SDK 使用手册

版本	描述	日期	创建人
V1.0.0	创建	2019.6.4	Mizue

目录

1 内容简介

1.1 概述

欢迎使用 SDK 编程手册，SDK 是基于 TCP 协议封装的 API，本文档详细描述 了开发包中各个函数实现的功能、接口及其函数之间的调用关系和示例实现.

1.2 适用范围

SDK 支持的系统说明： win32 SDK 开发语言：c#、c++、linux、Java

2 编程引导

2.1 接口调用流程

注:SDK 初始化在整个程序运行期间只需初始一次 全局回调函数为连接 到 SDK 的所有相机都会触发此回调函数相机回调函数只会被注册的相机 句柄所连接的相机触发

2.2 接口使用建议

若查询结果或上传数据内容在回调函数中获取如有复杂操作请拷贝数据 到自定义线程中进行。拷贝数据注意栈数据会在回调函数结束时立刻释放， 需自行申请内存

3 数据定义

3.1 SDK 版本信息

【定义】 struct ZBXSdkVersion {
char sdk_version[64];
char protocl_version[64];
char sdk_code_version[64];
char min_firmware_ver[64];
char algorithm_version[64]; };

【成员】

成员	描述
char dev	设备序列号
char protocol_ver	协议版本
char firmware_ver	固件版本
char code_ver	应用程序版本
char build_time	应用编译时间
char resv	保留
char systemp_type	系统类型
char plateform	硬件平台
char sensor_type	传感器型号
char algorithm_ver	算法版本
char min_sdk_ver	最低 sdk 版本
unsigned int min_client_ver	最低客户端版本
char kernel_version	内核版本
char lcd_type;	显示屏型号
char recv	保留

【注意事项】 无。

【关联】 ZBX_GetVersion

3.2 相机句柄

【定义】 struct ZBX _Cam;

3.3 网络参数

【定义】 struct SystemNetInfoEx {
char mac[20];
char ip[20];
char netmask[20];
char gateway[20];
char manufacturer[16];
char platform[32];
char system[32];
char version[64];
char ip_2[16];
char netmask_2[16];
char dns[16];
char dhcp_enable;
char resv[64];
};

【注意事项】 无

【关联】 ZBX_SetNetConfig
ZBX _GetNetConfig

3.4 注册人员信息

【定义】 struct FaceFlags {
char faceID[20];
char faceName[16];
int role;
unsigned int wgCardNO;
unsigned int effectTime;
unsigned int effectStartTime;
short version;
char resv[8182];
};

【成员】

成员	描述
char faceID	人员 ID
char faceName	人员姓名
int role	人员角色 0：普通人员 1：白名单人员 2：黑名单人员
unsigned int wgCardNO	韦根协议门禁卡号
unsigned int effectTime	有效期截止时间，从 1970 年 1 月 1 日 0 时 0 分 0 秒到截止时间的秒数 （0xFFFFFFFF 表示永久有效，0

	表示永久失效）
unsigned int effectStartTime	有效期起始时间，1970 年 1 月 1 日 0 时 0 分 0 秒到起始时间的秒数 （0 表示未初始化）
short version	特征数据版本，用于同步注册特征数据时相机校验 未使用特征数据注册填 0
char resv	预留

【注意事项】 无
【关联】 人员注册

3.5 图像数据

【定义】 struct FaceImage {
int img_seq;
int img_fmt;
unsigned char *img;
int img_len;
int width;
int height;
};
注：目前只支持 jpg、jpeg 图片

