeyConstants.CameraLinkageZoom,
AutelIntToBoolConverter()
    ).canGet(true).canSet(true)

    /** 丝滑变焦 */
    val KeySmoothZoom: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.SmoothZoom,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 视觉图传 */
    val KeyCameraVisualTransfer: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.CameraVisualTransfer,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 相机工作模式 */
    val KeyCameraWorkMode: AutelKeyInfo<CameraWorkModeEnum> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_WORK_MODE_NEW_MSG, CameraWorkModeConvert())
            .canGet(true).canSet(true)

    /** 相机拍照子模式 */
    val KeyCameraWorkModeTakePhoto: AutelKeyInfo<TakePhotoModeEnum> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_WORK_MODE_TAKE_PHOTO_MSG,
CameraWorkModeTakePhotoConvert())
            .canGet(true).canSet(true)

    /** 相机录像子模式 */
    val KeyCameraWorkModeVideo: AutelKeyInfo<RecordModeEnum> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_WORK_MODE_VIDEO_MSG, CameraWorkModeVideoConvert())
            .canGet(true).canSet(true)

    /** 图像风格类型 */
    val KeyCameraImageStyleType: AutelKeyInfo<ImageStyleEnum> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IMAGE_STYLE_TYPE_MSG, ImageStyleTypeConvert())
            .canGet(true).canSet(true)

    /** 图像风格亮度 */
    val KeyCameraImageStyleBrightness: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IMAGE_STYLE_BRIGHTNESS_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 图像风格对比度 */
    val KeyCameraImageStyleContrast: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IMAGE_STYLE_CONTRAST_MSG, AutelIntConvert())
            .canGet(true).canSet(true)
```

```
    /** 图像风格饱和度 */
    val KeyCameraImageStyleSaturation: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IMAGE_STYLE_SATURATION_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 图像风格色度 */
    val KeyCameraImageStyleHue: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IMAGE_STYLE_HUE_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 图像风格锐度 */
    val KeyCameraImageStyleSharpness: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IMAGE_STYLE_SHARPNESS_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 白平衡参数模式 */
    val KeyCameraWhiteBalanceType: AutelKeyInfo<WhiteBalanceEnum> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_WHITE_BALANCE_TYPE_MSG, WhiteBalanceTypeConvert())
            .canGet(true).canSet(true)

    /** 白平衡参数色温 */
    val KeyCameraWhiteBalanceColorTemp: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_WHITE_BALANCE_COLOR_TEMP_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** PIV录像状态是否开启 */
    val KeyCameraPivEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_PIV_ENABLE_MSG,
AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** PIV录像状态间隔时间 */
    val KeyCameraPivInterval: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_PIV_INTERVAL_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 透雾功能配置是否开启 */
    val KeyCameraDehazeEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_DEHAZE_ENABLE_MSG, AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 透雾功能配置 透雾强度 */
    val KeyCameraDehazeStrength: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_DEHAZE_STRENGTH_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 热成像图像模式类型 */
    val KeyCameraIrImageModeType: AutelKeyInfo<ThermalImageModeEnum> =
```

```
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_IMAGE_MODE_TYPE_MSG, IrImageModeConvert())
            .canGet(true).canSet(true)

    /** 热成像图像模式对比度 */
    val KeyCameraIrImageModeContrast: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_IMAGE_MODE_CONTRAST_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 热成像图像模式亮度 */
    val KeyCameraIrImageModeLum: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_IMAGE_MODE_LUM_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 热成像图像增强 使能 */
    val KeyCameraIrEnhanceEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_ENHANCE_ENABLE_MSG, AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 热成像图像增强 强度  */
    val KeyCameraIrEnhanceStrength: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_ENHANCE_STRENGTH_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 热成像温度属性 测温模式  */
    val KeyCameraIrTempAttrType: AutelKeyInfo<TemperatureModeEnum> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_TEMP_ATTR_TYPE_MSG, IrTempModeConvert())
            .canGet(true).canSet(true)

    /** 热成像温度属性 指点测温坐标  */
    val KeyCameraIrTempAttrTouch: AutelKeyInfo<ParameterPointBean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_TEMP_ATTR_TOUCH_MSG, ParameterPointConvert())
            .canGet(true).canSet(true)

    /** 热成像温度属性 区域测温  */
    val KeyCameraIrTempAttrRegion: AutelKeyInfo<ParameterRectBean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_TEMP_ATTR_REGION_MSG, ParameterRectConvert())
            .canGet(true).canSet(true)

    /** 热成像温度属性 限制测温 */
    val KeyCameraIrTempAttrLimitTemp: AutelKeyInfo<ParameterRectBean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_TEMP_ATTR_LIMITTEMP_MSG, ParameterRectConvert())
            .canGet(true).canSet(true)

    /** 热成像温度告警属性 使能 */
    val KeyCameraIrTempAlarmEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_TEMP_ALARM_ENABLE_MSG, AutelIntToBoolConverter())
            .canGet(true).canSet(true)
```

```
    /** 热成像温度告警属性 高温度阈值 */
    val KeyCameraIrTempAlarmHotthred: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_TEMP_ALARM_HOTTHRED_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 热成像温度告警属性 低温温度阈值 */
    val KeyCameraIrTempAlarmColdthred: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_IR_TEMP_ALARM_COLDTHRED_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 电子防抖开关 */
    val KeyCameraElectronicAntiShaking: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CameraElectronicAntiShaking, AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 任务文件夹名称 */
    val KeyCameraSaveMapTaskName: AutelKeyInfo<String> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_SAVE_MAP_TASK_NAME, AutelStringConvert())
            .canGet(true).canSet(true)

