押解设备 SDK 说明

version 1.2.9.1

更新说明

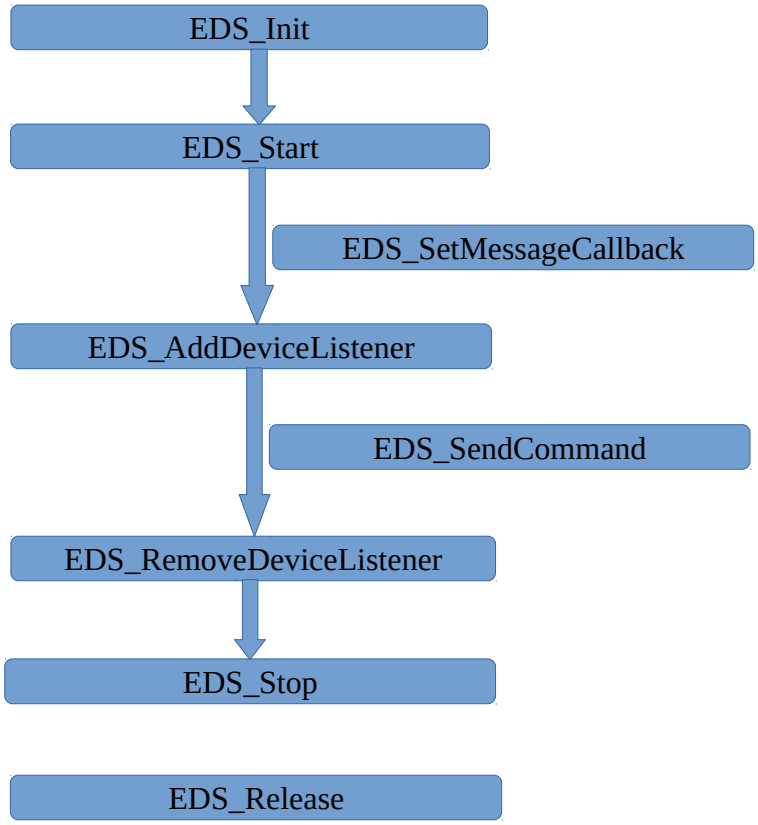
1.2.4.1	初始版本;
1.2.5.1	gps 定位信息中增加了 nCoordinate; lbs 定位信息种增加了 defLatitude,defLngitude,nCoordinate;
1.2.9.1	gps 定位信息删除了 usLatType,usLngType; 重新调整了 Gps 定位信息、Lbs 定位信息、设备消息格式; 修改了接口,将 EDS_Start 的返回值由 int 改成 unsigned long long

目录

1.SDK 简介

设备 sdk 与设备代理程序一起使用，通过 sdk 接收与控制设备代理程序中的设备。

2.接口调用顺序



3.数据结构说明

3-1.消息类型

type	value	describe
MT_ONLINE	1	device online message
MT_ALIVE	2	device alive message
MT_OFFLINE	3	device offline message
MT_ALARM_LOOSE	4	device loose alarm message
MT_ALARM_LOWPOWER	5	device low power alarm message
MT_LOCATE_GPS	6	device GPS location message
MT_LOCATE_LBS	7	device LBS location message
MT_COMMAND	8	device command reply message
MT_SERVER_CONNECT	0x0e	service connect message
MT_SERVER_DISCONNECT	0x0f	server disconnect message

3-2.控制指令类型

type	value	describe

CMD_RESET	0	重启设备
CMD_BIND	1	设备绑定/解绑
CMD_TASK	2	设备开启/关闭任务
CMD_FLEE	3	设备开启/关闭告警，
CMD_SET_INTERVAL	4	设置设备定位时间间隔

3-3.设备控制指令信息

```
struct DeviceCommandInfo
{
	char szDeviceId[20];
	char szFactoryId[4];
	int nCommand;
	int nParam1;
	int nParam2;
	int nParam3;
}
```

参数：

szDeviceId	in	设备编号
szFactoryId	in	设备厂商
nCommand	in	控制指令，参见 3-2
nParam1	in	控制参数值
nParam2	out	设备端返回值，0 表示成功，-1 表示失败
nParam3		预留字段

控制参数说明：

nCommand	param1
CMD_RESET	不使用，默认为 0
CMD_BIND	不使用，默认为 0
CMD_TASK	设为 1，表示开启任务；设为 0，表示结束任务
CMD_FLEE	设为 1，表示开启告警；设为 0，表示关闭任务
CMD_SET_INTERVAL	表示时间间隔值，单位为秒，最小值为 10，最大值不超过 300

3-4.基站信息

```
struct BaseStation
{
	int nLocateAreaCode;
	in nCellId;
	int nSignalIntensity;
}
```

