class DeviceLogManager

method queryLogInfoList

```
fun queryLogInfoList(sn: String, deviceType: DeviceLogTypeEnum, callback: 
CommonCallbacks.CompletionCallbackWithParam<DeviceSysLog?>)
```

Description: Queries device logs.

Input

- **sn**: Device SN
- **deviceType**: Device type. For details, see DeviceLogTypeEnum.
- **callback**: Query callback. For details, see CommonCallbacks. For details about the callback log objects, see DeviceSysLog.

Output

None

method downloadPowerLog

```
fun downloadPowerLog(powerLogs: List<DeviceSysOncePowerLog>,
                    callback: 
CommonCallbacks.DownLoadCallbackCompletionWithProgressAndFile<Double, 
DeviceSysLogModule>)
```

Description: Downloads logs by power-on records.

Input

- **powerLogs**: Power-on records.
- **callback**: Query callback. For details, see CommonCallbacks. For details about the callback log objects, see DeviceSysLogModule.

  Note: Every completed **DeviceSysLogModule** will be notified through **onProgressFileUpdate**.

Output

None

method downloadModuleLog

```
fun downloadModuleLog(moduleLogs: List<DeviceSysLogModule>,
                        callback: 
CommonCallbacks.DownLoadCallbackCompletionWithProgressAndFile<Double, 
DeviceSysLogModule>)
```

Description: Downloads logs by module.

Input

- **moduleLogs**: Device module list (module logs of different power-on records can be downloaded together).
- **callback**: Query callback. For details, see CommonCallbacks. For details about the callback log objects, see DeviceSysLogModule.

  Note: Every completed **DeviceSysLogModule** will be notified through **onProgressFileUpdate**.

Output

None

data class DeviceSysLog

```
data class DeviceSysLog(var type: DeviceLogTypeEnum,
                        var sn: String,
                        var powerLogList: List<DeviceSysOncePowerLog>) {

}
```

Description: Device log objects, with all logs included.

Properties

- **type**: Device type. For details, see DeviceLogTypeEnum.
- **sn**: Device SN.
- **powerLogList**: Power-on records. For details, see DeviceSysOncePowerLog.

data class DeviceSysOncePowerLog

```
data class DeviceSysOncePowerLog(var type: DeviceLogTypeEnum,
                                 var sn: String,
                                 var index: Int,
                                 var time: Long,
                                 var moduleList: List<DeviceSysLogModule>) {
}
```

Description: Power-on record objects, which contains the logs of all modules of the current power-on record. A device can store up to 16 power-on records, and the latest records will replace the earliest ones in turn once 16 records have been stored.

Properties

- **type**: Device type. For details, see DeviceLogTypeEnum.
- **sn**: Device SN.
- **index**: Ordering of power-on records.
- **time**: Power-on timestamps.
- **moduleList**: Device module log list. For details, see DeviceLogTypeEnum.

data class DeviceSysLogModule

```
data class DeviceSysLogModule(
    var mode: DeviceLogTypeEnum,
    var sn: String,
    var index: Int,
    var type: DeviceLogEnum,
    var time: Long,
    var size: Int,
    var downPath: String = "",
    var name: String = "",
    var localPath: String = "") {
}
```

Description: Logs of a specific device module.

Properties

- **mode**: Device type. For details, see DeviceLogTypeEnum.
- **sn**: Device SN.
- **index**: Ordering of power-on records.
- **type**: Log type. For details, see DeviceLogTypeEnum.
- **time**: Power-on timestamps.
- **size**: Module log size (value generated after download).
- **downPath**: Log file download path (not for users).
- **name**: Log file name.
- **localPath**: Path the log is downloaded to (value generated after download).

```
enum class DeviceLogEnum  (var value: String, val type: Int) {

    /**
     * unkown
     */
    UNKOWN("Unkown",0),

    /**
     * Rc Transmission
     */
    RCTRANSMISSION("RcTransmission",1),

    /**
     * Air Transmission
     */
    AIRTRANSMISSION("AirTransmission",2),

    /**
     * Air Skylink
     */
    AIRSKYLINK("AirSkylink",3),

    /**
     * Air Vision
     */
    AIRVISION("AirVision",4),

    /**
     * Rc Android
     */
    RCANDROID("RcAndroid",5),

    /**
     * Air Camera
     */
    AIRCAMERA("AirCamera",6);
}
```

Device module code sample

```
enum class DeviceLogTypeEnum  (var value: String) {
    /**
     * unkown
     */
    UNKOWN("Unkown"),

    /**
     * Rc
     */
    RC("RC"),

    /**
     * Drone
     */
    DRONE("UAV");
}
```

Device type code sample

```
class CommonCallbacks {
    /**
     * Completion callback with parameters
     */
    interface CompletionCallbackWithParam<T> {
        /**
         * Success callback
         * @param t result return when call successfully
         */
        fun onSuccess(t: T?)
        /**
         * Failure callback
         * @param error code when call failed
         * @param msg msg when call failed
         */
        fun onFailure(error: IAutelCode, msg: String? = null)
    }


    /**
     * Completion callback with progress
```

```
 */
interface CompletionCallbackWithProgress<T> {
    /**
     * Callback when call progress changed
     * @param progress the progress for the call
     */
    fun onProgressUpdate(progress: T)
    /**
     * Success callback
     */
    fun onSuccess()
    /**
     * Failure callback
     * @param error code when call failed
     */
    fun onFailure(error: IAutelCode)
}

/**
 * Completion callback with progress
 */

interface DownLoadCallbackCompletionWithProgressAndFile<T,D> {
    /**
     * Callback when call progress changed
     * @param progress the progress for the call
     */
    fun onProgressUpdate(progress: T, speed: Double)

    /**
     * Callback when all files have been downloaded
     * @param file file path for success downloaded
     */
    fun onProgressFileUpdate(file : File?, mode: D?)

    /**
     * Callback when all files have been downloaded
     */
    fun onSuccess()
    /**
     * Failure callback
     * @param error code when call failed
     */
    fun onFailure(error: IAutelCode)
}

/**
 * Completion callback with no parameters
 */
interface CompletionCallback {
    /**
     * Success callback
     */
    fun onSuccess()

    /**
     * Failure callback
     * @param error code when call failed
```

```
         */
        fun onFailure(code: IAutelCode, msg: String? = null)
    }
}
```