    /** 用户自定义文件夹名称 */
    val KeyCameraSaveMapUserDirName: AutelKeyInfo<String> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_SAVE_MAP_USER_DIR_NAME, AutelStringConvert())
            .canGet(true).canSet(true)

    /** 快速变焦 */
    val KeyCameraTypeXoomFixedFactor: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_TYPE_ZOOM_FIXED_FACTOR_MSG, AutelIntConvert())
            .canGet(true).canSet(true)

    /** 相机数据加密开关 */
    val KeyCameraTypeEncryptEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_TYPE_ENCRYPT_ENABLE, AutelIntToBoolConverter())
            .canGet(true)

    /** 相机自定义调试 */
    val KeyCameraDebugEvent: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_DEBUG_EVENT,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 视觉相机录像开关 */
    val KeyCameraRecordEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_RECORD_ENABLE,
AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 视觉相机录像帧率 */
    val KeyCameraRecordFps: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_RECORD_FPS,
AutelIntConvert())
```

```
    /** 视觉相机录像时长 */
    val KeyCameraRecordDuration: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_RECORD_DURATION,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 视觉相机录像保存数量 */
    val KeyCameraRecordNumber: AutelKeyInfo<Int> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_RECORD_NUMBER,
AutelIntConvert())
            .canGet(true).canSet(true)

    /** 相机能力集版本号获取*/
    val KeyCameraCapabilityVersion: AutelKeyInfo<String> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_CAPABILITY_VERSION, AutelStringConvert())
            .canGet(true).canSet(false)

    /** 相机开关 false 关，true开 */
    val KeyCameraEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_ENABLE,
AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 相机水印GPS开关 false 关，true开 */
    val KeyCameraWatermarkGpsEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_WATERMARK_GPS_ENABLE, AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 相机水印SN开关 false 关，true开 */
    val KeyCameraWaterMarkSnEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_WATERMARK_SN_ENABLE, AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 相机视觉开关 false 关，true开 */
    val KeyCameraVisualEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value, CameraKeyConstants.CAMERA_VISUAL_ENABLE,
AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** 拍照录像超感光开关 false 关，true开 */
    val KeyCameraUltraPixelEnable: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_ULTRA_PIXEL_ENABLE, AutelIntToBoolConverter())
            .canGet(true).canSet(true)

    /** ================================ 相机命令
====================================*/
    /** 重启相机 */
    val KeyCameraReboot: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.CameraReboot,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** SD卡格式化 */
```

```
    val KeyFormatSDCard: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.FormatSDCard,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** MMC机载闪存格式化 */
    val KeyFormatMMC: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.FormatMMC,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** 恢复出厂设置 */
    val KeyCameraReset: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.CameraReset,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** 开始拍照 */
    val KeyStartTakePhoto: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.StartTakePhoto,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** 停止拍照 */
    val KeyStopTakePhoto: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.StopTakePhoto,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** 开始录像 */
    val KeyStartRecord: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.StartRecord,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** 停止录像 */
    val KeyStopRecord: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.StopRecord,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** FFC快门 */
    val KeyCameraFfc: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(component.value, CameraKeyConstants.CAMERA_FFC_MSG,
AutelEmptyConvert(), AutelEmptyConvert())
            .canPerformAction(true)

    /** 视频/拍照存储类型获取 */
    val KeyCameraVideoPictureStorageTypeGet:
AutelActionKeyInfo<CameraVideoPhotoStorageBean, CameraVideoPhotoStorageListBean>
=
        AutelActionKeyInfo(
            component.value,
            CameraKeyConstants.CAMERA_VIDEO_PICTURE_STORAGE_TYPE_GET_MSG,
            CameraVideoPhotoStorageGetConvert(),
            CameraVideoPhotoStorageSetConvert()
        ).canPerformAction(true)
```

```
    /** 视频/拍照存储类型设置 */
    val KeyCameraVideoPictureStorageTypeSet:
AutelActionKeyInfo<CameraVideoPhotoStorageListBean, Void> =
        AutelActionKeyInfo(
            component.value,
            CameraKeyConstants.CAMERA_VIDEO_PICTURE_STORAGE_TYPE_SET_MSG,
            CameraVideoPhotoStorageSetConvert(),
            AutelEmptyConvert()
        ).canPerformAction(true)

    /** 相机数据加密设置 */
    val KeyCameraTypeEncryptionKey: AutelActionKeyInfo<CameraEncryptSetBean,
Void> =
        AutelActionKeyInfo(component.value,
CameraKeyConstants.CAMERA_TYPE_ENCRYPT_SET, CameraEncryptSetConvert(),
AutelEmptyConvert())
            .canPerformAction(true)

    /** ================================ 相机消息上报
================================ */
    /** 相机状态消息上报 */
    val KeyCameraStatus: AutelKeyInfo<CameraStatusBean> =
AutelKeyInfo(component.value, CameraKeyConstants.CameraStatus,
CameraStatusConvert())
        .canListen(true)

    /** 专业参数信息上报 */
    val KeyProfessionalParamInfo: AutelKeyInfo<ProfessionalParamInfoBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.ProfessionalParamInfo,
ProfessionalParamInfoBeanConvert()).canListen(true)

    /** 拍照文件信息上报 */
    val KeyPhotoFileInfo: AutelKeyInfo<PhotoFileInfoBean> =
AutelKeyInfo(component.value, CameraKeyConstants.PhotoFileInfo,
PhotoFileInfoConvert())
        .canListen(true).canSet(true)

    /** 拍照状态上报 */
    val KeyTakePhotoStatus: AutelKeyInfo<TakePhotoStatusBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.TakePhotoStatus,
TakePhotoStatusConvert())
            .canListen(true)