参数：

nLocateAreaCode	in	LAC,定位区域编号
nCellId	in	CI，信号扇面值
nSignalIntensity	in	信号强度

3-5.WIFI 信息

```
struct WifiInformation
{
    char szWifiTagName[32];
    char szWifiMacAddress[32];
    int nWifiSignalIntensity;
}
```

参数：

szWifiTagName	in	WIFI 名称
szWifiMacAddress	in	WIFI MAC 地址
nWifiSignalIntensity	in	WIFI 信号强度

3-6.设备消息

```
struct DeviceMessage
{
    char szDeviceId[20];
    char szFactoryId[4];
    unsigned short usMessageType;
    unsigned short usMessageTypeExtra;
    unsigned short usDeviceBattery;
    unsigned long long ulMessageTime;
}
```

参数：

szDeviceId	in	设备编号
szFactoryId	in	设备厂商
usMessageType	in	消息类型
usMessageTypeExtra	in	类型扩展，用于 alarm 时，1 表示产生告警，0 表示解除告警
usDeviceBattery	in	设备电量
ulMessageTime	in	消息时间，单位秒

3-7.设备 GPS 定位消息

```
struct DeviceLocateGpsMessage
{
    char szDeviceId[20];
    char szFactoryId[4];
    unsigned long ulLocateTime;
    unsigned short usSatelliteCount;
    unsigned short usSignalIntensity;
    unsigned short nCoordinate;
    unsigned short usDeviceBattery;
    double dLatitude;
    double dLngitude;
    double dSpeed;
    double dDirection;
}
```

参数：

szDeviceId	in	设备编号

szFactoryId	in	设备厂商
usDeviceBattery	in	设备电量
ulLocateTime	in	定位时间
usSatelliteCount	in	gps 卫星数
usSignalIntensity	in	信号强度
dLatitude	in	纬度
dLngitude	in	经度
nCoordinate	in	坐标系
dSpeed	in	速度
dDirection	in	方向

3-8.设备 LBS 定位消息

```
struct DeviceLocateLbsMessage
{
	char szDeviceId[20];
	char szFactoryId[4];
	unsigned long long ulLocateTime;
	double dRefLatitude;
	double dRefLngitude;
	unsigned short nCoordinate;
	unsigned short usDeviceBattery;
	unsigned short nNationCode;
	unsigned short nNetCode;
	int nBaseStationCount;
	int nDetectedWifiCount;
	BaseStation * pBaseStationList;
	WifiInformation * pDetectedWifiList;
} DeviceLocateLbsMessage;
```

参数：

szDeviceId	in	设备编号
szFactoryId	in	设备厂商
dRefLatitude	in	参考纬度
dRefLngitude	in	参考经度
nCoordinate	in	坐标系
usDeviceBattery	in	设备电量
ulLocateTime	in	定位时间
nNationCode	in	国家编码
nNetCode	in	网络编号
nBaseStationCount	in	基站个数
pBaseStationList	in	基站列表
nDetectedWifiCount	in	检测到的 wifi 个数
pDetectedWifiList	in	检测到的 wifi 列表

3-9.代理信息

```
struct ProxyInfo
{
	char szProxyIp[20];
	unsigned short usPort1;
	unsigned short usPort2;
}
```

参数：

szProxyIp	in	代理程序地址
usPort1	in	代理消息端口
usPort2	in	代理控制端口

4.接口函数说明

4-1.初始化 EDS_Init

函数：int __stdcall EDS_Init()

参数：

返回值：0 表示成功

说明：理论上在一个进程内只需要调用一次初始化函数

4-2.释放 EDS_Release

函数：int __stdcall EDS_Release()

参数：

返回值：0 表示成功

说明：EDS_Init()与 EDS_Release()需要成对使用，即调用一次 EDS_Init(),在释放时需要相应调用一次 EDS_Release()，这样保证能够释放成功。

4-3.开启 EDS_Start

函数：unsigned long long __stdcall EDS_Start(const char * host, unsigned short port1,unsigned short port2,

fMessageCallback fMsgCb, void * pUserData);

参数：

host	in	设备代理的地址
port1	in	设备代理的消息端口
port2	in	设备代理的控制端口
fMsgCb	in	消息回调函数
pUserData	in	用户传递的数据

返回值：开启成功返回一个大于 0 的实例值，用于其他操作；0 值表示失败；

说明：

消息回调函数原型：void (__stdcall fMessageCallback)(unsigned int uiMsgType, unsigned int uiMsgSeq, unsigned long long ulMsgTime, void * pMsg, void * pUserData);

参数：

uiMsgType	消息类型，同时决定消息数据类型
uiMsgSeq	回调消息的传递序号
ulMsgTime	回调消息的传递时间
pMsg	回调消息数据，具体数据类型与消息类型有关
pUserData	用户传递数据