其余定义 详见头文件

4 接口定义

```
/*
 * @brief   取得sdk版本
 * @param   version[out] sdk版本信息
 * @return  版本号    【返回值固定3】
 */
SDK_API int SDK_CALL ZBX_GetVersion(ZBXSdkVersion* version);

/**
 * @brief   sdk初始化，
 * @brief   1)最多连接9个相机。
 * @brief   2)与ZBX_InitEx这两个函数，二者只能调其中一个。
 * @brief   3)如果想连接更多相机，请调用ZBX_InitEx
 * @param   无
 * @return  无
```

```
 */
SDK_API void SDK_CALL ZBX_Init();

/**
 * @brief   sdk初始化
 * @param   maxCamNum[in] 最大连接相机数。
 * @return  无
 */
SDK_API void SDK_CALL ZBX_InitEx(unsigned int maxCamNum);



/**
 * @brief   sdk反初始化
 * @param   无
 * @return  无
 */
SDK_API void SDK_CALL ZBX_DeInit();

/**
 * @brief   注册连接事件回调函数,须先调用ZBX_SetNotifyConnected(1)才生效
 * @param   cb[in] 回调函数指针
 * @param   usrParam[in] 用户参数
 * @return  无
 */
SDK_API void SDK_CALL ZBX_RegConnectEventCb(ZBX_ConnectEventCb_t cb,
                                    int usrParam);

/**
 * @brief   注册连接事件回调函数,须先调用ZBX_SetNotifyConnected(1)才生效
 * @param   cb[in] 回调函数指针
 * @param   usrParam[in] 用户参数
 * @return  无
 */
SDK_API void SDK_CALL ZBX_RegConnectEventCbEx(ZBX_Cam* cam,
                                      ZBX_ConnectEventCb_t cb,
                                      int usrParam);


/**
 * @brief   注册人脸抓拍数据接收回调函数
 * @param   cb[in] 回调函数指针
 * @param   usrParam[in] 用户参数
 * @return  无
 */
SDK_API void SDK_CALL ZBX_RegFaceRecoCb(ZBX_Cam* cam, ZBX_FaceRecoCb_t cb,
                                void* usrParam);

/**
```

```
 * @brief   注册人脸查询数据回调函数
 * @param   cb[in] 回调函数指针
 * @param   usrParam[in] 用户参数
 * @return  无
 */
SDK_API void SDK_CALL ZBX_RegFaceQueryCb(ZBX_Cam* cam, ZBX_FaceQueryCb_t cb,
                                         void* usrParam);

/**
* @brief   注册人脸查询数据回调函数
* @param   cb[in] 回调函数指针
* @param   usrParam[in] 用户参数
* @return  无
*/
SDK_API void SDK_CALL ZBX_RegFaceQueryCbEx(ZBX_Cam* cam, ZBX_FaceQueryCbEx_t cb,
          void* usrParam);



/**
 * @brief   注册搜索相机回调函数
 * @param   cb[in] 回调函数指针
 * @param   usrParam[in] 用户参数
 * @return  无
 */
SDK_API void SDK_CALL ZBX_RegDiscoverIpscanCb(discover_ipscan_cb_t cb,
                                              int usrParam);

/**

 * @brief   搜索相机
 * @param   无
 * @return  无
 */
SDK_API void SDK_CALL ZBX_DiscoverIpscan();

/**
 * @brief   通过mac地址，跨网段设置相机IP
 * @param   mac[in] mac地址
 * @param   ip[in] 相机IP
 * @param   netmask[in] 子网掩码
 * @param   gateway[in] 默认网关
 * @return  无
 */
SDK_API void SDK_CALL ZBX_SetIpBymac(const char* mac, const char* ip,
                                     const char* netmask, const char* gateway);

/**
 * @brief   是否连接相机
```

```
 * @param   cam[in]   相机句柄
 * @return  0 未连接
 * @return  1 已连接
 */
SDK_API int SDK_CALL ZBX_Connected(ZBX_Cam* cam);

/**
 * @brief   连接相机。请用ZBX_ConnectEx替代
 * @brief   默认为不支持重连接,如需重连，请使用下一个函数
 * @param   ip[in]   相机ip
 * @param   port[in]   相机端口，固定为9527
 * @param   usrName[in]   用户名，目前版本无效，传空即可
 * @param   password[in]   密码，目前版本无效，传空即可
 * @param   errorNum[out]  连接失败错误号，0为连接成功，-1为连接失败
 * @return  NULL 连接失败
 * @return  非NULL 连接成功
 */
SDK_API ZBX_Cam* SDK_CALL ZBX_Connect(const char* ip, unsigned short port,
                                      const char* usrName, const char* password,
                                      int* errorNum);

/**
 * @brief   连接相机
 * @param   ip[in]   相机ip
 * @param   port[in]   相机端口，固定为9527
 * @param   usrName[in]   用户名，目前版本无效，传空即可
 * @param   password[in]   密码，目前版本无效，传空即可
 * @param   errorNum[out]  连接失败错误号，0为连接成功，-1为连接失败
 * @param   channel[in]
 * 码流通道号，有特殊需求的用户使用，无特殊需求用户直接传0通道即可
 * @param   autoReconn[in] 自动重连标志，为1自动重连，0不会自动重连
 * @return  autoReconn为1时，固定返回相机句柄
 * @return  autoReconn为0时，连接成功时返回相机句柄，连接失败时返回NULL
 * @return  建议用户使用自动重连,autoReconn为1
 */
SDK_API ZBX_Cam* SDK_CALL ZBX_ConnectEx(const char* ip, unsigned short port,
                                        const char* usrName, const char* password,
                                        int* errorNum, unsigned int channel,
                                        int autoReconn);


/**
 * @brief   断开相机
 * @param   cam[in]   相机句柄
 * @return  无
 */
SDK_API void SDK_CALL ZBX_DisConnect(ZBX_Cam* cam);
```

```
/**
 * @brief  开始显示视频流
 * @param  cam[in]  相机句柄
 * @param  hWnd[in]  窗口句柄
 * @return 无
 */
SDK_API void SDK_CALL ZBX_StartStream(ZBX_Cam* cam, HWND hWnd);


/**
 * @brief  终止显示某一个窗口句柄视频流
 * @param  cam[in]  相机句柄
 * @return 无
 */
SDK_API void SDK_CALL ZBX_StopStreamEx(ZBX_Cam* cam, HWND hWnd);

/**
 * @brief  终止显示视频流