    /** 录像状态上报 */
    val KeyRecordStatus: AutelKeyInfo<RecordStatusBean> =
AutelKeyInfo(component.value, CameraKeyConstants.RecordStatus,
RecordStatusConvert())
        .canListen(true)

    /** 录像文件信息上报 */
    val KeyRecordFileInfo: AutelKeyInfo<RecordFileInfoBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.RecordFileInfo,
RecordFileInfoConvert())
            .canListen(true)

    /** SD卡状态上报 */
```

```
    val KeySDCardStatus: AutelKeyInfo<CardStatusBean> =
AutelKeyInfo(component.value, CameraKeyConstants.SDCardStatus,
StorageStatusConvert())
        .canListen(true)

    /** MMC状态上报 */
    val KeyMMCStatusInfo: AutelKeyInfo<CardStatusBean> =
AutelKeyInfo(component.value, CameraKeyConstants.MMCStatusInfo,
StorageStatusConvert())
        .canListen(true)

    /** 定时拍倒计时上报 */
    val KeyIntervalShotStatus: AutelKeyInfo<Int> = AutelKeyInfo(component.value,
CameraKeyConstants.IntervalShotStatus, AutelIntConvert())
        .canListen(true)

    /** 全景状态上报 */
    val KeyPanoramaStatus: AutelKeyInfo<PanoramaStatusBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.PanoramaStatus,
PanoramaStatusConvert())
            .canListen(true)

    /** 延时摄影状态上报 */
    val KeyDelayShotStatus: AutelKeyInfo<DelayShotStatusBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.DelayShotStatus,
DelayShotStatusConvert())
            .canListen(true)

    /** 重置相机状态上报 */
    val KeyResetCameraState: AutelKeyInfo<Boolean> =
AutelKeyInfo(component.value, CameraKeyConstants.ResetCameraState,
AutelBooleanConvert())
        .canListen(true)

    /** AE/AF状态改变 */
    val KeyAeAfStatusChange: AutelKeyInfo<CameraAFAEStatusBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.AeAfStatusChange,
CameraAFAEStatusConvert())
            .canListen(true)

    /** 温度报警事件 */
    val KeyTempAlarm: AutelKeyInfo<TempAlarmBean> =
AutelKeyInfo(component.value, CameraKeyConstants.TempAlarm, TempAlarmConvert())
        .canListen(true)

    /** 移动延时摄影状态 */
    val KeyMotionDelayShotStatus: AutelKeyInfo<MotionDelayShotBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.MotionDelayShotStatus,
MotionDelayShotStatusConvert()).canListen(true)

    /** 相机曝光状态 */
    val KeyPhotoExposure: AutelKeyInfo<ExposureEnum> =
AutelKeyInfo(component.value, CameraKeyConstants.PhotoExposure,
PhotoExposureConvert())
        .canListen(true)

    /** 任务录制航点信息上报 */
    val KeyMissionRecordWaypoint: AutelKeyInfo<MissionRecordWaypointBean> =
```

```
        AutelKeyInfo(component.value, CameraKeyConstants.MissionRecordWaypoint,
MissionRecordWaypointConvert()).canListen(true)

    /** 相机对焦状态 */
    val KeyAFState: AutelKeyInfo<AFStateEnum> = AutelKeyInfo(component.value,
CameraKeyConstants.AFState, AFStateConvert())
        .canListen(true)

    /** 相机模式切换上报 */
    val KeyCameraModeSwitch: AutelKeyInfo<CameraModeSwitchBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.CameraModeSwitch,
CameraModeSwitchConvert())
            .canListen(true)

    /** 显示模式切换上报 */
    val KeyDisplayModeSwitch: AutelKeyInfo<DisplayModeEnum> =
        AutelKeyInfo(component.value, CameraKeyConstants.DisplayModeSwitch,
DisplayModeConvert())
            .canListen(true)

    /** 红外相机温度信息上报 */
    val KeyInfraredCameraTempInfo: AutelKeyInfo<InfraredCameraTempInfoBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.InfraredCameraTempInfo,
InfraredCameraTempInfoConvert()).canListen(true)


    /** 存储状态信息上报 */
    val KeyStorageStatusInfo: AutelKeyInfo<StorageStatusInfoBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.StorageStatusInfo,
StorageStatusInfoBeanConvert())
            .canListen(true)

    /** 点测光上报 */
    val KeyLocationMeterInfo: AutelKeyInfo<MeteringPointBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.LocationMeterInfo,
MeteringPointConvert())
            .canListen(true)

    /** 白平衡参数上报 */
    val KeyWhiteBalanceNtfy: AutelKeyInfo<WhiteBalanceBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.WhiteBalanceNtfy,
WhiteBalanceConvert())
            .canListen(true)

    /** AELock信息上报 */
    val KeyAeLockNtfy: AutelKeyInfo<Boolean> =
        AutelKeyInfo(component.value, CameraKeyConstants.AeLockNtfy,
AutelBooleanConvert())
            .canListen(true)

    /** 聚焦信息上报 */
    val KeyFocusNtfy: AutelKeyInfo<FocusInfoBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.FocusNtfy,
FocusInfoConvert())
            .canListen(true)

    /** HDR信息上报 */
    val KeyHdrNtfy: AutelKeyInfo<Boolean> =
```

```
        AutelKeyInfo(component.value, CameraKeyConstants.HdrNtfy,
AutelBooleanConvert())
            .canListen(true)

    /** ROI配置上报 */
    val KeyRoiNtfy: AutelKeyInfo<ROIBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.RoiNtfy, ROIConvert())
            .canListen(true)

