Table of contents

1 Overview

This SDK is based on TCP protocol. This document describes the functions, interfaces and data types and sample code of each function in the SDK.

1.1 System Requirement

OS supported：
Linux 64：G++(4.4.2+)
win64/win32
Programming language：c, c++

1.2 Revision History

Date	Version	Revision
1. 2018.04.20	SDK v0.6.12	1) Fix bugs that are too large for the registration image and that sdk dies when querying 2) Increase the ability to get debug images 3) Add the ability to get the deletion progress 4) Increase the ability to take snapshots of the system 5) Add to get the SDK version 6) Increase the interface to get, set, and draw the detection area 7) Increase the configuration of the rs485 protocol 8) Adding a query extension interface is a support condition query 9) jpg compression quality increased from 60 to 95
2. 2018.05.03	SDK v0.6.13	1) Add the switch function of the video stream 2) Increase the ability to obtain normalized images 3) Increase version detection of the SDK and camera firmware

		4) Fix bugs that are not set by the outer net penetration switch 5) Solve the problem of slow detection 6) Fix bugs that cover the first ten minutes of the video when recording it 7) Upgrade to 0.6.13
3. 2018-5-10	SDK v0.6.14	1) Increase the video rotation angle control 2) Increase the quality configuration of the output image 3) Increase the time-out automatic cleanup configuration 4) Fix the problem of deleting personnel information, error code exception 5) Upgrade the version to 0.6.14
4. 2018-5-28	SDK v0.6.15	1) Face detection parameters from 0.80 to 0.95, the detection rate is higher 2) Two-way video display with dual-eye camera support 3) Add login authentication 4) Added similarity calculations that support feature values 5) Add the 485 configuration service switch 6) Add face contouring
5. 2018-6-8	SDK v0.7.0	1) Increase the camera's online offline mode control 2) Add the Wigan open gate interface 3) Add an extended open gate interface for offline offline mode 4) Add registration images that limit incomplete faces 5) Add the start time of the registration person's validity period
6.2018-7-10	SDK v0.7.1	1) Increase re-enrollment progress information acquisition 2) Increase the reading and writing of the check code (the customer does the second check) 3) Increase the saving of infrared images 4) Increase camera operating mode control (in/offline mode) 5) Add a live detection switch
7.2018-7-16	SDK v0.7.2	1) The registration speed has been optimized 2) Solved the registration timeout error caused by incorrect image serial number in

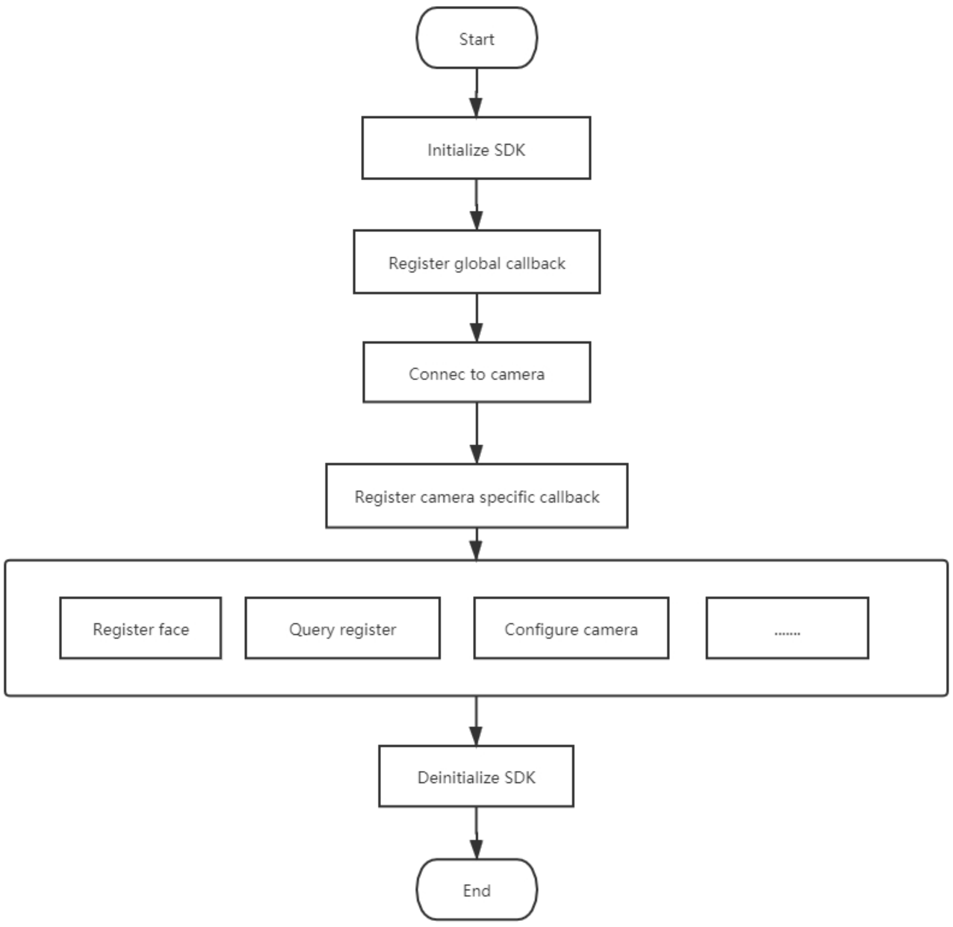
		the interface of the people registration with Ex 3) A personnel clusterer switch has been added
8.2018-8-22	SDK v0.7.3	1) Add a history query 2) Add logs for interface entry and exit 3) Increase the Gio input signal acquisition 4) Add the external display title configuration
9.2018-9-20	SDK v0.7.4	1) Increase the configuration of registered image synchronization parameters 2) Increase the face detection quality threshold setting 3) Add the face opener 4) Add the camera's built-in voice settings 5) The registration speed has been optimized
10.2018-10-18	SDK v0.7.4.1	1) Added queries to all ID interfaces
11.2018-11-13	SDK v0.7.5	1) Change how logs are written 2) Increase the acquisition of bicelyted camera image aerance information 3) Increase the get/set bicess image differential correction area 4) Increase the get setting distortion parameters 5) Add query normalized images 6) Increase server mode operation 7) Add the camera registration sync interface
12.2019-01-10	SDK v0.7.6	1) Increase the HTTP extranet penetration configuration 2) Add a two-pylon password verification 3) History query data adds role information 4) Face capture data adds device serial number and face feature point coordinate information 5) The length of the configuration parameters has been changed and the new configuration interface has been added 6) Change the camera firmware information data length to add a new query interface 7) Add a new network configuration parameter interface
13.2019-01-23	SDK v0.7.6.1	1) Added TTS
14.2019-02-25	SDK v0.7.6.2	Add more comparison modes Add a hard hat detection switch Add the setting LCD screen display item Add http heartbeat switch

		Add http domain name settings Adding filtering does not ratio the successful capture data switch Fix the rotation picture error registration BUG
15.2019-03-19	SDk v0.7.7	1) Increase WG66 long card number support 2) Increase the time-by-time scheduling of the opening of the floodgates 3) Add ID information 4) Add a functional authorization interface
16.2019-04-15	SDk v0.7.8	1) Add a remote access configuration interface 2) Add a remote upgrade server interface 3) Add a 4G status query interface 4) Add the camera Ping interface
17.2019-05-10	SDk v0.7.9	1) Increase the connection to the transit server 2) Add the log off switch 3) Increase TTS online playback
18.2019-06-19	SDk v0.8	1) Increase the capacity of registrars to query 2) Increase the number of crawl record strip queries 3) Increase access to Zhejiang Industrial Exchange platform 4) Capture data to increase face angle and ratio type 5) Add a new video stream callback interface to discard the original interface 6) Increase the registration picture legitimacy check interface 7) Modify the replay video to play slowly 8) Increase the sub-stream parameter setting 9) Add h264 frame number, channel number
19.2019-08-01	SDK v0.9	1) Add the name privacy display switch 2) Add wifi queries and connections 3) Add jpg pictures to register directly
20.2019-09-02	SDK v0.9.1	1) Increase the camera EMMC memory query 2) Increase EMMC memory formatting 3) Add sending the jason command to the camera 4) Reduce SDK memory footprint
20.2019-11-01	SDK v0.9.3	1) Modify the camera time-out BUG in the camera 2) The user name is expanded to 48 bytes 3) Increase reading QR code 4) Add sdk to transfer to each other through

		the camera 5) Increase the screensaver ratio to successfully display the item mirror switch 6) Increase the external network penetration domain name settings 7) Add NTP domain name proofing 8) Time scheduling support to 15 groups 9) Resolve the json library conflict
21.2019-11-27	SDK v0.9.4	1) Modify scheduling compatibility 2) Add VOCSC 3) Add read and write user logs 4) Add a single modification of face information
22.2020-1-03	SDK v0.9.5	1) Add the sip call interface 2) Increase the capture gps information
23.2020-2-04	SDK v0.9.6.1	1) Increase the body temperature information of the person 2) Increase the temperature opening limit 3) Add mask information 4) Add an open switch without a mask
24.2020-3-10	SDK v0.9.6.2	1) Add the AES encryption switch 2) Increase the ratio failure information 3) Add holiday settings
25.2020-5-20	SDK v0.9.7	1) Change the picture quality detection algorithm This version initialization needs to be imported into the model file 2) Windows platform dll compiles using vs2015 3) Resolve the failed bug
26.2020-10-20	SDK v0.10.0	1) Add a new extraction and normalization image interface to accommodate the new algorithm of EV200 2) Improve the visual talk interface

2 Guideline

2.1 API flow chart



Note:

1. **SDK initialization should be done once and only once.**
2. **Data passed into callback functions will be released after the callback is invoked, if the data is needed for further use, you should make a copy of it.**

3 Data Types

SDK version

```
structHaSdkVersion
{
charsdk_version[64];
charprotocl_version[64];
charsdk_code_version[64];
charmin_firmware_ver[64];
charalgorithm_version[64];
};
```

Field	Description:
char sdk_version	Sdk version
char protocl_version	Protocol version
charsdk_code_version	Sdk source code version
char min_firmware_ver	Min firmware version
charalgorithm_version	Algorithm version

Character Encoding

```
 typedef enum
{
char_ENCODE_GBK = 0,
char_ENCODE_UTF8
}char_ENCODE;
```

Field	Description
char_ENCODE_GBK	GBK Encoding
char_ENCODE_UTF8	UTF8 Encoding

Rectangle Region

```
structha_rect
{
    short x;
    short y;
    short width;
    short height;
};

struct HA_Point {
    int x;
    int y;
};
```

Field	Description
x	X-coordinate of upper left corner of rectangular region
y	Y-coordinate of upper left corner of rectangular region
width	The width of rectangular region
height	The height of rectangular region

See also:

```
structSRect
{
short x;
short y;
shortwidht;
short height;
};
```

System Version

```
structVERSION
{
charprotocl_version[64];
chararm_version[64];
chararm_buildtime[64];
chararm_code_version[64];
chardsp_version[64];
chardsp_buildtime[64];
chardsp_code_version[64];
unsignedshortvideo_width;
unsignedshortvideo_height;
};
```

Field	Description
charprotocl_version	Protocol version
chararm_version	Platform version
chararm_buildtime	Platform build time
chararm_code_version	Platform code version

chardsp_version	deprecated
chardsp_buildtime	deprecated
chardsp_code_version	deprecated
unsignedshortvideo_width	Original video width
unsignedshortvideo_height	Original video height

See also:
 SystemVersionInfo

Camera Handle

```
struct HA_Cam;
```

Note:
 Don’t change the value

See also:
 HA_ConnectEx
 HA_Connect

Camera System Time

```
structSystemTime
{
    chartime_zone;
    chardate[11];
    chartime[9];
    charresv[11];
};
```

Field	Description
chartime_zone	Time zone
char date	Date string: Format: year/month/day example: 2017/12/6
char time	Time string Format: hour:minute:second

	example:17:37:05
charresv	reserved

See also:

HA_GetSystemTime

Camera Parameter

```
structFaceAppParam
{
chardev_no[32];
charpoint_no[32];
charpoint_name[96];
char resv1[32];
unsignedcharheart_beat_interval;
charextranet_enale;
unsignedshortextranet_port;
charextranet_ip[16];
unsignedcharverify_enable;
charuser_name[15];
charpasswd[16];
unsignedchar resv2[12];
unsignedshortupload_mode;
unsignedshortupload_port;
charupload_ip[16];
Upload_Inforupload_info;
unsignedcharcam_mode;
char resv3[17];
int match_enable;
int match_score;
int dereplication_enable;
int dereplication_interval;
unsignedshortoutput_mode;
   charautothresholdScore;
   char resv4[255];
};
```

Field	Description
chardev_no	Device No.
charpoint_no	Position No.
charpoint_name	Position Name

char resv1	Reserved
unsignedcharheart_beat_interval	Heartbeat interval（default 5s）
charextranet_enable	Enable server
unsignedshortextranet_port	Server port
charextranet_ip	Server ip
unsignedcharverify_enable	Server login authentication 1: enable 0: disable
charuser_name[15]	User name
char passwd	Password
unsignedchar resv2	Reserved
unsignedshortupload_mode	Upload method: 0: disabled 1: TCP 2: FTP 3: HTTP
unsignedshortupload_port	Face Capture Upload Server Port
charupload_ip	Face Capture Upload Server IP
Upload_Inforupload_info	Face Capture Upload Union
unsignedcharcam_mode	Work Mode 1: Auto Mode Switch to offline mode when the mode camera is connected and automatically switch to online mode when it is not connected 2: Online Mode Gate access is controlled by pc 3: Offline Mode Gate access is controlled by camera
char resv3	Reserved
int match_enable	Matching Enable
int match_score	Matching Score
int dereplication_enable	Duplication Detect

	Enable: Open gate for the same person only once in a given time period
int dereplication_interval	Time period for duplication detect
unsignedshortoutput_mode	Image output mode: 0: disable 1: full image 2: close-up image 4: matching template 8: feature data
charautothresholdScore	Dynamic threshold in automatic threshold mode
char resv4	reserved

Note:
output_mode, Bitwise OR operator is supported

See also:

Upload_Infor
HA_GetFaceAppParam
HA_SetFaceAppParam

Substream Encoding Parameter

```
struct SubCodParam{
    unsigned charsubcod_enable;
    unsigned charsubcod_pic_size;
    unsigned intsubcod_rate;
    unsigned charsubcod_rcMode;
    unsigned charsubcod_frame_rate;
    char res[16];
};
```

Field	Description
subcod_enable	Substream switch

	disable by default 0: disable non-zero: enable
subcod_pic_size	Substream resolution 1:640 * 360 2:720 * 576 3:720 * 480 4:1280 * 720
subcod_rate	Substream bitrate (kbps) 1024 by default Average bitrate in constant bitrate mode Max bitrate in variable bitrate mode
subcod_rcMode	Substream bitrate mode 0 by default 0: constant bitrate mode Non-zero: variable bitrate mode
subcod_frame_rate	Substream frame rate av200 1~25 default to 25 cv500 1~30 default to 30