 * @param  cam[in]  相机句柄
 * @return 无
 */
SDK_API void SDK_CALL ZBX_StopStream(ZBX_Cam* cam);

/*
* @brief 获取视频实时fps
* @param  cam[in]  相机句柄
*/
SDK_API int SDK_CALL ZBX_GetStreamFps(ZBX_Cam* cam);

/**
 * @brief  触发相机
 * @param  cam  相机句柄
 * @param  usrParam  此版本无效，传0即可
 * @return 0 触发成功
 * @return -1 触发失败
 */
SDK_API int SDK_CALL ZBX_Trigger(ZBX_Cam* cam, int usrParam);


/**
 * @brief  获取人脸相机固件信息
 * @param  cam[in]  相机句柄
 * @param  version[out] 固件版本信息
 * @return 0 获取成功
 * @return <0 获取失败
 */
SDK_API int SDK_CALL ZBX_GetFaceSystemVersion(ZBX_Cam* cam,
                                              SystemVersionInfo* version);
```

```
/**
 * @brief   获取相机系统时间
 * @param   cam[in]   相机句柄
 * @param   sysTime[out] 相机时间信息
 * @return  0 获取成功
 * @return  <0 获取失败
 */
SDK_API int SDK_CALL ZBX_GetSystemTime(ZBX_Cam* cam, SystemTime* sysTime);

/**
 * @brief   设置相机系统时间
 * @param   cam[in]   相机句柄
 * @param   sysTime[in] 相机时间信息
 * @return  0 设置成功
 * @return  <0 设置失败
 */
SDK_API int SDK_CALL ZBX_SetSystemTime(ZBX_Cam* cam, SystemTime* sysTime);


/**
 * @brief   获取闸机韦根类型
 * @param   cam[in]   相机句柄
 * @param   type[out] 韦根类型 枚举类型为WG26,WG34, WG36, WG44，目前只支持WG26,
 * WG34
 * @return  0 获取成功
 * @return  <0 获取失败
 */
SDK_API int SDK_CALL ZBX_GetWiegandType(ZBX_Cam* cam, int* type);

/**
 * @brief   设置闸机韦根类型
 * @param   cam[in]   相机句柄
 * @param   type[in] 韦根类型 枚举类型为WG26,WG34, WG36, WG44，目前只支持WG26,
 * WG34
 * @return  0 设置成功
 * @return  <0 设置失败
 */
SDK_API int SDK_CALL ZBX_SetWiegandType(ZBX_Cam* cam, int type);


/**
 * @brief   获取计算人员通过次数的间隔时间，在该时间内人员只算通过一次
 * @param   cam[in]   相机句柄
 * @param   interval[out] 计算人员通过次数的间隔时间，单位：秒
 * @return  0 获取成功
 * @return  <0 获取失败
 */
SDK_API int SDK_CALL ZBX_GetClusterTimesInterval(ZBX_Cam* cam,
                                                 unsigned int* interval);
```

```
/**
 * @brief   设置计算人员通过次数的间隔时间，在该时间内人员只算通过一次
 * @param   cam[in]   相机句柄
 * @param   interval[out] 计算人员通过次数的间隔时间，单位：秒
 * @return  0 获取成功
 * @return  <0 获取失败
 */
SDK_API int SDK_CALL ZBX_SetClusterTimesInterval(ZBX_Cam* cam,
                                                 unsigned int interval);

/**
 * @brief   获取人脸比对确性分数
 * @param   cam[in]   相机句柄
 * @param   score[out] 确信分数（0-100分）
 * @return  0 获取成功
 * @return  <0 获取失败  配置.h并没有更新
 */
SDK_API int SDK_CALL ZBX_GetMatchScore(ZBX_Cam* cam, int* score);

/**
 * @brief   设置人脸比对确性分数
 * @param   cam[in]   相机句柄
 * @param   score[in] 确信分数（0-100分）
 * @return  0 设置成功
 * @return  <0 设置失败  配置.h并没有更新
 */
SDK_API int SDK_CALL ZBX_SetMatchScore(ZBX_Cam* cam, int score);


/**
 * @brief   获取活体检测开关状态
 * @param   cam[in]   相机句柄
 * @param   enable[out]  0:关 1:开
 * @return  0 获取成功
 * @return  <0 获取失败  配置.h并没有更新
 */
SDK_API int SDK_CALL ZBX_GetAliveDetectEnable(ZBX_Cam* cam, int* enable);

/**
 * @brief   设置活体检测开关
 * @param   cam[in]   相机句柄
 * @param   enable[in] 0:关 1:开
 * @return  0 设置成功
 * @return  <0 设置失败  配置.h并没有更新
 */
SDK_API int SDK_CALL ZBX_SetAliveDetectEnable(ZBX_Cam* cam, int enable);

/**
```

```
 * @brief   获取外接显示屏标题
 * @param   cam[in]   相机句柄
 * @param   screen_title[out]  外接显示屏标题 UTF8格式 最大64个字节
 * @return  0 获取成功
 * @return  <0 获取失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_GetScreenOsdTitle(ZBX_Cam* cam, char* screen_title);

/**
 * @brief   设置外接显示屏标题
 * @param   cam[in]   相机句柄
 * @param   screen_title[in] 外接显示屏标题 UTF8格式 最大64个字节
 * @return  0 设置成功
 * @return  <0 设置失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_SetScreenOsdTitle(ZBX_Cam* cam, char* screen_title);


/**
 * @brief   查看网络参数配置
 * @param   cam[in]   相机句柄
 * @param   netInfo[out] 网络参数
 * @return  0 获取成功
 * @return  <0 获取失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_GetNetConfig(ZBX_Cam* cam, SystemNetInfo* netInfo);

/**
 * @brief   设置网络参数配置
 * @param   cam[in]   相机句柄
 * @param   netInfo[in] 网络参数
 * @return  0 设置成功
 * @return  <0 设置失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_SetNetConfig(ZBX_Cam* cam, SystemNetInfo* netInfo);

/**
 * @brief   获取led灯模式
 * @param   cam[in]   相机句柄
 * @param   led_mode[out] led灯模式 1：常亮 2：自动控制 3：常闭
 * @return  0 获取成功
 * @return  <0 获取失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_GetLedMode(ZBX_Cam* cam, char* led_mode);

/**
 * @brief   设置led灯模式
 * @param   cam[in]   相机句柄
 * @param   led_mode[in] led灯模式 1：常亮 2：自动控制 3：常闭
```

```
 * @return  0 设置成功
 * @return  <0 设置失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_SetLedMode(ZBX_Cam* cam, char led_mode);