    /** 相机拍照分辨率信息上报 */
    val KeyPhotoResolutionNtfy: AutelKeyInfo<PhotoResolutionEnum> =
        AutelKeyInfo(component.value, CameraKeyConstants.PhotoResolutionNtfy,
TakePhotoResolutionConvert())
            .canListen(true)

    /** 相机录像分辨率信息上报 */
    val KeyVideoResolutionNtfy: AutelKeyInfo<VideoResolutionBean> =
        AutelKeyInfo(component.value, CameraKeyConstants.VideoResolutionNtfy,
RecordResolutionConvert())
            .canListen(true)

    /** 解密进度上报 */
    val KeyCameraEncryptProgressReportNtfy:
AutelKeyInfo<CameraEncryptProgressReportBean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CameraEncryptProgressReportNtfy,
CameraEncryptProgressReportConvert())
            .canListen(true)

    /** 相机图传信息上报 */
    val KeyCameraTransferInfoNtfy: AutelKeyInfo<CameraTransferInfoBean> =
        AutelKeyInfo(component.value,
CameraKeyConstants.CAMERA_TRANSFER_INFO_NTFY, CameraTransferInfoConvert())
            .canListen(true)

}
```

3.7.5 飞行控制模块

```
object FlightControlKey {

    private val component = ComponentType.FLIGHT_CONTROLLER

    /** 指南针校准 */
    val KeyCalibrateCompass: AutelActionKeyInfo<Void, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_CALIBRATE_COMPASS_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 起飞 */
    val KeyTakeOffAirCraft: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_TAKEOFF_AIRCRAFT_MSG,
        AutelEmptyConvert(),
```

```
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** false:取消自动降落，true:自动降落 */
    val KeySetLanding: AutelActionKeyInfo<Boolean, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_SET_LANDING_MSG,
        AutelBooleanConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 启停电机 */
    val KeyStartStopMotor: AutelActionKeyInfo<Boolean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_START_STOP_MOTOR_MSG,
        AutelBooleanConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** false:取消自动返航，true:自动返航 */
    val KeyStartStopAutoBack: AutelActionKeyInfo<Boolean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_START_STOP_AUTOBACK_MSG,
        AutelBooleanConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** Home返航点设置 */
    val KeySetHomeLocation: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_SET_HOMELOCATION_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 自定义返航点设置 */
    val KeyCustomHomeLocation: AutelActionKeyInfo<HomeLocation, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_CUSTOM_HOMELOCATION_MSG,
        CustomHomeLocationConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 检查是否需要上传禁飞区文件 */
    val KeyCheckNFZUpload: AutelActionKeyInfo<NoFlyQzoneBean, Boolean> =
        AutelActionKeyInfo(
            component.value,
            MessageTypeConstant.FCS_CHECK_NFZ_UPLOAD_MSG,
            CheckNFZUploadConvert(),
            AutelBooleanConvert()
        ).canPerformAction(true)

    /** 禁飞区使能 */
    val KeyEnableNFZ: AutelActionKeyInfo<Boolean, Void> = AutelActionKeyInfo(
        component.value,
```

```
        MessageTypeConstant.FCS_ENABLE_NFZ_MSG,
        AutelBooleanConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 姿态模式允许起飞 */
    val KeySetAttiTakeOff: AutelActionKeyInfo<Boolean, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_SET_ATTI_TAKEOFF_MSG,
        AutelBooleanConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** IMU校准 */
    val KeyCalibrateIMU: AutelActionKeyInfo<Void, Void> = AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_CALIBRATE_IMU_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 获取RTK授权信息 */
    val KeyGetRTKAuthInfo: AutelActionKeyInfo<Void, RTKAuthInfo> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_GET_RTK_AUTH_INFO_MSG,
        AutelEmptyConvert(),
        RTKAuthInfoConvert()
    ).canPerformAction(true)

    /** 设置RTK授权信息 */
    val KeySetRTKAuthInfo: AutelActionKeyInfo<RTKAuthInfo, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_SET_RTK_AUTH_INFO_MSG,
        RTKAuthInfoConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 获取当前任务GUID */
    val KeyGetMissionGuid: AutelActionKeyInfo<Void, String> =
        AutelActionKeyInfo(
            component.value,
            MessageTypeConstant.FCS_GET_MISSION_GUID_MSG,
            AutelEmptyConvert(),
            AutelStringConvert()
        ).canPerformAction(true)

    /** 取消低电返航 */
    val KeyCancelLowPowerBack: AutelActionKeyInfo<Void, Void> =
AutelActionKeyInfo(
        component.value,
        MessageTypeConstant.FCS_CANCEL_LOWPOWER_BACK_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)
```

```
    /** 设置磁力计异常允许起飞 */
    val KeySetCompassTakeOff: AutelActionKeyInfo<Boolean, Void> =
        AutelActionKeyInfo(
            component.value,
            MessageTypeConstant.FCS_SET_COMPASS_TAKEOFF_MSG,
            AutelBooleanConvert(),
            AutelEmptyConvert()
        ).canPerformAction(true)

    /** 设置第4轴云台竖屏模式 */
    val KeySetPortraitMode: AutelActionKeyInfo<Boolean, Void> =
        AutelActionKeyInfo(
            component.value,
            MessageTypeConstant.FCS_SET_PORTRAIT_MODE_MSG,
            AutelBooleanConvert(),
            AutelEmptyConvert()
        ).canPerformAction(true)

    /** 获取飞控参数集合 */
    val KeyGetCommonParams: AutelActionKeyInfo<Void, DroneCommonParamSetBean> =
        AutelActionKeyInfo(
            component.value,
            MessageTypeConstant.FCS_GET_COMMON_PARAMS_MSG,
            AutelEmptyConvert(),
            DroneCommonParamSetConverter()
        ).canPerformAction(true)