See also:

HA_SetSubCodParam

HA_GetSubCodParam

Gpio Input Method

```
enumGpioInType{
   IN_STATE_WG26  = 0,
   IN_STATE_WG34  = 1,
   IN_STATE_IO = 255
};
```

Field	Description
IN_STATE_WG26	Wiegand 26
IN_STATE_WG34	Wiegand 34
IN_STATE_IO	IO input mode

Data Upload Setting

```
union Upload_Infor {
    charupload_url[102];
    structFtpInfoftp_info;
    chartcp_resv[102];
    charother_resv[102];
};
```

Field	Description
charupload_url	http upload URL
structFtpInfoftp_info	ftp upload parameter
chartcp_resv	tcp upload reserved byte
charother_resv	reserved

```
struct FtpInfo{
    char user_name[15];
    char passwd[15];
    char path[70];
    char resv[2];
};
```

Field	Description
char user_name	user name
char passwd	password
char path	upload path
char resv	reserved

Note:
It is recommended to use ClientParam to set upload parameter

Data Upload Parameter

FTP Upload

```
structFtpClientParam
{
   charip[16];
   int port;
   charuser[15];
   charpassword[15];
   charpattern[70];
};
```

Field	Description
charip	ip address
int port	port number
char user	User name
char password	Password
char pattern	Upload destination directory

TCP Upload

```
struct TcpClientParam
{
   char ip[16];
   int port;
   char enable;
   unsigned char enable_verify;
   char user_name[16];
   char passwd[17];
   char resv[65];
};
```

Field	Description
charip	Ip address
int port	Port number
char enable	Deprecated
charenable_verify	Enable verification
char user_name	User name
char passwd	Password
char resv	reserved

HTTP Upload

```
structHttpClientParam {
    charip[16];
    unsignedshort port;
    charurl[102];
};
```

Field	Description
charip	Ip address
unsignedshort port	Port number
charurl	url

Upload Method

```
structClientParam{
    char mode;
    char enable_heart;
    charresv[2];
    union {
        FtpClientParam ftp;
        TcpClientParam tcp;
        HttpClientParam http;
        HttpClientParamweb_service;
    };
};
```

Field	Description
char mode	Upload method 0: disable upload 1: TCP upload 2: FTP upload 3: HTTP upload
char enable_heart	Heatbeat(http only)
charresv	reserved
FtpClientParam ftp	ftp upload parameters
TcpClientParam tcp	tcpupload parameters
HttpClientParam http	Http upload parameters

HttpClientParamweb_se rvice	reserved

See also:
- HA_GetUploadConfig
- HA_SetUploadConfig

Real-time Face Capture Info

```
structFaceRecoInfo
{
    unsignedintsequence;
    charcamId[32];
    charposId[32];
    charposName[96];
    unsignedinttvSec;
    unsignedinttvUsec;
    shortisRealtimeData;
    short matched;
    charmatchPersonId[20];
    charmatchPersonName[16];
    int matchRole;
    int existImg;
    charimgFormat[4];
    int imgLen;
    unsignedshortfaceXInImg;
    unsignedshortfaceYInImg;
    unsignedshortfaceWInImg;
    unsignedshortfaceHInImg;
    int existFaceImg;
    charfaceImgFormat[4];
    int faceImgLen;
    unsignedshortfaceXInFaceImg;
    unsignedshortfaceYInFaceImg;
    unsignedshortfaceWInFaceImg;
    unsignedshortfaceHInFaceImg;
    int existVideo;
    unsignedintvideoStartSec;
    unsignedintvideoStartUsec;
    unsignedintvideoEndSec;
    unsignedintvideoEndUsec;
```

```
charvideoFormat[4];
int videoLen;
unsignedchar sex;
unsignedchar age;
unsignedchar expression;
unsignedcharskinColour;
unsignedcharqValue;
unsignedcharsourceOfReg;
charattributeOfReg;
unsignedchar living;
char hatColour;
char FaceAngle;
char FaceAngleFlat;
char is_encry;
unsigned int math_type;
unsigned intwgCardNO;
unsigned long longwgCardNOLong;
charGPSN[16];
charGPSE[16];
unsigned intGPSNum;
int match_failed_reson;
char resv[55];
char has_mask;
float temperature;
unsignedchar* img;
unsignedchar* faceImg;
unsigned char* video;
int feature_size;
float *feature;
int modelFaceImgLen;
charmodelFaceImgFmt[4];
unsignedchar *modelFaceImg;
 HA_Point PointInImg[5];
 HA_Point PointInFaceImg[5];
char dev_id[32];
int existIDCard;
char IDCardnumber[36];
char IDCardname[43];
unsigned charIDCardsex;
char IDCardnational[19];
char IDCardbirth[17];
char IDCardresidence_address[103];
char IDCardorgan_issue[43];
char IDCardvalid_date_start[17];
```

```
  char IDCardvalid_date_end[17];
  char userParam[68];
  char matchPersonNameEx[64];
  char matchPersonIDEx[64];
  unsigned charperson_name_aes_len;
  unsigned charperson_id_aes_len;
};
```

Field	Description
unsignedint sequence	1-based sequence number for every capture, each capture will add the number by 1, rebooting device resets the number to 1
charcamId	Camera id
charposId	Position id
charposName	Position name
unsignedinttvSec	Capture time, elapsed seconds from epoch time
unsignedinttvUsec	Microseconds of capture time
shortisRealtimeData	Value indicating whether the data is realtime data. 0: non-realtime Non-zero: realtime
short matched	Match result: 0: match not performed -1: match failed Any value greater than0: confidence score for a successful match out of hundred mark system
charmatchPersonId	person ID
charmatchPersonName	Person name
intmatchRole	Person category. 0: normal 1: white name 2: black name
intexistImg	Value indicating whether include full image

	0: no include Non-zero: include
charimgFormat	Full image format
intimgLen	Size of full image in bytes
unsignedshortfaceXInImg	X-coordinate of face in full image
unsignedshortfaceYInImg	Y-coordinate of face in full image
unsignedshortfaceWInImg	Width of face
unsignedshortfaceHInImg	Height of face
intexistFaceImg	Value indicating the existencethe of face close-up 0: not include Non-zero: include
charfaceImgFormat	Image format of the face close-up
intfaceImgLen	Size of the face close-up in bytes
unsignedshortfaceXInFaceImg	x-coordinate of face inside the face close-up。
unsignedshortfaceYInFaceImg	y-coordinate of the face inside the face close-up
unsignedshortfaceWInFaceImg	Width of face inside the face close-up
unsignedshortfaceHInFaceImg	Height of face insidethe face close-up
intexistVideo	Value indicating whether include video 0: not include Non-zero: include
unsignedintvideoStartSec	Start time of video, seconds from epoch time
unsignedintvideoStartUsec	Microseconds of start time
unsignedintvideoEndSec	End time of video, seconds from epoch time
unsignedintvideoEndUsec	Microseconds of end time
charvideoFormat	Video format

intvideoLen	size of video in bytes
unsignedchar sex	Gender 0: n/a 1: male 2: female
unsignedchar age	age 0: n/a Other value: age
unsignedchar expression	Facial expression 0: n/a Other value: to be done
unsignedcharskinColour	Skin color 0: n/a Other value: to be done
unsignedcharqValue	Quality of image for registration, the higher the quality, the better for registration
unsignedcharsourceOfReg	Source of registration 0: n/a 1: registered by an app 2: auto registered 3: registered by cloud synchronization
unsigend char living	Value indicating liveness detection result 0: n/a 1 : liveness detect succeed 2 : liveness detectfailed
char hatColour	Color of safety helmet 0: n/a 1: blue 2: orange 3: red 4: white 5: yellow
char FaceAngle	Reserved
char FaceAngleFlat	Reserved
int match_failed_reson	Value indicating wherher name and id is

	encrypted 0: not encrypted 1: encrypted
unsigend intmath_type	Match type 0: n/a 1: face recognition 2: authentication 4: id card 8: Wiegand card 16： any person (face detected)
unsigned intwgCardNO	32 bitwiegand number
unsigned long longwgCardNOLong	46 bitwiegand number
charGPSN[16]	Gps longitude
charGPSE[16]	Gps latitude
unsigned intGPSNum	Gps satellite number
unsignedchar* video	reserved
int match_failed_reson	Matching faile reason See enum MatchFailedReasons
char has_mask	Value indicating mask wearing 0: n/a 1:weared 2: not weared
float temperature	Temperature
intfeature_size	Feature size
float *feature	The feature data
intmodelFaceImgLen	Face image length
HA_PointPointInImg	Feature point coordinate 5-element array in full image
HA_PointPointInFaceImg	Feature point coordinate 5-element array in close-up
char dev_id	Device id
int existIDCard	Value indicating existence of id info
char IDCardnumber	Id number from id card

char IDCardname	Name from id card
unsigned char IDCardsex	Gender from id card
char IDCardnational	Nationality from id card
char IDCardbirth	Date of birth from id card
char IDCardresidence_address	Address from id card
char IDCardorgan_issue	Issuing authority of id card
char IDCardvalid_date_start	Valid from date of id card
char IDCardvalid_date_end	Valid to date of id card
char userParam	User defined parameter
char matchPersonNameEx	Extended user name, when the user name is not long enough
char matchPersonIDEx	Extended face id
unsigned char person_name_aes_len	Name length after encrypted
unsigned char person_id_aes_len	Face id length after encrypted

Note:
Real-time face capture data is passed through callback function

See also:
HA_RegFaceRecoCb
HA_GetOutputCtl

NTP Info

```
structNtpInfo
{
    shortenable_state;
    shortupdate_cycle;
    charntp_server[16];
    charresv[12];
};
```

Field	Description
shortenable_state	Value indicating whether ntp is enabled 0: disabled 1:enabled
shortupdate_cycle	Update frequence in seconds (60-600s)
charntp_server	NTP server ip address
charresv	reserved

See also:
HA_SetNtpConfigEx

Network Parameter

```
structSystemNetInfoEx
{
    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];
};
```

Field	Description
char mac	Mac
charip	Ip
char netmask	Netmask
char gateway	Gateway
char manufacturer	Manufacturer
char platform	Platform
char system	System name
char version	Version

char ip_2	Ip address for network interface 2
char netmask_2	Netmask for network interface 2
char dns	Dns server
char dhcp_enable	DHCP enabled
char resv	reserved

See also:

HA_SetNetConfig
HA_GetNetConfig

Login Authentication Parameter

```
structAuthParam
{
    unsignedchar enable;
    charuser_name[16];
    charpasswd[17];
    charresv[14];
   };
```

Field	Description
unsignedchar enable	Value indicating whether enable authentication
charuser_name	User name
char passwd	Password
charresv	Reserved

See also:

HA_SetAuthInfo
HA_GetAuthInfo

Built-in Audio

```
structAudioItem
{
    int id;
    chardesc[64];
};
```

Field	Description
int id	Id
char desc	Description of the audio

See also:
built-in audio setting

Registration Info

```
structFaceFlags
{
    charfaceID[20];
    charfaceName[16];
    int role;
    unsignedintwgCardNO;
    unsignedinteffectTime;
    unsignedinteffectStartTime;
    short version;
    unsigned long longwgCardNOLong;
    unsigned charScheduleMode;
    char resv2;
    char userParam[68];
    char faceNameEx[64];
    charresv[8040];
};
```

Field	Description
charfaceID	Id
charfaceName	Name
int role	Category 0: normal 1: whitelisted 2: blacklisted
unsignedintwgCardNO	Wiegand card number
unsignedinteffectTime	Valid to time, seconds from epoch time 0xFFFFFFFF: never expire 0: expired
unsignedinteffectStartTime	Valid from time, seconds from epoch time
short version	The version of feature data
unsigned long	Extended Wiegand car

longwgCardNOLong;	dnumber, in case of the wgCardNO is not long enough,to enable this field, set wgCardNO to 0
unsigned charScheduleMode;	Gate access control rule 0: don’t use access rule 1-5: specific rule, seeKindSchedule->ScheduleNameCode
charuserParam	User defined parameter
char faceNameEx	Extended user name, when the user name is longer than 15 bytes
resv	reserved

See also:

Register face

Face Image

```
structFaceImage
{
    int img_seq;
    int img_fmt;
    unsignedchar *img;
    int img_len;
    int width;
    int height;
};
```

Field	Description
int img_seq	Index of image
int img_fmt	Format of image 0: jpg 、bmp、png 1: bgr data
unsignedchar *img	Image data
int img_len	Image data length
int width	Width of image, only for bgr data
int height	Height of image, only for bgr data

Note:
  When the struct is used as an Out parameter, fmt == 0 means jpg format

See also:
  Register face

Multi-Image Registration Error

```
structErrorFaceImage
{
    int img_seq;
    int err_code;
};
```

Field	Description
int img_seq	Index of failed image
int err_code;	Error code, see Error Code

See also:
  Register Face

Face Feature

```
structFaceFeature
{
    float *feature;
    shortfeatureseize;
    shortfeatureNum;
};
```

Field	Description
float *feature	The feature data array pointer, total length = featureseize*featureNum
shortfeatureseize;	Size of a single feature
shortfeatureNum	The count of feature

Note:
  Feature data is passed through callback function

See also:

Register Face

HA_RegFaceQueryCb

Face Query Info

```
structQueryFaceInfo
{
    int record_count;
    int record_no;
    charpersonID[20];
    charpersonName[16];
    int role;
    shortfeature_count;
    shortfeature_size;
    float *feature;
    int imgNum;
    int imgSize[5];
    charimgFmt[5][4];
    unsignedchar *imgBuff[5];
    unsignedintwgCardNO;
    unsignedinteffectTime;
    unsignedinteffectStartTime;
    short version;
    unsigned charScheduleMode;
    charresv;
    int twistImgNum;
    shorttwistwidth[5];
    shorttwistheight[5];
    int twischannel[5];
    char *twistimgBuff[5];
    unsigned long longwgCardNoLong;
    char userParam[68];
    char faceNameEx [64];
    char resv1[372];
};
```

Field	Description
int record_count	Count of matches
int record_no	Non-zero: Current record number