消息类型与实际消息关系：

uiMsgType	pMsg
MT_ONLINE	DeviceMessage
MT_ALIVE	DeviceMessage
MT_OFFLINE	DeviceMessage
MT_ALARM_LOOSE	DeviceMessage
MT_ALARM_LOWPOWER	DeviceMessage
MT_LOCATE_GPS	DeviceLocateGpsMessage
MT_LOCATE_LBS	DeviceLocateLbsMessage
MT_COMMAND	DeviceCommandInfo
MT_SERVER_CONNECT	ProxyInfo
MT_SERVER_DISCONNECT	ProxyInfo

在设置回调函数的前提下，实际连接代理成功时将在回调函数中回调出 MT_SERVERE_CONNECT 消息，如果没有成功，将在 90 秒之后返回 MT_SERVER_DISCONNECT 消息。后续其他操作实际需要等待 MT_SERVER_CONNECT 消息后才能调用成功，否则调用将会失败。基于这一特性，强烈建议 fMsgCb 传递一个可用的消息回调函数，以防止错过 MT_SERVER_CONNECT 或者 MT_SERVER_DISCONNECT 消息。使用 EDS_SetMessageCallback()进行设置消息回调函数也是可以的，但是因为时间差关系，不保证能够接收到 MT_SERVER_CONNECT 消息。

4-4.设置消息回调函数 EDS_SetMessageCallback

函数：int __stdcall EDS_SetMessageCallback(unsigned long long inst, fMessageCallback fMsgCb, void * pUserData);

参数：

inst	in	EDS_Start 的返回的实例值
fMsgCb	in	消息回调函数
pUserData	in	用户传递数据

返回值：0 表示成功，-1 表示失败

说明：如果在 EDS_Start()中未设置消息回调函数，可以调用该接口进行设备。

4-5.添加设备监听 EDS_AddDeviceListener

函数：int __stdcall EDS_AddDeviceListener(unsigned long long inst, const char * factroyId, const char * deviceId);

参数：

inst	in	EDS_Start 返回的实例值
factoryId	in	需要添加监听的设备厂商
deviceId	in	需要添加监听的设备编号

返回值：-1 表示添加监听失败，0 表示成功

说明：该接口每次只允许添加一个设备加入监听，加入监听的设备才能进行消息传递与控制传递；

要监听多个设备加入监听，需要对每个设备调用一次接口。当传递的 factoryId 与 deviceId 都为 NULL 值时，表示监听所有在当前设备代理中的设备。

4-6.移除设备监听 EDS_RemoveDeviceListener

函数：int __stdcall EDS_RemoveDeviceListener(unsigned long long inst, const char * factoryId, const char * deviceId);

参数：

inst	in	EDS_Start 返回的实例值
factoryId	in	需要移除监听的设备厂商
deviceId	in	需要移除监听的设备编号

返回值：0 表示成功，-1 表示失败

说明：该接口每次只移除一个监听设备，移除后不再接收该设备的信息。当传递的 factoryId 和 deviceId 都为 NULL 值，表示移除所有监听设备。

4-7.发送设备指令 EDS_SendCommand

函数：int __stdcall EDS_SendCommand(unsigned long long inst, DeviceCommandInfo cmdInfo);
参数：

inst	in	EDS_Start 返回的实例值
cmdInfo	in	设备指令信息

返回值：大于 0 表示指令下发设备成功，-1 表示失败

说明：返回值是实际 command 下发设备返回的命令序列值，该序列值在消息回调函数中将返回设备端执行下发指令的实际返回值，0 表示设备执行指令成功，-1 表示设备执行指令失败。参见 3-2，3-3

4-8.关闭 EDS_Stop

函数：int __stdcall EDS_Stop(unsigned long long inst);

参数：

inst	in	EDS_Start 的返回值

返回值：-1 表示失败，0 表示成功

说明：

5.示例代码

回调函数代码：

```
void __stdcall MsgCb(unsigned int uiMsgType, unsigned int uiMsgSeq, unsigned long long ulMsgTime, void *
                     pMsg, void * pUserData)

{

     ...

}
```

主运行逻辑代码：

```
EDS_Init();
INSTANCE_VALUE inst = EDS_Start(szHost, usPort1, usPort2, MsgCb, NULL);
if (inst > 0) {
     printf("start sdk, connect %s:%hu|%hu\n", szHost, usPort1, usPort2);
     g_bRun = true;
     g_nVal = nInst;
     EDS_AddDeviceListener(inst, NULL, NULL);
     EDS_RemoveDeviceListener(inst, NULL, NULL);
     EDS_Stop(inst);
}
EDS_Release();
```