    /**收桨控制*/
    val KeyNestRetractPaddleControl: AutelActionKeyInfo<Void, NestWaitTimeBean>?
= AutelActionKeyInfo(
        component.value,
        NestConstant.NestRetractPaddleControl,
        AutelEmptyConvert(),
        NestWaitTimeConvert()
    ).canPerformAction(true)
}
```

3.7.6 飞行参数读取和设置模块

```
object FlightPropertyKey {
    private val component = ComponentType.FLIGHT_PARAMS

    /** 新手模式 */
    val KeyBeginMode: AutelKeyInfo<OperatorModeEnum> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_BEGIN_MODE_MSG,
            FlightParamBeginModeConvert()
        ).canGet(true).canSet(true)

    /** 最大飞行高度设置/获取 */
    val KeyMaxHeight: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value, MessageTypeConstant.FCS_MAX_HEIGHT_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)
```

```
    /** 最大飞行半径设置/获取 */
    val KeyMaxRadius: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value, MessageTypeConstant.FCS_MAX_RADIUS_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 最大水平飞行速度 */
    val KeyMaxHorizontalSpeed: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_MAX_HORIZONTAL_SPEED_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 最大上升飞行速度 */
    val KeyMaxAscentSpeed: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_MAX_ASCENT_SPEED_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 最大下降飞行速度 */
    val KeyDescentSpeed: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_MAX_DESCENT_SPEED_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 最大下降飞行速度 */
    val KeyMissionManagerBackHeight: AutelKeyInfo<Float> = AutelKeyInfo(
        component.value,
        MessageTypeConstant.MISSIONMANAGER_BACK_HEIGHT_MSG,
        AutelFloatConvert()
    ).canGet(true).canSet(true)

    /** 蜂鸣状态（查找飞机） */
    val KeyBuzzingStatus: AutelKeyInfo<Boolean> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_BUZZING_STATUS_MSG,
            AutelIntToBoolConverter()
        ).canGet(true).canSet(true)

    /** ATTI模式 */
    val KeyAttiMode: AutelKeyInfo<Boolean> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_ATTI_MODE_MSG,
            AutelIntToBoolConverter()
        ).canGet(true).canSet(true)

    /** EXP左转右转设置/获取 */
    val KeyYawAngleSensitivity: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
```

```
            MessageTypeConstant.FCS_YAW_ANGLE_SENSITIVITY_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** EXP向前向后设置/获取 */
    val KeyPitchSensitivity: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_PITCH_SENSITIVITY_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** EXP向左向右设置获取 */
    val KeyRollSensitivity: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_ROLL_SENSITIVITY_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** EXP上升下降设置/获取 */
    val KeyThrustSensitivity: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_THRUST_SENSITIVITY_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 灵敏度姿态设置/获取 */
    val KeyAttitudeSensitivity: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_ATTITUDE_SENSITIVITY_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 灵敏度刹车设置/获取 */
    val KeyBrakeSensitivity: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_BRAKE_SENSITIVITY_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 灵敏度偏航行程设置/获取 */
    val KeyYawTripSensitivity: AutelKeyInfo<Float> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.FCS_YAW_TRIP_SENSITIVITY_MSG,
            AutelFloatConvert()
        ).canGet(true).canSet(true)

    /** 电池低电量告警 */
    val KeyBatteryLowWarning: AutelKeyInfo<Int> =
        AutelKeyInfo(
            component.value,
            MessageTypeConstant.MISSIONMANAGER_BATTERY_LOW_WARNING_MSG,
            AutelIntConvert()
```

```
    ).canGet(true).canSet(true)

/** 7.18 电池严重低电量告警 */
val KeyBatSeriousLowWarning: AutelKeyInfo<Int> =
    AutelKeyInfo(
        component.value,
        MessageTypeConstant.MISSIONMANAGER_BAT_SERIOUS_LOW_WARNING_MSG,
        AutelIntConvert()
    ).canGet(true).canSet(true)

/** 低电量返航 */
val KeyLowBatLowBack: AutelKeyInfo<Boolean> =
    AutelKeyInfo(
        component.value,
        MessageTypeConstant.MISSIONMANAGER_LOW_BAT_LOW_BACK_MSG,
        AutelBooleanConvert()
    ).canGet(true).canSet(true)


/** RTK定位 */
val KeyRTKLocationEnable: AutelKeyInfo<Boolean> =
    AutelKeyInfo(
        component.value,
        MessageTypeConstant.FCS_RTK_LOCATION_ENABLE_MSG,
        AutelBooleanConvert()
    ).canGet(true).canSet(true)

/** RTK信号方式 */
val KeyRTKSignalType: AutelKeyInfo<Int> =
    AutelKeyInfo(
        component.value,
        MessageTypeConstant.FCS_RTK_SIGNAl_TYPE_MSG,
        AutelIntConvert()
    ).canGet(true).canSet(true)

/** RTK坐标系 */
val KeyRTKCoordinateSystem: AutelKeyInfo<Int> =
    AutelKeyInfo(
        component.value,
        MessageTypeConstant.FCS_RTK_COORDINATE_SYSTEM_MSG,
        AutelIntConvert()
    ).canGet(true).canSet(true)

/** 飞行器激活状态 */
val KeyAircraftActivation: AutelKeyInfo<Boolean> =
    AutelKeyInfo(
        component.value,
        MessageTypeConstant.FCS_AIRCRAFT_ACTIVATION_MSG,
        AutelIntToBoolConverter()
    ).canGet(true).canSet(true)