	(ranging from 1 to record_count) 0: no more record
charpersonID	Person ID
charpersonName	Person name
int role	Person category 0: normal 1: white name 2: black name
shortfeature_count	Count of feature data
shortfeature_size	Size of single feature data
float *feature	Pointer to the feature data, size is feature_count*feature_size
int imgNum	Count of template image max: 5
int imgSize	Images size array , imgSize[i] is the size of the ith image
charimgFmt	Image format array, imgFmt[i] is the format of ith image
unsignedchar *imgBuff	Image buffer array, imgBuff[i] is the address of ith image buffer
unsignedintwgCardNO	Wiegand card number
unsignedinteffectTime	Valid to, seconds from epoch time (January 1, 1970, 00:00) 0xFFFFFFFF: never expire 0: expired
unsignedinteffectStartTime	Valid from, seconds from epoch time (January 1, 1970, 00:00) 0: not initialized
short version	Non-zero: current version of feature data 0: not supported
unsigned charScheduleMode;	Scheduling rule 0: n/a 1~5 : correspond toKindSchedule->Schedul

	eNameCode
charresv	reserved
int twistImgNum	Normalized image count
shorttwistwidth	Width of normalized image
shorttwistheight	Height of normalized image
int twischannel	Channel of normalized image
char *twistimgBuff	Normalized image buffer Image buffer size: width*height*channel
unsigned long longwgCardNoLong	Supplement wiegand card number in the case of wgCardNO is not long enough
char userParam	User defined data
char faceNameEx	Supplement name in the case of personName is not long enough
char resv1	reserved

Note:
This struct is passed through call back function

See also:
HA_RegFaceQueryCb

Registration Query

```
structQueryCondition
{
    charfaceID[20];
    charfaceName[16];
    unsignedintwgCardNO;
    unsignedinttimeStart;
    unsignedinttimeEnd;
    unsignedinttime1Start;
    unsignedinttime1End;
    unsigned long longwgCardNOLong;
    char faceNameEx [64];
    unsignedcharresv[180];
};
```

Field	Description
charfaceID	Person ID
charfaceName	Person name
unsignedintwgCardNO	Wiegand card number
unsigned int timeStart	Lower bound of valid to range
unsignedinttimeEnd	Upper bound of valid to range
unsignedinttime1Start	Lower bound of valid from range
unsignedinttime1End	Upper bound of valid from range
unsigned long longwgCardNOLong	Supplement wiegand card number, in case of wgCardNO is not long enough, in which case wgCardNO is set to 0
char faceNameEx	Supplement person name in caseoffaceName is longer than 15 bytes
unsignedcharresv	reserved

```
typedef enum{
    QUERY_BY_ID = 0x1,
    QUERY_BY_NAME = 0x2,
    QUERY_BY_WGNO = 0x4,
    QUERY_BY_EFFECT_TIME = 0x8,
    QUERY_BY_EFFECT_START_TIME = 0x10
}ConditionFlag;
```

Field	Description
QUERY_BY_ID	Query by id
QUERY_BY_NAME	Query by name
QUERY_BY_WGNO	Query by Wiegand card number
QUERY_BY_EFFECT_TIME	Query by valid to range
QUERY_BY_EFFECT_START_TIME	Query by valid from range

Record Query

Record Query Parameter

```
structRecordCondition{
    charimg_flag;
    charreg_img_flag;
    char resv1[2];
    unsignedshortquery_mode;
    unsignedshortcondition_flag;
    unsignedinttime_start;
    unsignedinttime_end;
    shortscore_start;
    shortscore_end;
    unsignedchar sex;
    unsignedcharage_start;
    unsignedcharage_end;
    char resv2[17];
    charperson_id[20];
    charperson_name[16];
    unsignedcharqvalue_start;
    unsignedcharqvalue_end;
    charupload_state;
    char resv3[65];
};

enumRecordQueryFlag{
    RECORD_QUERY_FLAG_TIME              = 0x1,
    RECORD_QUERY_FLAG_SCORE             = 0x1 << 1,
    RECORD_QUERY_FLAG_SEX               = 0x1 << 2,
    RECORD_QUERY_FLAG_AGE               = 0x1 << 3,
    RECORD_QUERY_FLAG_ID                = 0x1 << 4,
    RECORD_QUERY_FLAG_NAME              = 0x1 << 5,
    RECORD_QUERY_FLAG_QVALUE            = 0x1 << 6,
    RECORD_QUERY_FLAG_UPLOAD            = 0x1 << 7,
    RECORD_QUERY_FLAG_SEQUENCE          = 0x1 << 8
};
```

Field	Description
charimg_flag	Value indicating whether include captured image in the output

	0: don’t include Non-zero: include
charreg_img_flag	Value indicating whether include registration image in the output 0: don’t include Non-zero: include
char resv1	reserved
unsignedshortquery_mode	Query mode 0: match exactly None-zero: fuzzy match
unsignedshortcondition_flag	Value indicating which field is valid for the query See enum RecordQueryFlag
unsignedinttime_start	Lower bound of capture time range
unsignedinttime_end	Upper bound of capture time range
shortscore_start	Lower bound of match score
shortscore_end	Upper bound of match score
unsignedchar sex	Gender
unsignedcharage_start	Lower bound of age
unsignedcharage_end	Upper bound of age
char resv2	reserved
charperson_id	Person ID
charperson_name	Person name
unsignedcharqvalue_start	Lower bound of image quality
unsignedcharqvalue_end	Upper bound of image quality
charupload_state	Upload flag 1: uploaded 0: not uploaded
char resv3	reserved

See also:

HA_QueryFaceRecord

Record Data

```
structRecordData{
   int record_count;
   int record_no;
   unsignedintsequence;
   unsignedinttvSec;
   unsignedinttvUsec;
   short matched;
   unsignedchar sex;
   unsignedchar age;
   charperson_id[20];
   charperson_name[16];
   int face_image_len;
   unsignedchar* face_image;
   unsignedshortfaceXInFaceImg;
   unsignedshortfaceYInFaceImg;
   unsignedshortfaceWInFaceImg;
   unsignedshortfaceHInFaceImg;
   int reg_image_len;
   unsignedchar* reg_image;
   unsignedcharqvalue;
   charis_upload;
   char role;
   unsigned char aes_enable;
   unsigned int match_type;
   char customer_txt[64];
   charperson_name_ext[64];
   float temperature;
   int match_failed_reson;
   char person_id_aes[64];
   unsigned char person_id_aes_len;
   unsigned char person_name_aes_len;
};
```

Field	Description
int record_count	Count of matches
int record_no	Non-zero: Current record number (1 to record_count) 0: no more record
unsignedintsequence	Sequence number

unsignedinttvSec	Capture time, Seconds from epoch time
unsignedinttvUsec	Microseconds of capture time
short matched	The score of matching
unsignedchar sex	Gender
unsignedchar age	Age
charperson_id	Face id
charperson_name	Person name
int face_image_len	Size of face close-up in bytes
unsignedchar* face_image	Format of face close-up
unsignedshortfaceXInFaceImg	x-coordinate of face inside the face close-up
unsignedshortfaceYInFaceImg	y-coordinate of the face inside the face close-up
unsignedshortfaceWInFaceImg	Width of the face
unsignedshortfaceHInFaceImg	Height of the face
int reg_image_len	Size of matched image 0: not matched
unsignedchar* reg_image	Format of matched image Null: not matched
unsignedcharqvalue	The quality of image
charis_upload	Value indicating upload 1: uploaded 0: not uploaded
char role	Category of the matched person (applicable only if matched > 0) 0: normal 1: whitelisted name 2: blacklisted name -2: n/a
charaes_enable	Value indicating if id and name is encrypted
unsigned intmatch_type	Match type
char customer_txt	User-defined data
charperson_name_ext	Extended person name (applicable only if encryption is enabled)

float temperature	temperature
int match_failed_reson	Match failed reason See enum MatchFailedReasons
char person_id_aes	Encrypted person id
unsigned char person_id_aes_len	Length of encrypted id
unsigned char person_name_aes_len	Length of encrypted name

See also:

HA_RegFaceRecordQueryCb

HA_QueryFaceRecord

Device Info for Server

```
structDeviceInfor{
    char dev_id[32];
    charip[32];
    charcamId[32];
    charposId[32];
    charposName[96];
};
```

Field	Description
char dev_id	Device id
charip	IP
charcamId	Camera id
charposId	Position id
charposName	Position name

Face Rectangle

```
structFaceRect
{
    unsignedintfaceId;
    structha_rectfaceRect;
    charresv[4];
};
```

Field	Description
unsignedintfaceId	Face id
structha_rectfaceRect	Face bound
charresv	Reserved

Note:
This struct is passed through callback function, it is designed for debugging.
Debug mode must be enabled.

See also:
- HA_RegFaceRectCb
- HA_SetDebugEnable

Debug Image

```
structDebugImage
{
    int format;
    short width;
    short height;
    int imageLen;
    unsignedchar *imgData;
};
```

Field	Description
int format	The debug image format 0:rgb data 1:jpg
short width	Width of image
short height	Height of image
int imageLen	The length of image data
unsignedchar *imgData	Image data

```
structDebugImageInfo
{
    unsignedinttimeStamp_s;
    unsignedinttimeStamp_u;
    int matched;
    int matchScore;
    charfaceId[20];
    int imageNum;
    structDebugImagedebugImage[6];
  };
```

Field	Description
unsignedinttimeStamp_s	Timestamp
unsignedinttimeStamp_u	Microsecond of timestamp
int matched	Match result 0: match not performed -1: match failed 1: match succeed
int matchScore	Match confidence score
charfaceId	Face id, if match succeed (matched == 1)
int imageNum	Number of debug image, max 6
structDebugImagedebugImage	Debug image info

Infrared Image Debug Info

```
structBebugInfraredImage
{
    int lived;
    unsignedinttimeStamp_s;
    unsignedinttimeStamp_u;
    shortx_deviations;
    shorty_deviations;
    ha_rectnormalImgeRect;
    ha_rectinfraredImgeRect;
    unsignedintviewImgeNum;
    DebugImage viewImge[8];
    unsignedintcloseupImgeNum;
    DebugImagecloseupImge[8];
};
```

Field	Description
int lived	0: non-live 1: live
unsignedinttimeStamp_s;	Timestamp

unsignedinttimeStamp_u	Microseconds of timestamp
shortx_deviations	x-coordinate of image deviations
shorty_deviations	y-coordinate of image deviations
ha_rectnormalImgeRect	Face bound for normal lighting
ha_rectinfraredImgeRect	Face bound for infrated lighting
unsignedintviewImgeNum	Number of full image
DebugImage viewImge	Full image data
unsignedintcloseupImgeNum	Number of face close-up
DebugImagecloseupImge	Close-up image data

Registration Delete Progress

```
structFaceDelProgressInfo
{
    int delCount;
    int curDelNo;
    charfaceId[20];
};
```

Field	Description
int delCount	Count of total faces that is being deleted
int curDelNo	Current progress of deletion
charfaceId	Current face id being deleted

Note:
Progress is reported through callback function

See also:

HA_RegFaceDeleteProgressCb
Delete Registered Face

Snapshot Image

```
structSnapshotImage
{
    unsignedinttimeStamp_s;
    unsignedinttimeStamp_u;
    int snapImageSize;
    unsignedchar *snapImage;
};
```

Field	Description
unsignedinttimeStamp_s	Timestamp of the snapshot
unsignedinttimeStamp_u	Microseconds of timestamp
int snapImageSize	Length of snapshot
unsignedchar *snapImage	Snapshot image

Note:
  The struct is passed through callback function

See also:
  HA_RegSnapshotCb
  HA_Snapshot

Gate Opening Record

```
structAlarmInfoRecord
{
    charcameraID[32];
    charalarmTime[20];
    unsignedcharalarmDeviceType;
    charresv[3];
    charpersonID[20];
    unsignedintalarmDeviceId;
    unsigned long longwgCardNOLong;
};
```

Field	Description
charcameraID	Camera id
charalarmTime	Gate open time string Format : 2018/3/6

	16:38:20
unsignedcharalarmDevice Type	Device type 0: electric relay 1: Wiegand device
charresv	Reserved
charpersonID	Face id
unsignedintalarmDeviceI d	Device id If device is electricrelay, the GPIO index; If the device is Wiegand device, the Wiegand card number
unsigned long longwgCardNOLong;	Extended Wiegand card number, when alarmDeviceId is not long enough

Note:
The struct is passed through callback function

See also:
HA_RegAlarmRecordCb

Gate Open Request

```
    structAlarmRequest
    {
        charcameraID[32];
        charpersonID[20];
        charrequestTime[20];
        unsignedcharalarmDeviceType;
        unsignedcharalarmDeviceState;
        charresv[2];
        unsignedintalarmDeviceId;
        unsigned long longwgCardNOLong;
};
```

Field	Description
charcameraID	Camera id
charpersonID	Face id

charrequestTime	Request time. Format:2018/3/6 16:38:20
unsignedcharalarmDeviceType	Device type. 0: electric relay 1: Wiegand device
unsignedcharalarmDeviceState	Current state of thedevice. 0: not enabled 1: enabled
charresv	Reserved
unsignedintalarmDeviceId	Device id If device is electric relay, the GPIO index; If the device is Wiegand device, the Wiegand card number
unsigned long longwgCardNOLong;	Extended Wiegand card number, when alarmDeviceId is not long enough

Note:
Device in online mode doesn't open gate automatically, it sends open gate request instead.
The data is passed through callback function.

See also:
HA_RegAlarmRequestCb
HA_GetCameraWorkMode

Reregistration Progress

```
struct FaceReRegistProgressInfo
{
    int regist_count;
    int cur_regist_no;
    charfaceId[20];
};
```

Field	Description

int regist_count	Total count
int cur_regist_no	Current index
charfaceId	Current face id

Note:
During the process of reregistration, the camera stops working. Meanwhile, don’t operate the camera.

See also:
HA_RegFaceReRegistProgressCb

Camera IP Query

```
structipscan_t
{
    charmac[20];
    charip[20];
    charnetmask[20];
    charmanufacturer[16];
    charplatform[32];
    charsystem[32];
    charversion[64];
};
```

Field	Description
char mac	Mac
charip	Ip
char netmask	Netmask
char manufacture	Manufacturer
char platform	Platform
char system	System
char version	version

See also:
HA_RegDiscoverIpscanCb
HA_DiscoverIpscan

Live Stream Data

```
structHA_LiveStream
{
    int w;
    int h;
    STREAM_FORMAT format;
    int streamLen;
    int streamBufSize;
    char* streamBuf;
    unsigned int  h264_sequence;
    unsigned char channel;
};
```

Field	Description
int w	Width of video
int h	Height of wideo
STREAM_FORMAT format	Video format
int streamLen	Length of stream
int streamBufSize	Size of stream buffer
char* streamBuf	Stream buffer
unsigned int h264_sequence	Frame number
unsigned char channel	channel 0: mainstream 4:substream

```
typedef enum
{
   STREAM_FORMAT_JPEG = 1,
   STREAM_FORMAT_H264 = 2
}STREAM_FORMAT;
```

Field	Description
STREAM_FORMAT_JPEG	Jpeg streaming
STREAM_FORMAT_H264	H264 streaming

See also:

HA_RegLiveStreamCbEx

HA_RegLiveStreamCb

Camera System Config

```
structFaceSystemConfig{
    structTemporaryParam temp;
    structPlatformParam platform;
    structStreamParam stream;
    structAppParam app;
    structOutputerParamoutputer;
    struct FaceExtraParamextraParam;
};
```

Field	Description
structTemporaryParam temp	Temporary parameter
structPlatformParam platform	Platform parameter
structStreamParam stream	Stream parameter
structAppParam app	Application parameter
structOutputerParamoutputer	Output parameter
struct FaceExtraParam	Extra face parameter

Note:
This is the struct to configure all the parameters, it is not recommended to use this struct to configure camera.