/**
 * @brief   获取led亮度
 * @param   cam[in]   相机句柄
 * @param   led_level[out] 亮度 1~100
 * @return  0 获取成功
 * @return  <0 获取失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_GetLedLevel(ZBX_Cam* cam, char* led_level);

/**
 * @brief   设置led灯亮度
 * @param   cam[in]   相机句柄
 * @param   led_level[in] 亮度 1~100
 * @return  0 设置成功
 * @return  <0 设置失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_SetLedLevel(ZBX_Cam* cam, char led_level);



/**
 * @brief   获取485输出协议编号
 * @param   cam[in]   相机句柄
 * @param   rs485_protocal_no[out] rs485输出协议编号 0：表示关
 * @return  0 获取成功
 * @return  <0 获取失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_GetRS485ProtocalNo(ZBX_Cam* cam,
                                            char* rs485_protocal_no);

/**
 * @brief   设置485输出协议编号
 * @param   cam[in]   相机句柄
 * @param   rs485_protocal_no[in] rs485输出协议编号 0：表示关
 * @return  0 设置成功
 * @return  <0 设置失败  参考错误码
 */
SDK_API int SDK_CALL ZBX_SetRS485ProtocalNo(ZBX_Cam* cam, char rs485_protocal_no);


/**
 * @brief   触发白名单报警，强制模式开闸
 * @param   cam[in]   相机句柄
 * @param   inout[in]  输入、输出 1输入 2输出
```

```
 * @param   onoff[in] 开关 1连通 0断开
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_WhiteListAlarm(ZBX_Cam* cam, int inout, int onoff);

/**
 * @brief   触发黑名单报警,强制模式开闸
 * @param   cam[in]   相机句柄
 * @param   inout[in]  输入、输出 1输入 2输出
 * @param   onoff[in] 开关 1连通 0断开
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_BlackListAlarm(ZBX_Cam* cam, int inout, int onoff);



//============================透明串口
==============================begin

/**
 * @brief   注册读透明串口
 * @param   cb[in] 透明串口返回数据回调
 * @param   usrParam[in] 用户参数
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_RegReadTSerialCb(ZBX_ReadTSerialCb_t cb, int usrParam);

/**
 * @brief   注册读透明串口
 * @param   cb[in] 透明串口返回数据回调
 * @param   usrParam[in] 用户参数
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_RegReadTSerialCbEx(ZBX_Cam* cam, ZBX_ReadTSerialCb_t cb,
                                    int usrParam);

/**
 * @brief   获取透明串口参数
 * @param   cam[in]   相机句柄
 * @param   index[in] 第N路串口,目前只支持 ZBX_SERIA_RS485  ZBX_SERIA_RS232
 * @param   baudrate[out] 波特率
 * @param   baudrate:只能为以下值：1200, 2400, 4800, 9600, 14400, 19200, 38400,
 * 56000, 57600, 115200, 128000, 256000
 * @param   parity[out] 校验位。只能为0，无校验位
 * @param   databit[out] 数据位，只能为5，6，7，8
```

```
 * @param   stopbit[out] 停止位，只能为1，2
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_GetTSerial(ZBX_Cam* cam, int index, int* baudrate,
                                    int* parity, int* databit, int* stopbit);

/**
 * @brief   打开透明串口
 * @param   cam[in]   相机句柄
 * @param   index[in] 第N路串口,目前只支持 ZBX_SERIA_RS485  ZBX_SERIA_RS232
 * @param   baudrate[in] 波特率
 * @param   baudrate:只能为以下值：1200, 2400, 4800, 9600, 14400, 19200, 38400,
 * 56000, 57600, 115200, 128000, 256000
 * @param   parity[in] 校验位。0:none, 1:odd, 2:even, 3:mark, 4:space
 * @param   databit[in] 数据位，只能为5，6，7，8
 * @param   stopbit[in] 停止位，只能为1，2
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_OpenTSerial(ZBX_Cam* cam, int index, int baudrate,
                                     int parity, int databit, int stopbit);

/**
 * @brief   写透明串口
 * @param   cam[in]   相机句柄
 * @param   index[in] 第N路串口,目前只支持 ZBX_SERIA_RS485  ZBX_SERIA_RS232
 * @param   data[in] 数据
 * @param   size[in] 数据长度
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_WriteTSerial(ZBX_Cam* cam, int index,
                                      const unsigned char* data, int size);

/*
 * @brief   关闭透明串口
 * @param   cam[in]   相机句柄
 * @param   index[in] 第N路串口,目前只支持 ZBX_SERIA_RS485  ZBX_SERIA_RS232
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_CloseTSerial(ZBX_Cam* cam, int index);

//============================透明串口
==============================end

//============================相机参数
==============================begin
```

```
/**
 */
SDK_API void SDK_CALL ZBX_RebootCam(ZBX_Cam* cam);
// SDK_API void SDK_CALL ZBX_ResetCamParam(ZBX_Cam* cam);


/**
 * @brief   获取相机时间
 * @param   cam[in]   相机句柄
 * @param   time[in] Unix时间
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_GetSysTime(ZBX_Cam* cam, int* time);

/**
 * @brief   设置相机时间
 * @param   cam[in]   相机句柄
 * @param   year[in] 年份（如：2016）
 * @param   month[in] 月份（如：5）
 * @param   day[in] 日期（如：6）
 * @param   hour[in] 小时（如：14）
 * @param   minute[in] 分钟（如：35）