/** 飞机失联行为 */
val KeyRCLostAction: AutelKeyInfo<DroneLostActionEnum> =
    AutelKeyInfo(
        component.value, MessageTypeConstant.FCS_RC_LOST_ACTION_MSG,
        DroneLostActionConverter()
    ).canGet(true).canSet(true)
```

```
/** 飞行器档位设置/获取 */
val KeyGearLever: AutelKeyInfo<GearLevelEnum> =
    AutelKeyInfo(
        component.value, MessageTypeConstant.FCS_GEAR_LEVEL_MSG,
        GearLevelConverter()
    ).canGet(true).canSet(true)

/** 协调转弯 */
val KeyCoordinatedTurn: AutelKeyInfo<Boolean> =
    AutelKeyInfo(
        component.value,
        MessageTypeConstant.FCS_COORDINATED_TURN_MSG,
        AutelBooleanConvert()
    ).canGet(true).canSet(true)

/** 融合视觉定位开关 */
val KeyLocationStatus: AutelKeyInfo<Boolean> = AutelKeyInfo(
    component.value,
    MessageTypeConstant.FCS_VISION_LOCATION_STATUS_MSG,
    AutelIntToBoolConverter()
).canSet(true).canGet(true)

/** 绕障功能开关 */
val KeyFcsApasModeEn: AutelKeyInfo<Boolean> = AutelKeyInfo(
    component.value,
    MessageTypeConstant.FCS_APAS_MODE_EN,
    AutelIntToBoolConverter()
).canSet(true).canGet(true)

/** 静默模式状态开关 */
val KeySilentModeStatus: AutelKeyInfo<Boolean> = AutelKeyInfo(
    component.value,
    MessageTypeConstant.FCS_SILENT_MODE_STATUS_MSG,
    AutelIntToBoolConverter(),
).canSet(true).canGet(true)

/** 超级电容开关状态 */
val KeyFCSEnSuperCap: AutelKeyInfo<Boolean> = AutelKeyInfo(
    component.value,
    MessageTypeConstant.FCS_EN_SUPER_CAP_MSG,
    AutelIntToBoolConverter(),
).canSet(true).canGet(true)

/** GPS飞行开关 */
val KeyFcsEnGpsMode: AutelKeyInfo<Boolean> = AutelKeyInfo(
    component.value,
    MessageTypeConstant.FCS_EN_GPS_MODE,
    AutelIntToBoolConverter(),
).canSet(true).canGet(true)

/** GPS工作模式 */
val KeyFcsSwitchGpsMode: AutelKeyInfo<DroneGpsEnum> = AutelKeyInfo(
    component.value,
    MessageTypeConstant.FCS_SWITCH_GPS_MODE,
    DroneGpsConverter(),
).canSet(true).canGet(true)
```

```
}
```

3.7.7 云台模块

```
object GimbalKey {

    private val componentVal = ComponentType.GIMBAL.value

    /** 云台上报参数 */
    val KeyHeatBeat: AutelKeyInfo<DroneGimbalStateBean> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_HEART_BEAT_MSG,
        GimbalHeartBeatConvert()
    ).canListen(true)

    /** 云台ROLL轴微调角度 */
    val KeyRollAdjustAngle: AutelKeyInfo<Int> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_ROLL_ADJUST_ANGLE_MSG,
        AutelIntConvert(),
    ).canSet(true).canGet(true)

    /** 云台PITCH轴微调角度 */
    val KeyPitchAngleRange: AutelKeyInfo<Int> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_PITCH_ADJUST_ANGLE_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)

    /** 云台YAW轴微调角度 */
    val KeyYawAdjustAngle: AutelKeyInfo<Int> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_YAW_ADJUST_ANGLE_MSG,
        AutelIntConvert(),
    ).canSet(true).canGet(true)

    /** 开启云台IMU校准 */
    val KeyStartIMUCalibration: AutelActionKeyInfo<Void, Void> =
AutelActionKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_START_IMU_CALIBRATION_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 开启云台校准 */
    val KeyStartCalibration: AutelActionKeyInfo<Void, Void> =
AutelActionKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_START_CALIBRATION_MSG,
        AutelEmptyConvert(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 旋转四轴云台 */
    val KeyRotateFouraxisAngle: AutelActionKeyInfo<RotateFourAxisParamsBean,
Void> = AutelActionKeyInfo(
```

```
        componentVal,
        MessageTypeConstant.GIMBAL_ROTATE_FOURAXIS_ANGLE_MSG,
        RotateFourAxisParamsConverter(),
        AutelEmptyConvert()
    ).canPerformAction(true)

    /** 云台工作模式 */
    val KeyWordMode: AutelKeyInfo<Int> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_WORK_MODE_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)

    /** 云台俯仰限位开关 */
    val KeyPitchAngelRange: AutelKeyInfo<Boolean> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_PITCH_ANGLE_RANGE_MSG,
        AutelIntToBoolConverter()
    ).canSet(true).canGet(true)

    /** 云台俯仰速度 */
    val KeyPitchSpeed: AutelKeyInfo<Int> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_PITCH_SPEED_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)

    /** 云台角度控制 */
    val KeyAngleControl: AutelActionKeyInfo<Float, Void> = AutelActionKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_ANGLE_CONTROL_MSG,
        AutelFloatConvert(), AutelEmptyConvert()
    ).canPerformAction(true)

    /** 云台方向控制（朝下控制，云台归中） */
    val KeyOrientationControl: AutelActionKeyInfo<GimbalOrientationEnum, Void> =
AutelActionKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_ORIENTATION_CONTROL_MSG,
        GimbalOrientationConverter(), AutelEmptyConvert()
    ).canPerformAction(true)