Matching Mode

```
enumMatchMode{
    MATCH_MODE_NULL                 = 0,
    MATCH_MODE_NORMAL               = 1,
    MATCH_MODE_IDCARD_1TO1          = 2,
    MATCH_MODE_FACE_IDCARD          = 3,
    MATCH_MODE_WGCARD               = 4,
    MATCH_MODE_FACE_WGCARD          = 5,
    MATCH_MODE_ANY_FACE             = 6,
    MATCH_MODE_NORMAL_OR_WGCARD          = 7,
    MATCH_MODE_NORMAL_OR_IDCARD_1TO1     = 8,
    MATCH_MODE_NORMAL_OR_SNAPSHOT        = 20
};
```

Field	Description
MATCH_MODE_NULL	Disable match
MATCH_MODE_NORMAL	Open the gate if it's whitelisted name
MATCH_MODE_IDCARD_1TO1	By id card
MATCH_MODE_FACE_IDCARD	Face or id card
MATCH_MODE_WGCARD	Wiegand card
MATCH_MODE_FACE_WGCARD	Face or Wiegand card
MATCH_MODE_ANY_FACE	Any detected face will open the gate
MATCH_MODE_NORMAL_OR_WGCARD	Face and whitelisted name; Wiegand card and whitelisted name
MATCH_MODE_NORMAL_OR_IDCARD_1TO1	Face or id card
MATCH_MODE_NORMAL_OR_SNAPSHOT	Face or RFID full image

See also:
    HA_SetMatchMode

PTZ Control

```
enumPTZCTL {
    PTZ_CTRL_ZOOM_IN    =5,
    PTZ_CTRL_ZOOM_OUT   =6,
    PTZ_CTRL_FOCUS_IN   =7,
    PTZ_CTRL_FOCUS_OUT  =8
};
```

Field	Description
PTZ_CTRL_ZOOM_IN	Zoom in
PTZ_CTRL_ZOOM_OUT	Zoom out
PTZ_CTRL_FOCUS_IN	Focus
PTZ_CTRL_FOCUS_OUT	Defocus

```
enumPTZMODE {
   PTZ_MODE_ONCE   =1,
   PTZ_MODE_START  =2,
   PTZ_MODE_STOP   =3
};
```

Field	Description
PTZ_MODE_ONCE	One-time operation
PTZ_MODE_START	Start the operation
PTZ_MODE_STOP	Stop the operation

Server Address Configure

```
struct ExtranetParam {
   union {
   struct TcpClientParam client;
   struct HttpClientParam http;
   };
   unsigned char enable;
   char mode;
   char resv[6];
};
```

Field	Description
client	TCP configuration
http	http configuration
enable	switch to enable server upload 0: disabled 1: enabled
mode	Working mode 0: tcp 1: http comet

resv	reserved

See also:

[HA_SetExtranetParam]
[HA_GetExtranetParam]

```
struct TcpClientParam
{
    char ip[16];
    int port;
    char enable;
    unsigned char enable_verify;
    char user_name[16];
    char passwd[17];
    char resv[65];
};
```

Field	Description
ip	IP address
port	Port number
enable	Value indicating whether the function is enabled (deprecated, for compatible purpose only)
enable_verify	Work mode 0: tcp 1: http comet
enable_verify	Value indicating whether enable login authentication
user_name	User name
passwd	password
resv	reserved

```
struct HttpClientParam
{
    char ip[16];
    unsigned short port;
    char url[102];
};
```

Field	Description
ip	Ip address
port	Port number

url	http url

Note:
url and ip are mutually exclusive

OSD items

```
enumLcdDisplayItem{
    LCD_DISPLAY_ITEM_TITILE      = 0x1,
    LCD_DISPLAY_ITEM_TIME        = 0x1 << 1,
    LCD_DISPLAY_ITEM_IP          = 0x1 << 2,
    LCD_DISPLAY_ITEM_PERSON_NUM  = 0x1 << 3
};
```

Field	Description
LCD_DISPLAY_ITEM_TITILE	Title
LCD_DISPLAY_ITEM_TIME	Time
LCD_DISPLAY_ITEM_IP	Ip
LCD_DISPLAY_ITEM_PERSON_NUM	Count of registered faces

See also:
HA_GetLcdDisplayItems
HA_SetLcdDisplayItems

Gate Access Control Rule

```
struct MinuteSchedule {
    unsigned char hour;
    unsigned char minute;
};
```

Field	Description
hour	Hour (0-24)
minute	Minute (0-59)

```
structHourSchedule {
    MinuteSchedulestart;
    MinuteScheduleend;
};
```

Field	Description
start	Start time (inclusive)
end	End time (exclusive)

```
struct DailySchedule {
    unsigned int Sector;
    HourScheduleSchTime[6];
    char recv[16];
};
```

Field	Description
Sector	Time slot count. Max 6 slots: SchTime[0] to SchTime[5]. SchTim[0] to SchTime[1] is the slot when gate will be opened automatically
SchTime	Gate open time slots array
recv	Reserved

```
enumScheduleMode {
    SCHEDULE_MODE_NONE = 0,
    SCHEDULE_MODE_DAILY = 1,
    SCHEDULE_MODE_WEEKLY = 2
};
```

Field	Description
SCHEDULE_MODE_NONE	Do not use predefined gate access rule
SCHEDULE_MODE_DAILY	The rule works the same way on every weekday
SCHEDULE_MODE_WEEKLY	The rule works on specified weekday only

```
struct KindSchedule{
    enumScheduleMode Mode;
```

```
        char ScheduleName[16];
        unsigned short ScheduleNameCode;
        DailyScheduleSchedule[7];
        char recv[32];
    };
```

Field	Description
Mode	Rule mode
ScheduleName	Rule name
ScheduleNameCode	Rule index. Must be unique, see ScheduleMode
Schedule	Time parameter open the gate. If Mode==SCHEDULE_MODE_DAILY, only Schedule[0] are read. If Mode==SCHEDULE_MODE_WEEKLY, Schedule[0] to Schedule[6] is corresponding to Monday to Sunday
recv	Reserved

See also:

HA_GetScheduleModeCfg
HA_SetScheduleModeCfg

Holiday Setting

```
structFestivalItem {
    char festival_desc[32];
    unsigned int term_start;
    unsigned int term_end;
    char res[32];
};
```

Field	Description
festival_desc	Holiday name
term_start	Holiday start time, seconds from epoch time
term_end;	Holiday end time, seconds from epoch time

```
struct ScheduleFestival{
   unsigned int festival_num;
   FestivalItemfestival_item[15];
};
```

Field	Description
festival_num	Holidays array length, max 15
festival_item	Configure for a single holiday

See also:

- HA_SetScheduleFestival
- HA_GetScheduleFestival

Platform Integration Config

```
   struct PlatformAccess{
      unsigned char enable;
      char ID [67] ;
      union {
         Platform0 form0;
         Platform1 form1;
         Platform2 form2;
         char resv[1024];
      };
   };
```

Field	Description
enable	Enable or disable 0: disable 1: enable
ID	ID 000001:public 000002:Jigong

	000003:Chengdu House & Construction department platform
form0	000001
form1	000002
form2	000003
resv	reserved

```
struct Platform0{
    unsigned intsynInterval;
    char serverUrl[128];
    char resv[892];
};
```

Field	Description
synInterval	Sync interval
serverUrl	Server url
resv	Reserved

```
struct Platform1{
    unsigned int synInterval;
    char serverUrl[128];
    char personSynUrl[128];
    char signUploadUrl[128];
    char resv[636];
};
```

Field	Description
synInterval	Sync interval
serverUrl	Server url
personSynUrl	Person sync url
signUploadUrl	Sign in upload url
resv	Reserved

```
struct Platform2{
    unsigned intsynInterval;
    char serverUrl[128];
    char personSynUrl[128];
    char signUploadUrl[128];
    char personDeleteUrl[128];
    char manageFeedBackUrl[128];
    unsigned char key[8];
    char resv[372];
};
```

Field	Description
synInterval	Sync interval
serverUrl	Server url
personSynUrl	Person sync url
signUploadUrl	Sign in upload url
personDeleteUrl	Person delete request url
manageFeedBackUrl	Feedback url
Key	Encryption key
resv	Reserved

See also:

HA_GetPlatformAccessParam

HA_SetPlatformAccessParam

Update Server Parameter

```
struct UpdataServerParam
{
    char enable;
char addrtype;
unsigned short port;
char ip[64];
char URL[60];
    char resv[128];
};
```

Field	Description
enable	Remote upgrade enable or disable 0: disable Non-zero: enable
addrtype	Address type 0:IP 1: domain name
port	Port number
ip	Ip
URL	url
resv	Reserved

See also:
    HA_GetUpdataServerParam

4G Signal Status

```
    struct FourthGInfo{
       int  FGModuleReady;
    int  SIMReady;
    int  MCC;
    int  MCN;
    char dataCap[16];
    char ICCID[32];
    char IMSI[32];
    int  signalStrength;
    int  connected;
       char resv[256];
    };
```

Field	Description
FGModuleReady	4G module ready status 0: not ready Non-zero: ready
SIMReady	SIM ready status 0: not ready Non-zero: ready
MCC	MobileCountryCode
MCN	Mobile operator code: China mobile: 0, 2, 7 China unicom:1, 6, 9 China telecom:3, 5, 11
dataCap	LTE etc
ICCID	ICCID
IMSI	IMSI
signalStrength	Signal strength
connected	Connected or not
resv	reserved

See also:
    HA_Get4GInfo

Wifi Info

```
struct WifiSignal
{
    char ssid[36];
    unsigned int frequency;
    int rssi;
    char mac[20];
char encryptMethod[256];
int speed;
};
```

Field	Description
ssid	Wifissid name
frequency	Frequency 2.4G or 5G
rssi	rssi 2.4G: from -126 up to 0 5G: from 156 up to 200
mac	MAC
encryptMethod	Encryption method
speed	Connection speed

See also:

HA_SearchWifi

Capture Record Storage

```
struct CapacityHistory
{
    int maxCapacity;
    int uploadedNum;
    int unUploadedNum;
    int totalNum;
    char resv[32];
};
```

Field	Description
maxCapacity	Capacity of store capture record

uploadedNum	Count of uploaded record
unUploadedNum	Count of not uploaded record
totalNum	Current count of record
resv	Reserved

See also:

HA_QueryCapacityHistory

Face Database Storage

```
struct StorageCapacity
{
    int maxCapacity;
    int whiteListNum;
    int blackListNum;
    int normalListNum;
    int totalNum;
    char resv[32];
};
```

Field	Description
maxCapacity	Capacity of face database
whiteListNum	Count of registered whitelisted face
blackListNum	Count of registered blacklisted face
normalListNum	Count of registered normal face
totalNum	Total count of registered face
resv	Reserved

See also:

HA_StorageCapacity

Camera Storage Info

```
struct MemoryInfor
{
    unsigned intSDCardTotalSize;
    unsigned intSDCardUsedSize;
    unsigned intEMMCTotalSize;
    unsigned intEMMCUsedSize;
    char resv[64];
};
```

Field	Description
SDCardTotalSize	Capacity of SD card inKB
SDCardUsedSize	Used capacity of SD card in KB
EMMCTotalSize	Capacity of EMMC in KB
EMMCUsedSize	Used capacity of EMMC in KB
resv	Reserved

See also:

HA_QuerySDCardInfoEx

Data Modification Flag

```
enumParsonDataFlags
{
  DATA_FLAG_NULL                 = 0x0,
  DATA_FLAG_NAME                 = 0x1,
  DATA_FLAG_ROLE                 = 0x1 << 1,
  DATA_FLAG_WG                   = 0x1 << 2,
  DATA_FLAG_EFFET                = 0x1 << 3,
  DATA_FLAG_EFFECTSTART           = 0x1 << 4,
  DATA_FLAG_SCHEDULE             = 0x1 << 5,
  DATA_FLAG_USERPARAM            = 0x1 << 6,
  DATA_FLAG_NORM_IMAGE           = 0x1 << 7,
  DATA_FLAG_REG_IMAGES           = 0x1 << 8,
};
```

Field	Description
DATA_FLAG_NULL	none
DATA_FLAG_NAME	name
DATA_FLAG_ROLE	category
DATA_FLAG_WG	Wiegand
DATA_FLAG_EFFET	Valid from
DATA_FLAG_SCHEDULE	Schedule rule
DATA_FLAG_USERPARAM	User-defined parameter
DATA_FLAG_NORM_IMAGE	Normalized image
DATA_FLAG_REG_IMAGES	Register image

See also:

HA_QuerySDCardInfoEx

```
struct SipRoomIdGroup
{
    char roomId[64];
    char sipNum[128];
};
```

Field	Description
roomId	Room number
sipNum	Sipnumber

See also:
    HA_CamSipAddRoomId

SIP Call Event

```
enumSIPEventCall
{
    SIP_EVENT_CALL_CLOSED  = 0,
    SIP_EVENT_CALL_RING    = 1,
    SIP_EVENT_CALL_RINGING = 2,
    SIP_EVENT_CALL_REJECT  = 3,
    SIP_EVENT_CALL_ANSWERD = 4,
    SIP_EVENT_CALL_INVITE  = 5,
    SIP_EVENT_CALL_INVITE_ANSWERD = 6
};
```

Field	Description
SIP_EVENT_CALL_CLOSED	Call end
SIP_EVENT_CALL_RING	Call start
SIP_EVENT_CALL_RINGING	Ringing
SIP_EVENT_CALL_REJECT	Rejected
SIP_EVENT_CALL_ANSWERD	Answered
SIP_EVENT_CALL_INVITE	Invited
SIP_EVENT_CALL_INVITE_ANSWERD	Invited answered

See also:
    HA_SIPCallEventCb_t

Face Matching Fail Reason

```
enumMatchFailedReasons
{
  MATCH_FAILED_NULL                   = 0,
  MATCH_FAILED_NOT_WHITE              = -2,
  MATCH_FAILED_EXPIRE                 = -3,
  MATCH_FAILED_UN_CHEDULES            = -4,
  MATCH_FAILED_FESTIVAL               = -5,
  MATCH_FAILED_ABN_TEMPERATURE        = -6,
  MATCH_FAILED_MASK                    = -7,
  MATCH_FAILED_WITHOUT_HAT            = -8,
  MATCH_FAILED_INVAILED_CARD          = -9,
  MATCH_FAILED_UMMATCHED_ID           = -10,
  MATCH_FAILED_NOAUTH                 = -11
};
```