 * @param   second[in] 秒（如：20）
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_SetSysTimeEx(ZBX_Cam* cam, int year, int month, int day,
                                      int hour, int minute, int second);

/**
 * @brief   获取相机时间
 * @param   cam[in]   相机句柄
 * @param   year[in] 年份（如：2016）
 * @param   month[in] 月份（如：5）
 * @param   day[in] 日期（如：6）
 * @param   hour[in] 小时（如：14）
 * @param   minute[in] 分钟（如：35）
 * @param   second[in] 秒（如：20）
 * @return  0 成功
 * @return  <0 失败
 */
SDK_API int SDK_CALL ZBX_GetSysTimeEx(ZBX_Cam* cam, int* year, int* month,
                                      int* day, int* hour, int* minute,
                                      int* second);

//设置是否需要相机连接上的事件的回调通知
SDK_API void SDK_CALL ZBX_SetNotifyConnected(int notify);
```

```
/**
 * @brief 注册人脸到相机，支持单个id多张人脸 最大5张人脸图像
 * @param [IN] cam 要注册的相机句柄
 * @param [IN] faceID 人员标记，用于唯一标记注册的人脸
 * @param [IN] jpg 人脸图像路径数组，图像格式须为JPG
 * @param [IN] img_count 图像路径数量
 * @param [IN] picture_flags 下发图像到相机的标识，0表示不存图片到相机, >0
 * 表示存到相机的图片数量
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_AddJpgPaths(ZBX_Cam* cam, FaceFlags* faceID, char* jpg[],
                                     int img_count, int picture_Flags);

SDK_API int SDK_CALL ZBX_AddImagePaths(ZBX_Cam* cam, FaceFlags* faceID,
                                       char* jpg[], int img_count,
                                       int picture_flags);

/**
 * @brief 注册人脸到相机，支持单个id多张人脸
 * @param [IN] cam 要注册的相机句柄
 * @param [IN] type 注册类型， 0表示添加 1表示修改
 * @param [IN] faceID 人员标记，用于唯一标记注册的人脸
 * @param [IN] jpg 人脸图像路径数组
 * @param [IN] img_count 图像数量
 * @param [IN] picture_flags 下发图像到相机的标识，0表示不存图片到相机, >0
 * 表示存到相机的图片数量
 * @param [OUT] err_imgs 注册失败的人脸图像，图片编号对应imgs下的图片编号
 * @param [OUT] err_imgs_count 注册失败的人脸图像数量
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_JpgPathsEx(ZBX_Cam* cam, int type, FaceFlags* faceID,
                                    char* jpg[], int img_count,
                                    int picture_Flags, ErrorFaceImage* err_imgs,
                                    int* err_imgs_count);

/**
 * @brief 注册人脸到相机，支持单个id多张人脸 最大5张人脸图像
 * @param [IN] cam 要注册的相机句柄
 * @param [IN] faceID 人员标记，用于唯一标记注册的人脸
 * @param [IN] imgs 人脸图像，图像格式须为JPG
 * @param [IN] img_count 图像数量
 * @param [IN] picture_flags 下发图像到相机的标识，0表示不存图片到相机, >0
 * 表示存到相机的图片数量
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_AddJpgFaces(ZBX_Cam* cam, FaceFlags* faceID,
```

```
                                FaceImage* imgs, int img_count,
                                int picture_flags);

/**

 * @brief 注册人脸到相机，支持单个id多张人脸 最大5张人脸图像
 * @param  [IN] cam 要注册的相机句柄
 * @param  [IN] type 注册类型， 0表示添加 1表示修改
 * @param  [IN] faceID 人员标记，用于唯一标记注册的人脸
 * @param  [IN] imgs 人脸图像，图像格式须为JPG
 * @param  [IN] img_count 图像数量
 * @param  [IN] picture_flags 下发图像到相机的标识，0表示不存图片到相机, >0
 * 表示存到相机的图片数量
 * @param  [OUT] err_imgs 注册失败的人脸图像，图片编号对应imgs下的图片编号
 * @param  [OUT] err_imgs_count 注册失败的人脸图像数量
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_JpgFacesEx(ZBX_Cam* cam, int type, FaceFlags* faceID,
                                FaceImage* imgs, int img_count,
                                int picture_flags, ErrorFaceImage* err_imgs,
                                int* err_imgs_count);

/**
 * @brief 注册人脸到相机，注册图像尺寸要求大于100*100，注册人脸尺寸大于80*80
 * @param  [IN] cam 要注册的相机句柄
 * @param  [IN] faceID 人员标记，用于唯一标记注册的人脸
 * @param  [IN] jpg 人脸图像，图像格式须为JPG
 * @param  [IN] len jpg数据长度
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_AddJpgFace(ZBX_Cam* cam, FaceFlags* faceID,
                                const unsigned char* jpg, int len);

/**
 * @brief 注册人脸到相机，支持单个id多张人脸 最大5张人脸图像
 * @param  [IN] cam 要注册的相机句柄
 * @param  [IN] faceID 人员标记，用于唯一标记注册的人脸
 * @param  [IN] imgs 人脸图像，图像格式须为JPG
 * @param  [IN] img_count 图像数量
 * @param  [IN] picture_flags 下发图像到相机的标识，0表示不存图片到相机, >0
 * 表示存到相机的图片数量
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_AddFaces(ZBX_Cam* cam, FaceFlags* faceID,
                                FaceImage* imgs, int img_count,
                                int picture_flags);

/**
 * @brief 注册人脸到相机，支持单个id多张人脸 最大5张人脸图像
```

```
 * @param  [IN] cam 要注册的相机句柄
 * @param  [IN] type 注册类型， 0表示添加 1表示修改
 * @param  [IN] faceID 人员标记，用于唯一标记注册的人脸
 * @param  [IN] imgs 人脸图像，图像格式须为JPG
 * @param  [IN] img_count 图像数量
 * @param  [IN] picture_flags 下发图像到相机的标识，0表示不存图片到相机, >0
 * 表示存到相机的图片数量
 * @param  [OUT] err_imgs 注册失败的人脸图像，图片编号对应imgs下的图片编号