    /** 云台激光测距开关 true 开 false 关*/
    val KeyLaserRangingSwitch: AutelKeyInfo<Boolean> = AutelKeyInfo(
        componentVal,
        MessageTypeConstant.GIMBAL_LASER_RANGING_SWITCH,
        AutelIntToBoolConverter(),
    ).canSet(true).canGet(true)

}
```

3.7.8 视觉模块

```
object VisionKey {

    val component = ComponentType.VISION
```

```
    /** 视觉雷达图告警 */
    val KeyReportEmergency: AutelKeyInfo<List<VisionRadarInfoBean>> =
AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_REPORT_EMERGENCY_MSG,
        VisionWarningConvert()
    ).canListen(true)

    /** 水平避障开关 */
    val KeyHorizontalObstacleAvoidance: AutelKeyInfo<Boolean> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_HORIZONTAL_OBSTACLE_AVOIDANCE_MSG,
        AutelIntToBoolConverter()
    ).canSet(true).canGet(true)

    /** 水平避障刹车距离 */
    val KeyHorizontalBrakeDistance: AutelKeyInfo<Float> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_HORIZONTAL_BRAKE_DISTANCE_MSG,
        AutelFloatConvert()
    ).canSet(true).canGet(true)

    /** 水平告警距离 */
    val KeyHorizontalWarningDistance: AutelKeyInfo<Float> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_HORIZONTAL_WARNING_DISTANCE_MSG,
        AutelFloatConvert()
    ).canSet(true).canGet(true)

    /** 上方避障开关 */
    val KeyTopObstacleAvoidance: AutelKeyInfo<Boolean> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_TOP_OBSTACLE_AVOIDANCE_MSG,
        AutelIntToBoolConverter()
    ).canSet(true).canGet(true)

    /** 上方避障刹车距离 */
    val KeyTopBrakeDistance: AutelKeyInfo<Float> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_TOP_BRAKE_DISTANCE_MSG,
        AutelFloatConvert()
    ).canSet(true).canGet(true)

    /** 上方告警距离 */
    val KeyTopWarningDistance: AutelKeyInfo<Float> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_TOP_WARNING_DISTANCE_MSG,
        AutelFloatConvert()
    ).canSet(true).canGet(true)

    /** 下方避障开关 */
    val KeyBottomObstacleAvoidance: AutelKeyInfo<Boolean> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_BOTTOM_OBSTACLE_AVOIDANCE_MSG,
        AutelIntToBoolConverter()
    ).canSet(true).canGet(true)

    /** 下方避障刹车距离 */
```

```
    val KeyBottomBrakeDistance: AutelKeyInfo<Float> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_BOTTOM_BRAKE_DISTANCE_MSG,
        AutelFloatConvert()
    ).canSet(true).canGet(true)

    /** 下方告警距离 */
    val KeyBottomWarningDistance: AutelKeyInfo<Float> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_BOTTOM_WARNING_DISTANCE_MSG,
        AutelFloatConvert()
    ).canSet(true).canGet(true)

    /** 雷达开关 */
    val KeyRadarDetection: AutelKeyInfo<Boolean> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_RADAR_DETECTION_MSG,
        AutelBooleanConvert()
    ).canSet(true).canGet(true)

    /** MIF视觉定位工作状态 */
    val KeyAutonomyMifWorkStatus: AutelKeyInfo<Boolean> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.AUTONOMY_MIF_WORK_STATUS_MSG,
        AutelIntToBoolConverter()
    ).canSet(true).canGet(true)

    /** 提供给视觉测试使用 */
    val VISION_PERCEPTION_DATA_HEAD_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_HEAD_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_REAR_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_REAR_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_BOTTOM_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_BOTTOM_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_RIGHT_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_RIGHT_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_LEFT_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_LEFT_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_TOP_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_TOP_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
```

```
    val VISION_PERCEPTION_DATA_HEAD_KEY_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_HEAD_KEY_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_REAR_KEY_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_REAR_KEY_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_BOTTOM_KEY_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_BOTTOM_KEY_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_RIGHT_KEY_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_RIGHT_KEY_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_LEFT_KEY_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_LEFT_KEY_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DATA_TOP_KEY_MSG: AutelKeyInfo<Int> = AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DATA_TOP_KEY_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)
    val VISION_PERCEPTION_DISTORTED_IMAGE_KEY_MSG: AutelKeyInfo<Int> =
AutelKeyInfo(
        FlightMissionKey.component.value,
        MessageTypeConstant.VISION_PERCEPTION_DISTORTED_IMAGE_KEY_MSG,
        AutelIntConvert()
    ).canSet(true).canGet(true)

}
```

3.7.9 Airlink 图传模块

```
object AirLinkKey {
    private val component = ComponentType.ALINK

    /** 图传频段读写 */
    val KeyALinkBandMode: AutelKeyInfo<AirLinkBandModeEnum> = AutelKeyInfo(
        component.value,
        ALinkConstant.AirLinkBandMode,
        AirLinkBandModeConvert()
    ).canGet(true).canSet(true)

    /** 图传清晰度模式 */
    val KeyALinkTransmissionMode: AutelKeyInfo<VideoTransMissionModeEnum> =
        AutelKeyInfo(
            component.value,
            ALinkConstant.AirLinkTransmissionMode,
            AirLinkTransmissionModeConvert()
```

```
        ).canGet(true).canSet(true)

    /** 辐射功率模式 */
    val KeyALinkFccCeMode: AutelKeyInfo<FccCeModeEnum> =
        AutelKeyInfo(
            component.value,
            ALinkConstant.AIRLINK_FCC_CE_MODE_MSG,
            AirLinkFccCeModeConvert()
        ).canGet(true).canSet(true)