Field	Description
MATCH_FAILED_NULL	none
MATCH_FAILED_NOT_WHITE	restricted
MATCH_FAILED_EXPIRE	expired
MATCH_FAILED_UN_CHEDULES	Not in schedule time
MATCH_FAILED_FESTIVAL	holiday
MATCH_FAILED_ABN_TEMPERATURE	Temperature too high
MATCH_FAILED_MASK	maskless
MATCH_FAILED_WITHOUT_HAT	No helmet
MATCH_FAILED_INVAILED_CARD	Card not registered
MATCH_FAILED_UMMATCHED_ID	Face doesn’t match id
MATCH_FAILED_NOAUTH	unauthorized

Error Code

Error code	value	Description
ERR_NONE	0	No errors

ERR_INVALID_PARAM	-1	Illegal value
ERR_TIMEOUT	-2	Response timeout
ERR_SEND_BUF_FULL	-3	Send cache full
ERR_SYS_NOT_MATCH	-4	retain
ERR_UNCONNECTED	-5	Camera not connected
ERR_SNAPSHOT_UNAVAILABLE	-6	Screenshot failed
ERR_JPEG_ENCODE_ERROR	-7	JPEG decoding failed
ERR_BUF_TOO_SMALL	-8	Insufficient cache space
ERR_CANCEL	-9	retain
ERR_UNABLE_TO_OPEN_FILE	-10	File open failed
ERR_DEVICE_NOT_SUPPORTED	-11	Device not supported
ERR_COUNT_INVALID	-12	Reserved
ERR_OUT_PUT_OF_ARRAY	-13	Reserved
ERR_GET_FACE_FEATURE	-14	The face feature extraction failed, so it is necessary to ensure that there is only one face in the image
ERR_FACE_ID_REPEAT	-15	Failed to add personnel, personnel ID is repeated
ERR_FACE_ID_NOT_EXITS	-16	Failed to modify face feature, no corresponding ID found
ERR_GET_FACE_INIT	-17	Face extractor is not initialized. Please call ha_ Initfacemodel initialization
ERR_GET_FACE_ID	-18	No current record
ERR_SERIAL_INDEX	-19	Wrong serial number, only ha is supported at present_ SERIAL_ RS485 and ha_ SERIAL_ RS232
ERR_SYSTEM_REBOOT	-20	System restart failed
ERR_APP_REBOOT	-21	Failed to restart application
ERR_ENCODE_JPG	-22	Failed to compress image
ERR_FACES_NUM	-23	Maximum support for 5 images by a single person
ERR_IMAGE_DECODE	-24	Image decoding failed
ERR_IMAGE_SIZE	-25	The image is too large, and the width and height of JPG image sent to the camera cannot exceed 128x128
ERR_IMAGE_PATH	-26	The file does not exist. Please check the picture path is correct

ERR_SAVE_IMG_NUM	-27	Currently, only one template image is supported in the camera. Pictureflags must be <=1
ERR_PTZ_CTRL	-28	The cloud platform control that is not supported currently supports only 5: Zoom (zoom in) 6: Zoom (zoom out) 7: focus (close) 8: focus (pull away)
ERR_PTZ_CTRL_MODE	-29	Unsupported control mode, 1: one motion 2: start 3: stop
ERR_UPPER_LIMIT	-31	The number of personnel has reached the maximum
ERR_PROTOCAL_VER	-32	Protocol version does not match. Please confirm the SDK and camera firmware version
ERR_REQUEST_CMD	-33	Unsupported operation request
ERR_PACKET_DATA	-34	Request packet contains illegal fields
ERR_AUTH_FAILED	-35	Authentication failed
ERR_WRITE_DATA	-36	Write data failed
ERR_READ_DATA	-37	Failed to read data
ERR_TWIST_FACE	-38	Normalized image failed
ERR_EXTRACT_FEATURE	-39	Failed to extract feature
ERR_MIN_FACE	-40	Face size is too small, and the face profile must be greater than 96 * 96
ERR_QVALUE_TOO_SMALL	-41	The quality of the portrait is too poor to meet the registration conditions
ERR_INVALID_SYNC_MODE	-42	Invalid synchronization operation
ERR_WG_QUERY_MODE	-43	Wegenka number does not support fuzzy query
ERR_SYSTEM_BUSY	-44	Operating system busy
ERR_VERSION_MATCH	-45	Version mismatch
ERR_TOO_MUCH_FACE	-46	The number of faces in the image is less than 1
ERR_FACE_INCOMPLETE	-47	Incomplete face in image
ERR_ANGLE_PITCH	-48	Face pitch angle too large
ERR_ANGLE_YAW	-49	Face side angle too large
ERR_ANGLE_ROLL	-50	Face is not correct

ERR_MOUTH_OPEN	-51	Too much mouth opening
ERR_YINYANG_FACE	-52	Uneven illumination
ERR_VISIBLE_TARGET	-54	No visible targets specified were checked out
ERR_INFRARED_TARGET	-55	No infrared targets specified were checked out
ERR_ABERRATION_TOO_BIG	-56	Visible light infrared aberration too large
ERR_REPLYCODE_FEATURE_VERSION	-60	Feature data version mismatch
ERR_LACK_TWISTIMGE	-61	Missing normalized image
ERR_FUNC_AUTH	-70	Function authorization failed
ERR_FUNC_AUTHORIZED	-71	Feature authorized
ERR_UN_AUTH	-72	Function not authorized
ERR_4GINFO	-75	4G module error
ERR_PING_BLOCK	-76	Ping command error
ERR_UNKNOWN	-1000	unknown error

4 Functions

4.1 SDK Initialization

```
HASDK_API void HA_Init();
```

Description:
SDK initialization

Note:
Initialize the SDK by calling HA_Init() or HA_InitEx() only once

HA_InitEx

```
HASDK_API void HASDK_CALL HA_InitEx(unsigned int maxCamNum);
```

Description:
SDK initialization

Arguments:

Argument	Description	In/Out

maxCamNum	0 by default	In

Note:

Initialize the SDK by calling HA_InitEx() or HA_Init() only once

HA_InitFaceModel

```
   HASDK_API int HASDK_CALL HA_InitFaceModel(const char
*modelDir);
```

Description:

Initialize the face model

Arguments:

Argument	Description	In/Out
modelDir	path to the model	In

Returns:

Value	Description
0	success
-1	fail

Note:

Fail to initialize the face model will result in all person registering related function errors. SDKv0.9.7 and later versions must import model files

4.2 SDK Cleanup

HA_DeInit

```
   HASDK_API void HASDK_CALL HA_DeInit();
```

Description:

Clean up SDK and release resources

Note:

Should be called only once

4.3 Device Configuration

4.3.1 SDK Version

HA_GetVersion

```
    HASDK_API int HASDK_CALL
HA_GetVersion(structHaSdkVersion*version);
```

Description:

Get SDK version

Arguments:

Argument	Description	In/Out
version	SDK version	Out

Returns:

Value	Description
0	Success
-1	fail

HA_VersionToInt

```
    HASDK_API int HASDK_CALL HA_VersionToInt(char *version);
```

Description:

Convert SDK version to int

Arguments:

Argumen t	Description	In/Out
version	SDK version Format: %02d.%02d.%02d	In

Returns:

Value	Description
0	Success
-1	Fail

HA_VersionCheck

```
    HASDK_API int HASDK_CALL HA_VersionCheck(
    structHaSdkVersion*sdkVersion,
    structSystemVersionInfo*systemVersion
    );
```

Description:
    Check if system version matches SDK version

Arguments:

Argument	Description	In/Out
sdkVersion	SDK version	In
systemVersion	System version	In

Returns:

Value	Description
0	match
Non-zero	See error code

4.3.2 Firmware Info

Camera Diagnose

HA_GetReq_DiaGgnose

```
    HASDK_API int HASDK_CALL HA_GetReq_DiaGgnose(
    structHA_Cam* cam,
    char* json,
    int *jsonLen
    );
```

Description:
    Get diagnostic info from camera
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
json	Buffer to receive json	Out
jsonLen	Length of the buffer	Out

Returns:

Value	Description
0	success
Non-zero	See error code

Note:
  The buffer must be > 5 KB

Camera System Info

HA_GetFaceSystemVersionEx

```
HASDK_API int HASDK_CALL HA_GetFaceSystemVersionEx(
structHA_Cam *cam,
structSystemVersionInfo *version
);
```

Description:
  Get system version info from camera

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
version	The buffer to receive version info	Out

Returns:

Value	Description
0	Success
Non-zero	See error code

Camera System Time

HA_GetSystemTime

```
HASDK_API int HASDK_CALL HA_GetSystemTime(
structHA_Cam *cam,
structSystemTime *sysTime
);
```

Description:

Get current system time from camera

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
sysTime	System time	Out

Returns:

Value	Description
0	Success
Non-zero	See error code

HA_SetSystemTime

```
HASDK_API int HASDK_CALL HA_SetSystemTime(
structHA_Cam *cam,
structSystemTime *sysTime
);
```

Description:

Set system time of camera

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
sysTime	System time	In

Note:

无

Example:

```
SystemTimesystime;
strcpy(systime.date, "2018/8/8");//the date
strcpy(systime.time, "15:50:05");//the time
systime.time_zone = 8; //time zone, GMT+8
int ret = HA_SetSystemTime(
    g_cam[Using_cam].cam, //Handle to camera
    &systime
);
if (ret == ERR_NONE) {
    printf("success");
}
```

4.3.3 Face Recognition Parameter

HA_SetFaceCheckEnable

```
HASDK_API int HASDK_CALL HA_SetFaceCheckEnable(int onoff);
```

Description:
Enable or disable face image validation before registration

Arguments:

Argument	Description	In/Out
onoff	0: disable Non-zero: enable	In

Returns:
Always returns 0

Face Detection Rectangle

HA_GetDetectRect

```
HASDK_API int HASDK_CALL HA_GetDetectRect(
structHA_Cam* cam,
structha_rect* rect
);
```

Description:
Get region of interest (ROI) of face recognition of camera

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rect	The roi	Out

Returns:

Value	Description
0	Success
Non-zero	error code

HA_SetDetectRect

```
HASDK_API int HASDK_CALL HA_SetDetectRect(
structHA_Cam* cam,
structha_rect* rect
);
```

Description:

Set region of interest (ROI) of face recognition for the camera

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rect	The roi	In

Returns:

Value	Description
0	Success
Non-zero	Error code

Note:

Face recognition outside the ROI is disabled.

Face Detect Min Rectangle

HA_GetFaceDetectMinRect

```
HASDK_API int HASDK_CALL HA_GetFaceDetectMinRect(
structHA_Cam* cam,
unsignedint*size);
```

Description:

Get the minimal face detect region

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
size	The minimal face detect region = size*size	Out

HA_SetFaceDetectMinRect

```
    HASDK_API int HASDK_CALL HA_SetFaceDetectMinRect(
    structHA_Cam* cam,
    unsignedintsize);
```

Description:
    Set the minimal face detect region

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
size	The minimal face detect region = size*size	In

Note:
    If the face area is smaller than the specified minimum detection size, it will be ignored.

Face Angle Filtering

HA_GetValidAngleEnable

```
    HASDK_API int HASDK_CALL HA_GetValidAngleEnable(
    structHA_Cam* cam,
    char *angle,
    char *enable
    );
```

Description:
    Get face to camera angle filtering value

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
angle	The angle (0-90)	Out
enable	Enable or disable 0: disable 1: enable	Out

HA_SetValidAngleEnable

```
HASDK_API int HASDK_CALL HA_SetValidAngleEnable(
structHA_Cam* cam,
char *angle,
charenable
);
```

Description:
    Set face to camera angle filtering value

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
angle	The angle (0-90)	In
enable	Enable or disable 0: disable 1: enable	In

Note:
    If the face to camera angle is bigger than specified value, it will be ignored

Face Image Quality Threshold

HA_GetQvalueThresholdEnable

```
HASDK_API int HASDK_CALL HA_GetQvalueThresholdEnable(
structHA_Cam* cam,
char *threshold,
char *enable
);
```

Description:
    Get the value indicating whether the image quality threshold is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
threshold	The threshold value	Out

	（0-100）	
enable	Enable or disable 0: disable 1: enable	Out

HA_GetQvalueThresholdEnable

```
    HASDK_API int HASDK_CALL HA_SetQvalueThresholdEnable(
    structHA_Cam* cam,
    char *threshold,
    char *enable
    );
```

Description:
    Enable or disable image quality threshold

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
threshold	The threshold value （0-100）	In
enable	Enable or disable 0: disable 1: enable	In

Note:
    If enabled, the image of which quality is lower than the specified value will be ignored for face detection.

Face Matching Confidence Score

HA_GetMatchScore

```
    HASDK_API int HASDK_CALL HA_GetMatchScore(
    structHA_Cam* cam,
    int *score
    );
```

Description:
    Get the face matching confidence score

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
score	The confidence score （0-100）	Out

HA_SetMatchScore

```
HASDK_API int HASDK_CALL HA_SetMatchScore(
structHA_Cam* cam,
int *score
);
```

Description:
    Set face matching confidence score

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
score	The confidence score （0-100）	In

Note:
    The matching score that is smaller than the specified value
  is considered failed matching

Face Capture Output Control

HA_GetOutputCtl

```
HASDK_API int HASDK_CALL HA_GetOutputCtl(
structHA_Cam* cam,
int *ctl
);
```

Description:
    Get the output options of face recognition result
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ctl	Output configure	Out

	0: no image 1: full image 2: close-up 4: match template image 8: feature data 16: debug image	

HA_SetOutputCtl

```
    HASDK_API int HASDK_CALL HA_SetOutputCtl(
    structHA_Cam* cam,
    int ctl
    );
```

Description:

Set output options of face recognition result

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ctl	Output options (bitwise OR operation is supported) 0: none 1: full image 2: close-up 4: template image 8: feature data 16: debug image	In

Face Matching Enable

HA_GetMatchEnable

```
    HASDK_API int HASDK_CALL HA_GetMatchEnable(
    structHA_Cam* cam,
    int *enable
    );
```

Description:

Get whether face matching is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

enable	Value indicating whether face matching is enabled 0: disabled 1: enabled	Out

HA_SetMatchEnable

```
HASDK_API int HASDK_CALL HA_SetMatchEnable(
structHA_Cam* cam,
int enable
);
```

Description:
Enable or disable face matching

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether enable face matching 0: disable 1: enable	In

Liveness Detection Enable (binocular version only)

HA_GetAliveDetectEnable

```
HASDK_API int HASDK_CALL HA_GetAliveDetectEnable(
structHA_Cam* cam,
int *enable
);
```

Description:
Get the value indicating whether liveness detection is enabled

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
enable	Value indicating whether liveness detection is enabled 0: disabled 1: enabled	Out

HA_SetAliveDetectEnable

```
HASDK_API int HASDK_CALL HA_SetAliveDetectEnable(
structHA_Cam* cam,
int enable
);
```

Description:
Set whether enable liveness detection

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether liveness detection is enabled 0: disable 1: enable	In

Body Temperature Detect (temperature detect module must be installed)

HA_GetTemperaturEnable

```
HASDK_API int HASDK_CALL HA_GetTemperaturEnable(
structHA_Cam* cam,
int *enable
);
```

Description:
Get value indicating whether temperature detection is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

enable	Value indicating whether temperature detection is enabled 0: disabled 1: enabled	Out

HA_SetTemperaturEnable

```
HASDK_API int HASDK_CALL HA_SetTemperaturEnable(
structHA_Cam* cam,
int enable
);
```

Description:

Set whether temperature detection is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether temperature detection is enabled 0: disabled 1: enabled	In

Mask Detection

HA_GetMaskInspectEnable

```
HASDK_API int HASDK_CALL HA_GetMaskInspectEnable(
structHA_Cam* cam,
int *enable
);
```

Description:

Get whether mask detection is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether mask detection is enabled 0: disable	Out

	1: enable	

HA_SetMaskInspectEnable

```
HASDK_API int HASDK_CALL HA_SetMaskInspectEnable(
structHA_Cam* cam,
int enable
);
```

Description:

Set whether mask detection is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether mask detection is enabled 0: disable 1: enable	In

Gate Opening Temperature Threshold

HA_SetTemperaturLimit

```
HASDK_API int HASDK_CALL HA_SetTemperaturLimit(
structHA_Cam* cam,
float temperatur,
int enable
);
```

Description:

Set temperature threshold value for gate access control. Gate will not open for those people whose body temperature is higher than this value.