 * @param  [OUT] err_imgs_count 注册失败的人脸图像数量
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_FacesEx(ZBX_Cam* cam, int type, FaceFlags* faceID,
                                 FaceImage* imgs, int img_count,
                                 int picture_flags, ErrorFaceImage* err_imgs,
                                 int* err_imgs_count);

/**
 * @brief 删除一个人员信息
 * @param  [IN] cam 要注销的相机句柄
 * @param  [IN] personID  需要删除的人脸ID
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_DeleteFaceDataByPersonID(ZBX_Cam* cam,
                                                  const char* personID);

/**
 * @brief 删除所有人脸
 * @param  [IN] cam 要删除的相机句柄
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_DeleteFaceDataAll(ZBX_Cam* cam);

/**
 * @brief 获取已经添加的人脸中总数
 * @param  [IN] cam 相机句柄
 * @return 返回已经添加的人脸总数，返回值小于0表示出错，参考错误码
 */
SDK_API int SDK_CALL ZBX_GetFaceIDTotal(ZBX_Cam* cam);

/**
/**
 * @brief 通过人员角色查询
 * @param  [IN] cam 要查看的相机句柄
 * @param  [IN] role 要查询的人员角色 0：普通人员。 1：白名单人员。
 * 2：黑名单人员。 -1：所有人员。
 * @param  [IN] page_no 要查询的页码
 * @param  [IN] page_size 每页的数据条数，用于数据分页 最大值100
 * @param  [IN] featureFlags
 * 特征查询标记，是否查询特征信息，0表示需要，非0表示不需要
```

```
 * @param  [IN] imgFlags 是否查询人脸图像
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_QueryByRole(ZBX_Cam* cam, int role, int page_no,
                                     int page_size, char featureFlags,
                                     char imgFlags);

/**
 * @brief 人员查询扩展，支持按条件查询，以及模糊查询
 * @param  [IN] cam 要查看的相机句柄
 * @param  [IN] role 要查询的人员角色 0：普通人员。 1：白名单人员。
 * 2：黑名单人员。 -1：所有人员。
 * @param  [IN] page_no 要查询的页码
 * @param  [IN] page_size 每页的数据条数，用于数据分页 最大值100
 * @param  [IN] featureFlags
 * 特征查询标记，是否查询特征信息，0表示需要，非0表示不需要
 * @param  [IN] imgFlags 是否查询人脸图像
 * @param  [IN] condition_flag 用于标记按哪些条件查询，使用enum
 * ConditionFlag来控制，0：无效，非0：有效
 * @param  [IN] query_mode 0表示精确查询，非0表示模糊查询
 * @param  [IN] condition 查询条件
 * @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
 */
SDK_API int SDK_CALL ZBX_QueryFaceEx(ZBX_Cam* cam, int role, int page_no,
                                     int page_size, char featureFlags,
                                     char imgFlags, short condition_flag,
                                     short query_mode,
                                     QueryCondition* condition);

/**
* @brief  获取对比失败是否上传数据配置
* @param   cam[in]   相机句柄
* @param   //0：都上传，1：只上传对比成功的数据
* @return  0 获取成功
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_GetPushMode(ZBX_Cam* cam, char* mode);

/**
* @brief  设置对比失败是否上传数据配置
* @param   cam[in]   相机句柄
* @param   //0：都上传，1：只上传对比成功的数据
* @return  0 获取成功
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_SetPushMode(ZBX_Cam* cam, char mode);


/**
```

```
* @brief 设置语音配置
* @param  cam[in]  相机句柄
* @param  //0：自定义语音，1：固定语音1，2：固定语音2
* @return 0 获取成功
* @return <0 获取失败 参考错误码
*/
SDK_API int SDK_CALL ZBX_SetAudioConfig(ZBX_Cam* cam, ZBX_AudioConfig audioconfig);

/**
* @brief 获取语音配置
* @param  cam[in]  相机句柄
* @param  //0：自定义语音，1：固定语音1，2：固定语音2
* @return 0 获取成功
* @return <0 获取失败 参考错误码
*/
SDK_API int SDK_CALL ZBX_GetAudioConfig(ZBX_Cam* cam, ZBX_AudioConfig* audioconfig);

/**
* @brief 设置是否开启戴帽检测
* @param  cam[in]  相机句柄
* @param  //0：不开启，1：开启
* @return 0 获取成功
* @return <0 获取失败 参考错误码
*/
SDK_API int SDK_CALL ZBX_SetSafetyHat(ZBX_Cam* cam, int enable);

/**
* @brief 获取是否开启戴帽检测
* @param  cam[in]  相机句柄
* @param  //0：不开启，1：开启
* @return 0 获取成功
* @return <0 获取失败 参考错误码
*/
SDK_API int SDK_CALL ZBX_GetSafetyHat(ZBX_Cam* cam, int* enable);

/**
* @brief 设置是否开启戴帽通行
* @param  cam[in]  相机句柄
* @param  //0：不开启，1：开启 开启不带安全帽 则禁止通行
* @return 0 获取成功
* @return <0 获取失败 参考错误码
*/
SDK_API int SDK_CALL ZBX_SetSafetyHatStatus(ZBX_Cam* cam, int enable);

/**
* @brief 获取是否开启戴帽通行
* @param  cam[in]  相机句柄
* @param  //0：不开启，1：开启  开启不带安全帽 则禁止通行
* @return 0 获取成功
```

```
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_GetSafetyHatStatus(ZBX_Cam* cam, int* enable);

/**
* @brief  获取亮灯时段
* @param   cam[in]   相机句柄
* @param   CtrlTime[out]  亮度设置
* @return  0 获取成功
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_GetLedCtrlTime(ZBX_Cam* cam, ZBX_LedCtrlTime* CtrlTime);

/**
* @brief  设置亮灯时段
* @param   cam[in]   相机句柄
* @param   Level[in]  亮度等级
* @return  0 获取成功
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_SetLedCtrlTime(ZBX_Cam* cam, ZBX_LedCtrlTime CtrlTime);

/**
* @brief  设置 Http 推送配置
* @param   cam[in]   相机句柄