    /** 开始对频 */
    val KeyALinkStartMatching: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(
            ComponentType.REMOTE_CONTROLLER.value,
            ALinkConstant.AirLinkStartMatching,
            AutelEmptyConvert(),
            AutelEmptyConvert()
        ).canPerformAction(true)

    /** 对频进度上报事件 */
    val KeyALinkMatchingStatus: AutelKeyInfo<AirLinkMatchStatusEnum> =
        AutelKeyInfo(
            ComponentType.REMOTE_CONTROLLER.value,
            ALinkConstant.AirLinkMatchingStatus,
            AirLinkMatchingStatusConvert()
        ).canListen(true)

    /** 图传对频时长上报 会上报两次，以短时间为准 调试工具使用 */
    val KeyALinkMatchCostTime: AutelKeyInfo<AirlinkMatchCostTimeBean> =
        AutelKeyInfo(
            ComponentType.REMOTE_CONTROLLER.value,
            ALinkConstant.AIRLINK_MATCH_COST_TIME_NTFY,
            AirLinkMatchCostTimeConvert()
        ).canListen(true)

    /** 图传信号强度上报 调试工具使用 */
    val KeyALinkSignalStrength: AutelKeyInfo<AirLinkSignalStrengthBean> =
        AutelKeyInfo(
            ComponentType.REMOTE_CONTROLLER.value,
            ALinkConstant.AIRLINK_SIGNAL_STRENGTH_NTFY,
            AirLinkSignalStrengthConvert()
        ).canListen(true)

    /** 重置对频标志 */
    val KeyAirlinkResetMatchFlag: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(
            component.value,
            ALinkConstant.AirlinkResetMatchFlag,
            AutelEmptyConvert(),
            AutelEmptyConvert()
        ).canPerformAction(true)

    /** 用户单击电池确认连接通知 */
    val KeyALinkConnectConfirm: AutelKeyInfo<Void> =
        AutelKeyInfo(
            component.value,
            ALinkConstant.ALRLINK_CONNECT_CONFIRM_NTFY,
            AutelEmptyConvert(),
```

```
        ).canListen(true)

    /** 切换高速模式接口
     * 0-普通模式, 1-高速上传文件（上传任务文件）, 2-高速下载模式（下载照片或视频）, 3-独占模式（升级）*/
    val KeyAirlinkControlHighSpeed: AutelActionKeyInfo<HighSpeedEnum, Void> =
        AutelActionKeyInfo(
            component.value,
            ALinkConstant.AIRLINK_CONTROL_HIGH_SPEED_MSG,
            AirLinkHighSpeedConvert(),
            AutelEmptyConvert(),
        ).canPerformAction(true)

    /** 获取高速模式接口 */
    val KeyAirlinkGetHighSpeed: AutelActionKeyInfo<Void, HighSpeedEnum> =
        AutelActionKeyInfo(
            component.value,
            ALinkConstant.AIRLINK_GET_HIGH_SPEED_MSG,
            AutelEmptyConvert(),
            AirLinkHighSpeedConvert(),
        ).canPerformAction(true)

    /** 频段设置接口（debug） */
    val KeyAirlinkSetDebugBandmode: AutelActionKeyInfo<Int, Void> =
        AutelActionKeyInfo(
            component.value,
            ALinkConstant.AIRLINK_SET_DEBUG_BANDMODE_MSG,
            AutelIntConvert(),
            AutelEmptyConvert(),
        ).canPerformAction(true)

    /** 动态调整频段方式（debug） */
    val KeyAirlinkSetDebugDynamicAdjust: AutelActionKeyInfo<DynamicAdjustEnum, Void> =
        AutelActionKeyInfo(
            component.value,
            ALinkConstant.AIRLINK_SET_DEBUG_DYNAMIC_ADJUST_MSG,
            AirLinkDynamicAdjustConvert(),
            AutelEmptyConvert(),
        ).canPerformAction(true)

    /** 进入图传静默模式 */
    val KeyAirlinkEnterSilenceMode: AutelActionKeyInfo<Void, Void> =
        AutelActionKeyInfo(
            component.value,
            ALinkConstant.AIRLINK_ENTER_SILENCE_MODE_MSG,
            AutelEmptyConvert(),
            AutelEmptyConvert(),
        ).canPerformAction(true)

    /** App图传显示模式 */
    val KeyAirlinkAppSplitScreenInfo:
AutelActionKeyInfo<List<GimbalTransmissionBean>, Void> =
        AutelActionKeyInfo(
            component.value,
            ALinkConstant.AIRLINK_APP_SPLIT_SCREEN_INFO_MSG,
            SplitScreenInfoConvert(),
            AutelEmptyConvert(),
```

```
        ).canPerformAction(true)

}
```

3.8 遥控器相关功能介绍

3.8.1 遥控器相关命令发送，和飞机一样都是通过KeyManager发送命令，只是获取KeyManager的方式不同，以查询遥控器设备信息为例：

```
val remoteKeyManager =
DeviceManager.getDeviceManager().getFirstRemoteDevice()?.getKeyManager()
        remoteKeyManager?.performAction(
            KeyTools.createKey(RemoteControllerKey.KeyRcDeviceInfo),
            null,
            object :
CommonCallbacks.CompletionCallbackWithParam<List<DroneVersionItemBean>> {
                override fun onSuccess(t: List<DroneVersionItemBean>?) {
                }

                override fun onFailure(error: IAutelCode, msg: String?) {
                }

            })
```

3.9 注意事项

3.9.1 performAction方法和setValue以及getValue不能直接通过事件名称来判断，需要根据对应模块的Key来判断，例如只有canGet(true)的时候，才可以调用getValue方法