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
temperatur	The body temperature threshold value	In
enable	Value indicating whether enable this function 0: disable 1: enable	In

HA_GetTemperaturLimit

```
    HASDK_API int HASDK_CALL HA_GetTemperaturLimit(
    structHA_Cam* cam,
    float* temperatur,
    int* enable
    );
```

Description:

Get the temperature threshold value for gate access control.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
temperatur	The body temperature threshold value	Out
enable	Value indicating whether enable this function 0: disable 1: enable	Out

Gate Open Mask Configuration

HA_GetProhibitSafetyMask

```
    HASDK_API int HASDK_CALL HA_GetProhibitSafetyMask(
    structHA_Cam* cam,
    int *enable
    );
```

Description:

Get the value thatindicats whether open the gate only ifone wears mask.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value to disable/enable this feature 0: disable 1: enable	Out

HA_SetProhibitSafetyMask

```
HASDK_API int HASDK_CALL HA_SetProhibitSafetyMask(
structHA_Cam* cam,
int enable
);
```

Description:

Enable or disable the feature that open the gate only if one wears mask.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable 0: disable 1: enable	In

Face Detect Auto Score

HA_SetAutoScoreEnable

```
HASDK_API int HASDK_CALL HA_SetAutoScoreEnable(
structHA_Cam* cam,
int enable
);
```

Description:

Use the confidence score that is adjusted automatically for facial recognition

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable 0: disable 1: enable	In

HA_GetAutoScoreEnable

```
    HASDK_API int HASDK_CALL HA_GetAutoScoreEnable(
    structHA_Cam* cam,
    int* enable
    );
```

Description:

Get the value thatindicats whether automatic confidence score is used

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enabled or disabled 0:disabled 1:enabled	Out

Duplicated Face Detection

HA_GetDereplicationgConfig

```
    HASDK_API int HASDK_CALL HA_GetDereplicationgConfig(
    structHA_Cam* cam,
    int *enable,
    int *timeout
    );
```

Description:

Get whether duplication detection is enabled and the time span when recurrence of same person will be considered duplicated and thus only be recorded once.

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
enable	Enable or disable 0:disable 1:enable	Out
timeout	The time span in seconds (1s~60s)	Out

HA_SetDereplicationEnable

```
    HASDK_API int HASDK_CALL HA_SetDereplicationEnable(
    structHA_Cam* cam,
    int enable,
    int timeout
    );
```

Description:

Set whether duplication detection is enabled and the time span when recurrence of same person will be considered duplicated and thus only be recorded once.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable 0:disable 1:enable	In
timeout	Time span (1s~60s)	In

Registration Expiry Reminder

HA_GeteExpireAlarm

```
    HASDK_API int HASDK_CALL HA_GeteExpireAlarm(
    structHA_Cam* cam,
    unsigned int *timeout
    );
```

Description:

Get the expiry reminder

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

timeout	The time before the expiry time to send an reminder in seconds	Out

HA_SeteExpireAlarm

```
    HASDK_API int HASDK_CALL HA_GeteExpireAlarm(
    structHA_Cam* cam,
    unsigned int timeout
    );
Description:
    Set the expiry reminder
Arguments:
```

Argument	Description	In/Out
cam	Handle to camera	In
timeout	The time before the expiry time to send an reminder in seconds	In

Debug Mode

HA_GetDebugEnable

```
    HASDK_API int HASDK_CALL HA_GetDebugEnable(
    structHA_Cam* cam,
    int *enable
    );

Description:
    Get value indicating whether debug is enabled

Arguments:
```

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable 0: disable 1: enable	Out

HA_SetDebugEnable

```
    HASDK_API int HASDK_CALL HA_SetDebugEnable(
    structHA_Cam* cam,
    int enable
    );
```

Description:
    Enable or disable debug
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable 0: disable 1: enable	In

4.3.4 Camera Parameter Configuration

Reset Configuration

HA_ResetFaceConfig

```
    HASDK_API int HASDK_CALL HA_ResetFaceConfig(structHA_Cam*
cam);
```

Description:
    Reset all config

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

Note:
    Network config remain unchanged

Get All Configuration

HA_GetFaceSystemCfgEx

```
    HASDK_API int HASDK_CALL HA_GetFaceSystemCfg(
    structHA_Cam* cam,
    structFaceSystemConfig* cfg
    );
```

Description:
    Get all configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	The configure parameter	Out

HA_SetFaceSystemCfgEx

```
    HASDK_API int HASDK_CALL HA_SetFaceSystemCfg(
    structHA_Cam* cam,
    structFaceSystemConfig* cfg
    );
```

Description:
    Set all configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	The configure parameter	In

HA_SetFaceAppParam

```
    HASDK_API int HASDK_CALL HA_SetFaceAppParam(
    structHA_Cam* cam,
    conststructFaceAppParam* param
    );
```

Description:
    Set face app configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	The configure parameter	In

NTP Configuration

HA_GetNtpConfig

```
HASDK_API int HASDK_CALL HA_GetNtpConfig(
structHA_Cam* cam,
structNtpInfo*ntpInfo
);
```

Description:
    Get ntp configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ntpInfo	Ntp info	Out

HA_SetNtpConfig

```
HASDK_API int HASDK_CALL HA_SetNtpConfig(
structHA_Cam* cam,
structNtpInfo*ntpInfo
);
```

Description:
    Set ntp configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ntpInfo	Ntp info	In

Network Configuration

HA_GetNetConfigEx

```
HASDK_API int HASDK_CALL HA_GetNetConfigEx(
    structHA_Cam* cam,
    structSystemNetInfoEx* netInfo
    );
```

Description:
    Get network configure
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
netInfo	Network configure	Out

HA_SetNetConfigEx

```
    HASDK_API int HASDK_CALL HA_SetNetConfigEx(
    structHA_Cam* cam,
    structSystemNetInfoEx* netInfo
    );
```

Description:
    Set network configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
netInfo	Network configure	In

Update Server Configuration

HA_GetUpdataServerParam

```
    HASDK_API int HASDK_CALL HA_GetUpdataServerParam(
        struct    HA_Cam* cam,
        struct UpdataServerParam* param
        );
```

Description:
    Get update server configure
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
param	Update server parameter	Out

HA_SetUpdataServerParam

```
    HASDK_API int HASDK_CALL HA_SetUpdataServerParam(
        struct HA_Cam* cam,
        struct UpdataServerParam* param
        );
Description:
    Set update server parameter
```

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
param	Update server configure	In

SD Card Info Query

HA_QuerySDCardInfo

```
    HASDK_API int HASDK_CALL HA_QuerySDCardInfo(
    HA_Cam* cam,
    int* hasSDCard,
    int* totalSize,
    int* usedSize
    );
Description:
    Get SD card info
```

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
hasSDCard	Whether SD card is installed 0: no 1: yes	Out
totalSize	Capacity (MB)	Out
usedSize	Usec capacity (MB)	Out

Storage Query

HA_QuerySDCardInfoEx

```
    HASDK_API int HASDK_CALL HA_QuerySDCardInfoEx(
    HA_Cam* cam,
    MemoryInfor* infor
    );
```

Description:

   Get SD card info

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
infor	Sd card info	Out

Format Storage

HA_FormatSDCard

```
    HASDK_API int HASDK_CALL HA_FormatSDCard(HA_Cam *cam);
```

Description:

   Format SD card

Note:

   Send the command to format sd card (FAT32), the successful return from the function means the command is sent successfully, the whole format process takes some time to complete (abount 2 minutes to format a 16GB sd card)

HA_FormatEMMC

```
    HASDK_API int HASDK_CALL HA_FormatEMMC(HA_Cam *cam);
```

Description

   Format EMMC

Note:

   Send the command to format EMMC, the successful return from the function means the command is sent successfully, the whole format process takes some time to complete (abount 2 minutes to format a 16GB sd card)

4.3.5 Live Video Stream Configuration

Video Encoding Format

HA_SetStreamFmt

```
    HASDK_API void HASDK_CALL HA_SetStreamFmt(
    structHA_Cam* cam,
    int decodeFmt
    );
```

Description:

    Set streaming format

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
decodeFmt	Format: 0: BGR24 Non-zero: RGB24	In

Video Stream Enable

HA_LiveStreamCtl

```
    HASDK_API int HASDK_CALL HA_LiveStreamCtl(
    structHA_Cam* cam,
    int flag
    );
```

Description:

    Enable or disable live streaming (enabled by default)

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
flag	Enable or disable 0: disable 1: enable	In

Note:

    Works for current connection only

Substream Parameter

HA_GetSubCodParam

```
HASDK_API int HASDK_CALL HA_GetSubCodParam(
    struct HA_Cam* cam,
    struct SubCodParam* param
    );
```

Description:
    Get substream parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
param	parameter	Out

Note:
    Works for current connection only

HA_SetSubCodParam

```
HASDK_API int HASDK_CALL HA_SetSubCodParam(
struct HA_Cam* cam,
const struct SubCodParam* param
);
```

Description:
    Set substream parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
param	The parameter	In

Note:
    Works only for current connection only

Start Receiving Video Stream

HA_StartStreamRv

```
    HASDK_API void HASDK_CALL HA_StartStream(
    structHA_Cam* cam,
    HWND hWnd,
HA_DecodeImageCbEx_tcb,
    void*usrParam
    );
```
Description:
    Start streaming

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
hWnd	The handle to the window to render the video, null to disable video rendering	In
cb	the callback function, can be null	In

Note:
    To add more than one video window, this function can be called repeatedly
    The data is RGB decoded frame

HA_StopStreamEx

```
    HASDK_API void HASDK_CALL HA_StopStreamEx(
    structHA_Cam* cam,
    HWNDhWnd
    );
```

Description:
    Stop video playing of specified window

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
hWnd	The handle of the window	In

HA_StopStream

```
    HASDK_API void HASDK_CALL HA_StopStream(structHA_Cam*
cam);
```

Description:

Stop all video stream of the camera

Note:

Stop all video playing of related window

HA_StopStreamEx1

```
    HASDK_API void HASDK_CALL HA_StopStreamEx1(
    structHA_Cam* cam,
    HWNDhWnd
    );
```

Description:

Stop the infrared camera streaming in specified window

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
hWnd	The handle to the window to stop rendering video	In

HA_StopStream1

```
    HASDK_API void HASDK_CALL HA_StopStream1(structHA_Cam*
cam);
```

Description:

Stop all infrared camera streaming

Note:

This function clears all registered window handles

Draw Face Rectangle

HA_DrawFaceRects

```
    HASDK_API void HASDK_CALL HA_DrawFaceRects(
    structHA_Cam *cam,
    structha_rect *rect,
    int rect_num
    );
```

Description:

    Draw face rectangles

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rect	The rectangle array	In
rect_num	Length of the rectangle array	In

HA_SetDrawRect

```
    HASDK_API void HASDK_CALL HA_SetDrawRect(
    structHA_Cam *cam,
    structha_rect *rect
    );
```

Description:

    Draw one rectangle

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rect	The rectangle to draw	In

HA_StartDrawRect

```
    HASDK_API void HASDK_CALL HA_StartDrawRect(structHA_Cam
*cam);
```

Description:

    Start drawing face tracking rectangle

Note:

Enabled by default

HA_StopDrawRect

```
    HASDK_API void HASDK_CALL HA_StopDrawRect(structHA_Cam
*cam);
```

Description:

Stop drawing face tracking rectangle

Image Quality

HA_GetOutputImageQuality

```
    HASDK_API int HASDK_CALL HA_GetOutputImageQuality(
    structHA_Cam* cam,
    int *quality
    );
```

Description:

Get output image jpg encoding quality

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
quality	Jpg quality（1-100）	Out

HA_SetOutputImageQuality

```
    HASDK_API int HASDK_CALL HA_SetOutputImageQuality(
    structHA_Cam* cam,
    int *quality
    );
```

Description:

Set output image jpg encoding quality

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
quality	Jpg quality（1-100）	In

Distortion Correction

HA_GetDeformityCorret

```
HASDK_API int HASDK_CALL HA_GetDeformityCorret(
structHA_Cam* cam,
char* ldc_enable,
int* ldc_ratio
);
```

Description:
    Get camera distortion correction parameter
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ldc_enable	Value indicating whether distortion correction is enabled. 0: disabled 1: enabled	Out
ldc_ratio	Correction parameter (-300-500)	Out

HA_SetDeformityCorret

```
HASDK_API int HASDK_CALL HA_SetDeformityCorret(
structHA_Cam* cam,
char* ldc_enable,
int* ldc_ratio
);
```

Description:
    Set camera distortion correction parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ldc_enable	Enable or disable correction 0: disable 1: enable Null: keep it unchanged	In
ldc_ratio	Correction parameter	In

	(-300-500) Null: keep it unchanged	

Camera Max Exposure Time

HA_GetMaxExposure

```
    HASDK_API int HASDK_CALL HA_GetMaxExposure(
    structHA_Cam* cam,
    short *max_exposure
    );
```

Description:

Get max exposure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
max_exposure	The max exposure （0~100ms）	Out

HA_SetMaxExposure

```
    HASDK_API int HASDK_CALL HA_SetMaxExposure(
    structHA_Cam* cam,
    shortmax_exposure
    );
```

Description:

Set max exposure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
max_exposure	The max exposure （0-100ms）	In

Camera Max Gain

HA_GetMaxGain

```
    HASDK_API int HASDK_CALL HA_GetMaxGain(
    structHA_Cam* cam,
    short *max_gain
    );
```
Description:

Get max gain

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
max_gain	The max gain value （0~40DB）	Out

HA_SetMaxGain

```
    HASDK_API int HASDK_CALL HA_SetMaxGain(
    structHA_Cam* cam,
    shortmax_gain
    );
```