* @param   Level[in]  亮度等级
* @return  0 获取成功
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_SetHttpConfig(ZBX_Cam* cam, ZBX_HTTPCONFIG httpConfig);


/**
* @brief  获取 Http 推送配置
* @param   cam[in]   相机句柄
* @param   Level[in]  亮度等级
* @return  0 获取成功
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_GetHttpConfig(ZBX_Cam* cam, ZBX_HTTPCONFIG* httpConfig);

/*
* @brief   升级接口
* @param   系统文件路径
* @param   系统升级完成必须重启设备  或者调用接口ZBX_RebootCam
* @return 0 成功
*/
SDK_API int SDK_CALL ZBX_UpdateSystem(ZBX_Cam* cam, const char* filePth);
```

```
SDK_API void SDK_CALL ZBX_SetUpdateSystem_CB(ZBX_Cam* cam, ZBX_HTTPRESULT_PROCESS cb,
void* user_data);

/**
* @brief   注册人脸抓拍数据离线回调函数
* @param   cb[in] 回调函数指针
* @param   usrParam[in] 用户参数
* @return  无
*/
SDK_API void SDK_CALL ZBX_RegFaceOffLineRecoCb(ZBX_Cam* cam, ZBX_FaceRecoCb_t cb,
        void* usrParam);

/**
* @brief   屏幕熄屏时间获取
* @param   time[out] 时间s
* @param    //1 - 24 * 60 * 60,范围外为永不休眠
* @return  无
**/
SDK_API int SDK_CALL ZBX_GetExtinguishingScreenTime(ZBX_Cam* cam, int* time);

/**
* @brief   屏幕熄屏时间设置
* @param    time[in] 时间 单位：秒
* @param    //1 - 24 * 60 * 60,范围外为永不休眠
* @return  无
**/
SDK_API int SDK_CALL ZBX_SetExtinguishingScreenTime(ZBX_Cam* cam, int time);

/**
* @brief  水印开关设置
* @param    status[in]
* @param    //0:关闭，1：开启
* @return  无
**/
SDK_API int SDK_CALL ZBX_SetWaterMark(ZBX_Cam* cam, int status);

/**
* @brief   水印开关获取
* @param    status[in]
* @param    //0:关闭，1：开启
* @return  无
**/
SDK_API int SDK_CALL ZBX_GetWaterMark(ZBX_Cam* cam, int* status);

/**
* @brief  人脸跟踪框开关设置
* @param    status[in]
```

```
* @param    //0:关闭，1：开启
* @return  无
**/
SDK_API int SDK_CALL ZBX_SetFaceShow(ZBX_Cam* cam, int status);

/**
* @brief   人脸跟踪框开关获取
* @param    status[in]
* @param    //0:关闭，1：开启
* @return  无
**/
SDK_API int SDK_CALL ZBX_GetFaceShow(ZBX_Cam* cam, int* status);

/**
* @brief   注册韦根输入信息回调
* @param   cb[in] 回调函数指针
* @param   usrParam[in] 用户参数
* @return  无
*/
SDK_API void SDK_CALL ZBX_RegWgAlarmRecordCb(ZBX_Cam* cam,
        ZBX_WGAlarmRecordCb_t cb,
        void *usrParam);

/**
* @brief   注册身份证输入信息回调
* @param   cb[in] 回调函数指针
* @param   usrParam[in] 用户参数
* @return  无
*/
SDK_API void SDK_CALL ZBX_RegIDCardRecordCb(ZBX_Cam* cam,
        ZBX_IDCardRecordCb_t cb,
        void *usrParam);


/**
* @brief   进入厂测模式
* @ 请勿进入厂测模式 否则将断开与相机的连接
* @return  无
*/
SDK_API void SDK_CALL ZBX_EnterFactory(ZBX_Cam* cam);

/**
* @brief   进入老化模式
* @设备将全速运行 请谨慎进入
* @ 进入老化模式
* @return  无
*/
SDK_API int SDK_CALL ZBX_SetAging(ZBX_Cam* cam, int status);
```

```
/**
* @brief   停止老化模式
* @
* @return  无
*/
SDK_API int SDK_CALL ZBX_GetAging(ZBX_Cam* cam, int *status);


/**
* @brief   获取闸机继电器闭合持续时间
* @param   cam[in]   相机句柄
* @param   duration[out] 继电器闭合持续时间(500-5000ms)
* @return  0 获取成功
* @return  <0 获取失败
*/
SDK_API int SDK_CALL ZBX_GetAlarmDuration(ZBX_Cam* cam, int* duration);

/**
* @brief   设置闸机继电器闭合持续时间
* @param   cam[in]   相机句柄
* @param   duration[in] 继电器闭合持续时间(500-5000ms)
* @return  0 设置成功
* @return  <0 设置失败
*/
SDK_API int SDK_CALL ZBX_SetAlarmDuration(ZBX_Cam* cam, int duration);


/**
* @brief   获取闸机开闸类型
* @param   cam[in]  相机句柄
* @param   type[in]  //0：刷脸，1：刷卡，2：刷身份证，3：刷脸+刷卡
* @mode //4：刷脸+刷身份证，5：刷脸或刷卡，6：刷脸或刷身份证，7：过人开闸
* @return  0 获取成功
* @return  <0 获取失败
*/
SDK_API int SDK_CALL ZBX_GetGatewayControlType(ZBX_Cam* cam, int* type);

/**
* @brief   设置闸机开闸类型
* @param   cam[in]  相机句柄
* @param   type[in]  //0：刷脸，1：刷卡，2：刷身份证，3：刷脸+刷卡
* @mode //4：刷脸+刷身份证，5：刷脸或刷卡，6：刷脸或刷身份证，7：过人开闸
* @return  0 设置成功
* @return  <0 设置失败
*/
SDK_API int SDK_CALL ZBX_SetGatewayControlType(ZBX_Cam* cam, int type);

/**
* @brief   设置身份证对比准确性数据
```

```
* @return  无
*/
SDK_API int SDK_CALL ZBX_SetIDCardScore(ZBX_Cam* cam, int score);



/**
* @brief   获取身份证对比准确性数据
* @return  无
*/
SDK_API int SDK_CALL ZBX_GetIDCardScore(ZBX_Cam* cam, int *score);

/**
* @brief   查看过期自动清理开关
* @param   cam[in]   相机句柄
* @param   enable[out] 清理开关:关 1:开
* @return  0 获取成功
* @return  <0 获取失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_GetAutoCleanEnable(ZBX_Cam* cam, int* enable);

/**
* @brief   设置过期自动清理开关
* @param   cam[in]   相机句柄
* @param   enable[in] 清理开关，0:关 1:开
* @return  0 设置成功
* @return  <0 设置失败  参考错误码
*/
SDK_API int SDK_CALL ZBX_SetAutoCleanEnable(ZBX_Cam* cam, int enable);


/**
* @brief   设置相机点位编号
* @return  无
*/
SDK_API int SDK_CALL ZBX_SetDevName(ZBX_Cam* cam, const char* name);

/**