Description:

Set max gain

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
max_gain	The max gain（0~40DB）	In

Camera Contrast

HA_GetContrast

```
    HASDK_API int HASDK_CALL HA_GetContrast(
    structHA_Cam* cam,
    short *contrast
    );
```

Description:

Get contrast

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
contrast	The contrast value （0~128）	Out

HA_SetContrast

```
HASDK_API int HASDK_CALL HA_SetContrast(
structHA_Cam* cam,
short contrast
);
```

Description:

Set contrast

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
contrast	The contrast value （0~128）	In

Camera Brightness

HA_GetBrightness

```
HASDK_API int HASDK_CALL HA_GetBrightness(
structHA_Cam* cam,
unsignedchar *brightness
);
```

Description:

Get brightness

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
brightness	The bright value (0~100）	Out

HA_SetBrightness

```
    HASDK_API int HASDK_CALL HA_SetBrightness(
    structHA_Cam* cam,
    unsignedchar brightness
    );
Description:
    Get brightness

Arguments:
```

Argument	Description	In/Out
cam	Handle to camera	In
brightness	The brightness（0~100）	In

Camera Saturation

HA_GetSaturation

```
    HASDK_API int HASDK_CALL HA_GetSaturation(
    structHA_Cam* cam,
    unsignedchar *saturation
    );

Description:
    Get saturation

Arguments:
```

Argument	Description	In/Out
cam	Handle to camera	In
saturation	The saturation value (0~255）	Out

HA_SetSaturation

```
    HASDK_API int HASDK_CALL HA_SetSaturation(
    structHA_Cam* cam,
    unsignedchar saturation
    );

Description:
```

Set saturation

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
saturation	The value (0~255)	In

Video Rotation Angle

HA_GetVideoRotate

```
HASDK_API int HASDK_CALL HA_GetVideoRotate(
structHA_Cam* cam,
unsignedchar *rotate
);
```

Description:
Get video rotation angle

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rotate	0: default value 1: 90 degree clockwise 2: 180 degree clockwise 3: 270 degree clockwise	Out

HA_SetVideoRotate

```
HASDK_API int HASDK_CALL HA_SetVideoRotate(
structHA_Cam* cam,
unsignedchar rotate
);
```

Description:
Set video rotation angle

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rotate	0: default value 1: 90 degree clockwise 2: 180 degree clockwise 3: 270 degree clockwise	Out

4.4 Connect Camera

4.4.1 Discovery Camera IP

HA_DiscoverIpscan

```
HASDK_API void HASDK_CALL HA_DiscoverIpscan();
```

Description:
Search camera in LAN

Note:
Turn off firewall before search.
The result will be passed through callback function.

See also:
HA_RegDiscoverIpscanCb

4.4.2 Configure Camera IP

HA_SetIpBymac

```
HASDK_API void HASDK_CALL HA_SetIpBymac(
constchar* mac,
constchar* ip,
constchar* netmask,
constchar* gateway
);
```

Description:
Set ip by MAC address

Arguments:

Argument	Description	In/Out
mac	MAC	In
ip	Ip	In
netmask	Netmask	In
gateway	Gateway	In

Note:

Configure ip doesn’t require connection to the camera

4.4.3 Connect Camera

HA_Connect

```
HASDK_API structHA_Cam* HASDK_CALL HA_Connect(
constchar* ip,
unsignedshort port,
constchar* usrName,
constchar* password,
int* errorNum
);
```

Description:

Connect to camera, auto-reconnect is enabled by default

Arguments:

Argument	Description	In/Out
ip	Ip address	in
port	Port number, 9527 by default	in
usrName	User name If null value used, “admin” will be used	in
password	Password If null, “admin” will be used	in
errorNum	Error code	out

Returns:

Handle of camera

Note:
    Use HA_Connected to check if connected successfully

See also:
    HA_RegVerifyStatusCb
    HA_RegConnectEventCbEx
    HA_RegConnectEventCb

Sample code:

```
        interroNum;
        HA_Cam* cam = HA_Connect(
            "192.168.0.111",
            9527, NULL, NULL, //for default user name and password, use
 nullvalue
            &erroNum
        );//connect to the camera
        if(!HA_Connected(cam)) {
            printf("connect failed\n");
            return;
        }
```

HA_ConnectEx

```
    HASDK_API structHA_Cam* HASDK_CALL HA_ConnectEx(
    constchar* ip,
    unsignedshort port,
    constchar* usrName,
    constchar* password,
    int* errorNum,
    unsignedintchannel,
    int autoReconn
    );
```

Description:
    Connect to camera

Argument:

Argument	Description	In/Out
ip	IP address	In
port	Port number, fixed to 9527	In
usrName	User name	In

	If null, “admin” will be used	
password	password if null, “admin” will be used instead	In
errorNum	Error code	Out
channel	Channel number, pass 0	In
autoReconn	Value indicating whether auto-reconnect 0 : disable auto-reconnect 1: auto-reconnect	In

Returns:
Handle to the camera (if autoReconn=1)
Handle to the camera if success, otherwise null (if autoReconn = 0)

Note:
The extended version of HA_Connect function
Setting autoReconn to 1 is recommended

See also:
HA_RegVerifyStatusCb
HA_RegConnectEventCbEx
HA_RegConnectEventCb

4.4.4 Disconnect Camera

HA_DisConnect

Signature:
```
HASDK_API void HASDK_CALL HA_DisConnect(structHA_Cam* cam);
```
Description:
Disconnect from camera
Note:
SDK cleanup function will release the handle automatically
If app only connect to one camera, calling this function when exiting the app is not necessary

4.4.5 Update Camera Login Info

HA_GetAuthEnableInfo

```
HASDK_API int HASDK_CALL HA_GetAuthEnableInfo(
HA_Cam* cam,
int *enable
);
```

Description:
Get the value indicating whether login authentication is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether login authentication is enabled 0: disable 1: enable	Out

HA_SetAuthInfo

```
HASDK_API int HASDK_CALL HA_SetAuthInfo(
HA_Cam* cam,
constchar *user_name,
constchar *passwd,
structAuthParam*authInfo
);
```

Description:
Enable or disable login authentication

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
user_name	current user name	In

passwd	Current password	In
authInfo	New login info	In

Note:
Setting new authentication info disconnect all connections, the SDK client that invokes the function reconnect automatically, other SDK clients need call Connect function with new login info

4.4.6 Connection Event Notification

HA_SetNotifyConnected

```
HASDK_API void HASDK_CALL HA_SetNotifyConnected(int
notify);
```

Description:
Enable or disable connection event notification (disabled by default)

Arguments:

Argument	Description	In/Out
notify	0: disable 1: enable	In

See also:
HA_RegConnectEventCbEx
HA_RegConnectEventCb

4.4.7 SDK Authentication

HA_SetSDKPassword

```
HASDK_API int HASDK_CALL HA_SetSDKPassword(
const char* password
);
```

Description:
Set SDK password, this is the password set by calling

HA_SetCamSDKPassword function.

Arguments:

Argument	Description	In/Out
password	SDK authentication password, which is used by camera side SDK functionality authentication	In

HA_SetCamSDKPassword

```
HASDK_API int HASDK_CALL HA_SetCamSDKPassword(
struct HA_Cam* cam,
const char* password
);
```

Description:

Set camera side SDK authentication password

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
password	Sdk authentication password	In

Note:

By default, SDK password is not enabled on camera side.

Once the password is set, it can NOT be cleared or reset by configure tool. To reset the password, you have to contact dealer.

4.6 Registration Query

4.6.1 Query All Ids

HA_GetAllPersonId

```
HASDK_API int HASDK_CALL HA_GetAllPersonId(
    structHA_Cam *cam,
    char*PersonIdBuff,
    constintBuffsize,
    int* count,
    int *total
    );
```

Description:
    Query all person ids registered

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
PersonIdBuff	Pointer to the buffer to receive the id (20 bytes for each id)	Out
Buffsize	The size of the buffer	In
count	Count of ids in the buffer	Out
total	Total number registered	Out

Note:
    If the buffer is big enough to hold all ids, count is equals to total, otherwise count < total

4.6.2 Query Total Registration Count

HA_GetFaceIDTotal

```
    HASDK_API int HASDK_CALL HA_GetFaceIDTotal(HA_Cam* cam);
```

Description:
    Query the total number of registration

Returns:

Value	Description
Any value >0	The total number
Other value	Error code

4.6.3 Query by Category

HA_QueryByRole

```
HASDK_API int HASDK_CALL HA_QueryByRole(
HA_Cam* cam,
int role,
int page_no,
int page_size,
charfeatureFlags,
charimgFlags
);
```

Description:

Query by person category

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
role	The category 0: white name 1: normal 2: black name -1: all above	In
page_no	1-based page number	In
page_size	Page size	In
featureFlags	Whether include feature data in the result 0: no 1: yes	In
imgFlags	Whether include image in the result 0: no 1: yes	In

Note:

Result is passed through callback function

See also:

HA_RegFaceQueryCb

4.6.4 Query by criteria

HA_QueryFaceEx

```
HASDK_API int HASDK_CALL HA_QueryFaceEx(
HA_Cam* cam,
int role,
int page_no,
int page_size,
charfeatureFlags,
charimgFlags,
shortcondition_flag,
shortquery_mode,
structQueryCondition*condition
);
```

Description:

Query by criteria

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
role	category 0:white name 1:normal 2:black name -1:all	In
page_no	1-based page number	In
page_size	Page size	In
featureFlags	Whether include feature data 0:no 1:yes	In
imgFlags	Whether include image 0:no 1:yes	In
condition_flag	Flag indicating which field is enabled. see enumConditionFlag	In
query_mode	Query mode 0: exact 1: fuzzy	In

condition	Query criteria	In

Note:
    The result is passed by callback function

See also:
    HA_RegFaceQueryCb

Example:

```
        QueryConditionQ_Condition;    //query criteria
        memset(&Q_Condition,0,sizeof(QueryCondition));    //reset memory
    //fuzzy query
        bool vague = ((CButton*)GetDlgItem(IDC_RADIO3))->GetCheck();
        CStringstr_id;
        short condition=0; //query criteria
        GetDlgItem(IDC_EDIT2)->GetWindowText(str_id);
        CStringstr_name;
        GetDlgItem(IDC_EDIT4)->GetWindowText(str_name);
        char* id=NULL;
        char* name=NULL;
        USES_CONVERSION;
        if(!str_id.IsEmpty()) {
            id = T2A(str_id.GetBuffer(0));
            condition |= QUERY_BY_ID; //set id flag, bitwise OR is supported
            strcpy(Q_Condition.faceID,id);    //the ID to query
        }
        if(!str_name.IsEmpty()) {
            name = T2A(str_name.GetBuffer(0));
            condition |= QUERY_BY_NAME;    //set name flag
            strcpy(Q_Condition.faceName,name); //the name to query
        }
    QueryFaceComplete = false;  //query complete flag
int ret = HA_QueryFaceEx(g_cam[Using_cam].cam, -1, 1, 100, -1, 1,
condition, vague, &Q_Condition);
    //query result is pass through callback function, you register
callback with HA_RegFaceQueryCb function
    if (ret == ERR_NONE) {
        while(!QueryFaceComplete) { //whether query is finished
            Sleep(500);
        }
    }
printf("query finished\n");
```

4.6.5 Face Database Capacity Query

HA_StorageCapacity

```
HASDK_API int HASDK_CALL HA_StorageCapacity(
structHA_Cam *cam,
struct StorageCapacity*infor
);
```

Description:

Query face template database capacity

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
infor	Pointer to buffer to receive result	Out

4.7 Registration

Terms:

Feature data

A series of binary data of face features calculated from the registration image after algorithm processing inside the camera. During face comparison, we can quickly judge whether it is the same person or not through the feature data. The feature data extracted from the same face with different algorithm versions are different, so the feature data can only be extracted inside the camera, and can be extracted between cameras of the same algorithm version The machine of a characteristic data is directly registered into the machine of B.

Normalized image

Before extracting features, the camera will first zoom the photo according to the face position, and map it to a fixed size image with only face. The normalized image can be extracted inside the camera or by using the SDK. Except for the new ev200 algorithm, all versions of normalized images can be registered with each other.

Abbreviated image

In order to reduce the data distribution during registration and reduce the memory occupation of camera face database, the original image of registration will not be stored in the camera. When storing personnel data, the original image will be cropped and scaled to a small resolution face abbreviated image for display only. Different from the normalized image, the thumbnail does not contain face details and will not participate in the comparison process If there is no interface display requirement, the abbreviated image can be omitted.

Note:
The registration interface will take time to extract the normalized image. If the same registered image is distributed to multiple cameras, the normalized image can be extracted first and then the normalized image registration interface can be called. If the processor of the SDK operating environment is weak and the extraction time is high, the JPG image direct registration interface can be called.

4.7.1 Auto-clean Expired Registration

HA_GetAutoCleanEnable

```
HASDK_API int HASDK_CALL HA_GetAutoCleanEnable(
structHA_Cam* cam,
int *enable
);
```

Description:
Get whether auto clean expired face registration is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enabled or disabled 0: disabled 1: enabled	Out

HA_SetAutoCleanEnable

```
HASDK_API int HASDK_CALL HA_SetAutoCleanEnable(
    structHA_Cam* cam,
    int enable
    );
```

Description:
    Enable or disable auto clean expired face registration

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable 0: disable 1: enable	In

4.7.2 Register by Image Path

HA_AddJpgPaths

```
    HASDK_API int HASDK_CALL HA_AddJpgPaths(
    HA_Cam* cam,
    structFaceFlags*faceID,
    char *jpg[],
    int img_count,
    int picture_Flags
    );
```

Description:
    Register by image path

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
faceID	Face info	In
jpg	Array of image path (.jpg,.bmp, .png are supported)	In
img_count	the size of the array	In
picture_Flags	Value indicating whether download minimized image to camera	In

	0: no 1: yes	

Note:

    Failed registration of either image will fail the function.
    The max size of image should be <= 10MB

Example:

```
FaceFlagsfaceID;
memset(&faceID, 0, sizeof(faceID)); //reset memory
strcpy(faceID.faceID, "123456"); //set   face ID
strcpy(faceID.faceName, "Halle"); //set name
faceID.role = 1; //set category
char* jpg[5] = {NULL};//max 5 images
jpg[0] = "1.bmp"; //image path
faceID.effectTime = 0xFFFFFFFF; //valid to time: never expire
int ret= HA_AddJpgPaths(g_cam[Using_cam].cam,&faceID,jpg,1,1);
if (ret == ERR_NONE) {
    printf("success");
}
else
    printf("fail,error code=%d\n", ret);
```

4.7.3 Register by Image Buffer

HA_AddJpgFaces

```
HASDK_API int HASDK_CALL HA_AddJpgFaces(
HA_Cam* cam,
structFaceFlags*faceID,
structFaceImage*imgs,
int img_count,
int picture_flags
);
```

Description:

    Register by image buffer

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
faceID	Face info	In
imgs	Array of image buffer (.jpg, .bmp, .png are supported)	In
img_count	Size of array	In
picture_Flags	Value indicating whether download minimized image to camera 0: no 1: yes	In

Note:

Failed registration of either image will fail the function.