* @brief   获取相机点位编号
* @return  无
*/
SDK_API int SDK_CALL ZBX_GetDevName(ZBX_Cam* cam, char* name, int* nameSize);



/**
* @brief   设置用户权限
* @return  无
*/
```

```
SDK_API int SDK_CALL ZBX_SetUserRightsCfg(ZBX_Cam* cam, UserRightsCfgs cfg);


/**
* @brief   注册用户权限信息回调
* @param   cb[in] 回调函数指针
* @param   usrParam[in] 用户参数
* @return  无
*/
SDK_API void SDK_CALL ZBX_RegUserRightsCfgCb(ZBX_Cam* cam,ZBX_UserRightsCfgCb_t cb,void
*usrParam);


/**
* @brief   设置设备加密密码
* @return  无
*/
SDK_API int SDK_CALL ZBX_SetDeviceKey(ZBX_Cam* cam, DeviceKey devKey);

/**
* @brief   获取用户权限
* @return  无
*/
SDK_API void SDK_CALL ZBX_GetUserRightsCfg(ZBX_Cam* cam);

/*
* @brief   上传背景图接口
* @param   文件路径（图片名必须为）
* @param
* @return 0 成功
*/
SDK_API int SDK_CALL ZBX_UpdateBackgroud(ZBX_Cam* cam, const char* filePth);

/**
* @brief   恢复出厂界面
* @return  无
*/
SDK_API int SDK_CALL ZBX_SetBackGroudDefault(ZBX_Cam* cam);

/**
* @brief   更新屏幕显示界面
* @return  无
*/
SDK_API int SDK_CALL ZBX_UpdateBackGroudDefault(ZBX_Cam* cam);

/**
* @brief   设置群组
* @return  无
*/
```

```
SDK_API int SDK_CALL ZBX_SetGroupRec(ZBX_Cam* cam, Group_Rec grop_rec);

/**
* @brief  获取群组
* @return  无
*/
SDK_API int SDK_CALL ZBX_GetGroupRec(ZBX_Cam* cam, Group_Rec *grop_rec);

/**
* @brief  验证密钥
* @return  无
*/
SDK_API void SDK_CALL ZBX_SetAppKey(ZBX_Cam* cam, const char* key, int nsize);

/**
* @brief  获取相机工作模式
* @param   cam[in]  相机句柄
* @param   work_mode[out] 相机工作模式。//0：离线，1：在线，2：自动
* @return  0 获取成功
* @return  <0 获取失败
*/
SDK_API int SDK_CALL ZBX_GetCameraWorkMode(ZBX_Cam* cam,unsigned char* work_mode);

/**
* @brief  设置相机工作模式
* @param   cam[in]  相机句柄
* @param   work_mode[in] 相机工作模式。//0：离线，1：在线，2：自动
* @return  0 设置成功
* @return  <0 设置失败
*/
SDK_API int SDK_CALL ZBX_SetCameraWorkMode(ZBX_Cam* cam, unsigned char work_mode);


/**
*  @brief 修改人脸信息
*  @param  [IN] cam 要修改的相机句柄
*  @param  [IN] faceID 人员标记，用于唯一标记注册的人脸
*  @param  [IN] imgs 人脸图像，图像格式须为JPG
*  @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
*/
SDK_API int SDK_CALL ZBX_ModifyFaces(ZBX_Cam* cam, FaceFlags* faceID, FaceImage* imgs);



/**
*  @brief 修改语音文件
*  @param  [IN] cam 要修改的相机句柄
*  @param  [IN] filepath 文件的路径  目前只支持wav格式
*  @param  [IN] 需要上传的语音类型
```

```
* @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
*/
SDK_API int SDK_CALL ZBX_ModifyAudio(ZBX_Cam* cam, const char* filePth, VOL_TYPE ntype);

/**
* @brief 恢复语音文件
* @param [IN] cam 要修改的相机句柄
* @param [IN] 需要上传的语音类型
* @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
*/
SDK_API int SDK_CALL ZBX_ModifyFactoryAudio(ZBX_Cam* cam, VOL_TYPE ntype);


/**
* @brief 试听语音文件
* @param [IN] cam 要修改的相机句柄
* @param [IN] 需要上传的语音类型
* @return 返回值为0表示成功，返回负数表示失败，具体参考错误码
*/
SDK_API int SDK_CALL ZBX_TestAudio(ZBX_Cam* cam, VOL_TYPE ntype);


/**
* @brief 开始录像
* @param cam[in] 相机句柄
* @param sFileName[in] 录像文件路径 目前只支持录制avi
* @return 0 成功
* @return <0 失败
*/
SDK_API int SDK_CALL ZBX_SaveRealDate(ZBX_Cam* cam, char* sFileName);

/**
* @brief 停止录像
* @param cam[in] 相机句柄
* @return 0 成功
* @return <0 失败
*/
SDK_API int SDK_CALL ZBX_StopSaveRealDate(ZBX_Cam* cam);


/**
* @brief 获取相机gpio工作状态
* @param cam[in] 相机句柄
* @param state[out] gpio状态。 //0:自动，1：常闭，2：常开
* @return 0 获取成功
* @return <0 获取失败
*/
SDK_API int SDK_CALL ZBX_GetGpioWorkState(ZBX_Cam* cam, int* state);
```

```
/**
* @brief   设置相机gpio工作状态
* @param   cam[in]   相机句柄
* @param   state[in] gpio状态。 //0:自动，1：常闭，2：常开
* @return  0 获取成功
* @return  <0 获取失败
*/
SDK_API int SDK_CALL ZBX_SetGpioWorkState(ZBX_Cam* cam,int state);


/**
* @brief   获取人脸抓拍图片输出控制
* @param   cam[in]   相机句柄
* @param   ctl[out] 输出控制，//0:普通，1：大脸模式
* 16:调试图像（功能复选的时候，对对应数字做与操作）
* @return  0 获取成功
* @return  <0 获取失败
*/
SDK_API int SDK_CALL ZBX_GetOutputCtl(ZBX_Cam* cam, int* ctl);

/**
* @brief   设置设置识别距离
* @param   cam[in]   相机句柄
* @param   ctl[in] 输出控制，//0:普通，1：大脸模式
* 16:调试图像（功能复选的话，对应数字做或操作）
* @return  0 设置成功
* @return  <0 设置失败
*/
SDK_API int SDK_CALL ZBX_SetOutputCtl(ZBX_Cam* cam, int ctl);


/**
* @brief   获取人脸识别距离
* @param   cam[in]   相机句柄
* @param   ctl[out] 输出控制，//0:普通，1：大脸模式
* 16:调试图像（功能复选的时候，对对应数字做与操作）
* @return  0 获取成功
* @return  <0 获取失败
*/
SDK_API int SDK_CALL ZBX_GetRecoDist(ZBX_Cam* cam, double* min,double* max);

/**
* @brief   设置人脸识别距离
* @param   cam[in]   相机句柄
* @param   ctl[in] 输出控制，//0:普通，1：大脸模式
* 16:调试图像（功能复选的话，对应数字做或操作）
* @return  0 设置成功
* @return  <0 设置失败
*/
```

SDK_API int SDK_CALL ZBX_SetRecoDist(ZBX_Cam* cam, double min, double max);