The max size of image should be <= 10MB

Example:

```
FILE* fp = fopen("1.bmp", "rb");
if(!fp)
    return;
size_t size = _filelength(_fileno(fp)); //file size
unsignedchar *data = newunsignedchar[size];    //image buffer
fread(data, size, 1, fp);
fclose(fp);
FaceImageimg;
memset(&img, 0, sizeof(img));
img.img_fmt = 0; //non-bgr
img.img_len = size; //image size
img.img = data; //image buffer

FaceFlagsfaceID;
memset(&faceID, 0, sizeof(faceID));
strcpy(faceID.faceID, "123456"); //face id
strcpy(faceID.faceName, "Halle"); //name
faceID.role = 1; //category
faceID.effectTime = 0xFFFFFFFF; //valid to time: never expire


int ret= HA_AddJpgFaces(g_cam[Using_cam].cam,&faceID, &img,1,1);
if (ret == ERR_NONE) {
    printf("success");
}
```

```
    else
        printf("failed, error code=%d\n", ret);
```

HA_AddFaces

```
HASDK_API int HASDK_CALL HA_AddFaces(
HA_Cam* cam,
structFaceFlags*faceID,
structFaceImage*imgs,
int img_count,
int picture_flags
);
```

Description:

Register by image buffer in RGB format

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
faceID	Face info	In
imgs	Image buffer array Only decoded RGB data is supported	In
img_count	Size of array	In
picture_Flags	Value indicating whether download minimized image to camera 0: no 1: yes	In

Note:

Failed registration of either image will fail the function.
Only decoded RGB data is supported

4.7.4 Update Registration

HA_JpgPathsEx

```
    HASDK_API int HASDK_CALL HA_JpgPathsEx(
    HA_Cam* cam,
    int type,
    struct FaceFlags *faceID,
    char *jpg[],
    int img_count,
    int picture_Flags,
    struct ErrorFaceImage *err_imgs,
    int *err_imgs_count
    );
```

Description:

Update or register by image path

The function is considered success if either one image is registered or updated successfully

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
type	Value indicating what function to perform 0: register 1: update	In
faceID	Face info	In
jpg	Array of image path (.jpg, .bmp, .png are supported)	In
img_count	Size of image path array	In
picture_Flags	Whether download minimized image to camera 0: no 1: yes	In
err_imgs	Error code array for image	Out
err_imgs_count	Size of error code array	Out

Example:

```
char* patch[5] = {NULL}; //image path array
patch[0] = "1.jpg";
patch[1] = "1.bmp";
patch[2] = "1.png";
```

```
            int ret = 0;
            FaceFlagsfaceID;
            memset(&faceID, 0, sizeof(faceID)); //reset memory
            strcpy(faceID.faceID, "123456"); //face ID
            strcpy(faceID.faceName, "Halle"); //name
            faceID.role = 1; //category
            faceID.effectTime = 0xFFFFFFFF; //valid to:    never expire
            int type = 0; //action,    0: register    1: update
            ErrorFaceImageerrimgs;
            memset(&errimgs, 0, sizeof(errimgs)); //reset memory
            interrimgcount=0; //erro code array size
            ret= HA_JpgPathsEx(g_cam[Using_cam].cam, type,&faceID,
    patch,3,1, &errimgs, &errimgcount);
            if (ret == ERR_NONE) {
                printf("success");
            }
            else
                printf("fail, error code=%d\n", ret);
            for (inti = 0; i<errimgcount; i++) {
                printf("image index =%d,error code=%d \n", errimgs.img_seq,
    errimgs.err_code);
                }
```

HA_JpgFacesEx

```
ASDK_API int HASDK_CALL HA_JpgFacesEx(
HA_Cam* cam,
int type,
structFaceFlags*faceID,
structFaceImage*imgs,
int img_count,
int picture_flags,
struct ErrorFaceImage *err_imgs,
int *err_imgs_count
);
```

Description:

Update or register by image path

The function will succeed if either one image is registered or updated successfully

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
type	action 0: register 1: update	In
faceID	Face info	In
imgs	Array of image path (.jpg, .bmp, .png are supported)	In
img_count	Size of image path array	In
picture_Flags	Whether download minimized image to camera 0: no 1: yes	In
err_imgs	Error code array	Out
err_imgs_coun t	Size of error code array	Out

HA_FacesEx

```
HASDK_API int HASDK_CALL HA_FacesEx(
HA_Cam* cam,
int type,
structFaceFlags*faceID,
structFaceImage*imgs,
int img_count,
int picture_flags,
structErrorFaceImage*err_imgs,
int *err_imgs_count
);
```

Description:

Register or update by RGB data

The function succeed if either one image succeed

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
type	action 0: register 1: update	In
faceID	Face info	In
imgs	Image data array	In

	Only support decoded RGB data	
img_count	Size of image data array	In
picture_Flags	Whether download minimized image to camera 0: no 1: yes	In
err_imgs	Error code array	Out
err_imgs_coun t	Size of error code array	Out

Note:
  Only support decoded RGB data

4.7.5 Delete Registration

HA_DeleteFaceDataByPersonID

```
HASDK_API int HASDK_CALL HA_DeleteFaceDataByPersonID(
HA_Cam* cam,
constchar* personID
);
```

Description:
  Delete registration by id

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
faceID	The registration with the id to delete	In

HA_DeleteFaceDataByPersonRole

```
HASDK_API int HASDK_CALL HA_DeleteFaceDataByPersonRole(
HA_Cam* cam,
int role
);
```

Description:

Delete registration by category

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
role	Registration of category to delete	In

Note:
All registration with the specified category will be deleted. The deletion progress can be notified though HA_RegFaceDeleteProgressCb callback function.

See also:
HA_RegFaceDeleteProgressCb

HA_DeleteFaceDataAll

```
HASDK_API int HASDK_CALL HA_DeleteFaceDataAll(HA_Cam* cam);
```

Description:
Delete all registration.

Note:
The deletion progress can be notified though HA_RegFaceDeleteProgressCb callback function.

See also:
HA_RegFaceDeleteProgressCb

4.7.6 Compound Registration

HA_FaceSyncInterface

```
HASDK_API int HASDK_CALL HA_FaceSyncInterface(
HA_Cam* cam,
structFaceFlags*faceID,
structFaceImage*imgs,
int img_count,
int picture_flags,
int syncFlag
);
```

Description

The general function that can update, register and delete face

Arguments:

Argument	Description:	In/Out
cam	Handle to camera	In
faceID	Face info	In
imgs	Image data array Only decoded RGB data is supported	In
img_count	Size of image array	In
picture_Flags	Whether download minimized image to camera 0: no 1: yes	In
syncFlag	Value indicating what action to perform 1: register 2: update 3: delete	In

Note:

For deletion, only id field is required

4.7.7 Normalized Image Registration

HA_FacePacketSync

```
HASDK_API int HASDK_CALL HA_FacePacketSync(
HA_Cam* cam,
structFaceFlags*faceID,
structFaceFeature* feature,
structFaceImage* twist_imgs,
int twist_num,
structFaceImage*imgs,
    int img_num
    );
```

Description:

  Register with normalized image

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
faceID	Face id	In
feature	Feature data, can be null	In
twist_imgs	Normalized images	In
twist_num	Count of normalized images	In
imgs	Face images	In
img_num	Coun of face images	In

Note:

  This function does not extract features directly, and it takes a short time to register the normalized graphics or feature data

  To extract normalized image, see extract face feature / normalized image

  When register to multiple cameras, you can extract features before calling this function to register to avoid multiple extraction features

Example:

```
HA_Cam* cam = g_cam[Using_cam].cam; // the handle to connected camera

unsignedcharjpg[1024 * 100];
unsignedcharfeature_image[1024 * 110];
unsignedcharface_img_jpg[1024 * 10];
unsignedchartwist_image_buff[1024 * 70];
intface_jpg_len = 1024 * 10;
intfeature_size = 1024 * 110;

FILE* fp = fopen("./1.jpg", "rb");
if (fp == NULL) {
        return;
    }
    intjpg_len = fread(jpg, 1, 1024 * 100, fp);
    fclose(fp);

    int ret = HA_GetJpgFeatureImageNew(jpg, jpg_len, feature_image, &feature_size,
face_img_jpg, &face_jpg_len, NULL);
    if (ret != 0) {
        return;
    } // extract feature data with new algorithm

    FaceImage face_image = { 0 }; // the face image
    face_image.img = face_img_jpg;
    face_image.img_len = face_jpg_len;

    FaceImagetwist_image = { 0 }; // normalized image
    twist_image.img_fmt = 1; // normalized image format, RGB
    twist_image.img = twist_image_buff;
    twist_image.img_len = 1024 * 70;  // the buffer size

    ret = HA_FeatureConvert(cam, feature_image, feature_size, &twist_image, NULL);
    if (ret != 0) {
        return;
    } // convert to registerable normalized image

    FaceFlagsface_flags = { 0 };
    strcpy(face_flags.faceName, "mike");
    strcpy(face_flags.faceID, "110");
    face_flags.effectTime = 0xFFFFFFFF; //never expire
    face_flags.role = 1;

    ret = HA_FacePacketSync(cam, &face_flags, NULL, &twist_image, 1, &face_image, 1);
    if (ret == 0) {
        printf("register succeed");
    }
```

4.7.8 Register by JPEG Image

HA_AddFacesByJpg

```
HASDK_API int HASDK_CALL HA_AddFacesByJpg(
HA_Cam* cam,
structFaceFlags*faceID,
structFaceImage*imgs,
    int img_num
    );
```

Description:
    Register by jpg image

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
faceID	Face info	In
imgs	Jpg image data	In
img_num	Fixed to 1	In

Note:
    Only CV500, DV300 are supported

HA_ModifyFacesByJpg

```
    HASDK_API int HASDK_CALL HA_ModifyFacesByJpg(
    HA_Cam* cam,
    structFaceFlags*faceID,
    structFaceFlags*imgs,
    int img_num
    );
```

Description:
    Update by jpg image

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
faceID	Face info	In
imgs	Jpg image data	In
img_num	Fixed to 1	In

Note:
    Only support CV500, DV300

4.7.8 Update Single Registration

HA_SeparateModifyFace

```
HASDK_API int HASDK_CALL HA_SeparateModifyFace(
HA_Cam* cam,
struct FaceFlags *faceID,
struct FaceFlags * twist_imgs,
unsigned inttwist_num,
struct FaceFlags *imgs,
unsigned intimg_num,
unsigned intupdate_flags
);
```

Description:

Update registration

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
faceID	Face id	In
twist_imgs	Normalized image array null, if don't update	In
twist_num	Size of normalized image array	In
imgs	Thumbnail image array null, it don't update	In
img_num	Size of thumbnail image array	In
update_flags	Update flags See ParsonDataFlags, bitwise OR operation is supported	In

4.9 Face Capture Record Persistence

4.9.1 Face Capture Record Persistence Config

HA_GetRecorderEnable

```
HASDK_API int HASDK_CALL HA_GetRecorderEnable(
structHA_Cam* cam,
char* enable
);
```

Description:
    Get value indicating whether captured record will be saved
  to storage

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Whether the feature is enabled 0: disabled 1: enabled	Out

HA_SetRecorderEnable

```
    HASDK_API int HASDK_CALL HA_SetRecorderEnable(
    structHA_Cam* cam,
    char enable
    );
```

Description:
    Enable or disable saving capture record to storage

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable 0: disable 1: enable	In

4.9.2 Query Face Capture Record

HA_QueryFaceRecord

```
    HASDK_API int HASDK_CALL HA_QueryFaceRecord(
    HA_Cam* cam,
    int page_no,
    int page_size,
    structRecordCondition*condition
    );
```

Description:

    Query face capture records

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
page_no	1-based page number	In
page_size	Page size 100maximum	In
condition	Criteria for the query	In

Note:

    Results are passed back through callback function

See also:

    HA_RegFaceRecordQueryCb

Example:

```
        RecordCondition condition;
        memset(&condition, 0, sizeof(condition));
        condition.img_flag = 0;//don't include image in the query result
        condition.query_mode = 1;//fuzzy query
        strcpy(condition.person_id, "1234"); //id
        condition.condition_flag |= RECORD_QUERY_FLAG_ID;//enable id field
        strcpy(condition.person_name, "Hal"); //name
        condition.condition_flag |= RECORD_QUERY_FLAG_NAME; //enable
name field
        HA_QueryFaceRecord(g_cam[Using_cam].cam, 1, 100, &condition);
```

4.9.2 Query Face Capture Record Storage

HA_QueryCapacityHistory

```
    HASDK_API int HASDK_CALL HA_QueryCapacityHistory(
    HA_Cam* cam,
    CapacityHistory* infor
    );
```

Description:

    Query history capture record storage info

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
infor	The info	Out

4.9.3 Clean Up Face Capture Record

HA_DeleteFaceRecordBySequence

```
HASDK_API int HASDK_CALL HA_DeleteFaceRecordBySequence(
HA_Cam* cam,
unsignedintsequence
);
```

Description:
Delete capture record by sequence number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
sequence	The sequence number	In

HA_DeleteFaceRecordAll

```
   HASDK_API int HASDK_CALL HA_DeleteFaceRecordAll(HA_Cam*
cam);
```

Description:
Delete all capture record

4.10 Snapshot and Video

HA_CapImgToFile

```
HASDK_API int HASDK_CALL HA_CapImgToFile(
structHA_Cam* cam,
constchar* fileName
);
```

Description:

Take a snapshot and save it to specified file in jpg format

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
fileName	The file name	In

Note:

Grab snapshot from video stream, there might be some delays. The function will not create directory.

HA_CapImgToBuffer

```
HASDK_API int HASDK_CALL HA_CapImgToBuffer(
structHA_Cam* cam,
unsignedchar* buffer,
int bufferSize,
int* len
);
```

Description:

Take a snapshot and save it to memory buffer in jpg format

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
buffer	The buffer to receive the image	In
bufferSize	The length of buffer	In
len	The length of the image	Out

Note:

Allocate a buffer that is bigger than 2 MB. The snapshot is grabbed from video stream, there might be some delays

HA_Snapshot

```
HASDK_API int HASDK_CALL HA_Snapshot(HA_Cam* cam);
```

Description:
    Take snapshot from camera
Note:
    The snapshot is passed by callback function

See also:
    HA_RegSnapshotCb

HA_SaveRealDate

```
HASDK_API int HASDK_CALL HA_SaveRealDate(
HA_Cam *cam ,
char *sFileName
);
```

Description:
    Begin recording video stream to specified file in avi format

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
sFileName	The file name	In

Note:
    No directory will be created

HA_StopSaveRealDate

```
HASDK_API int HASDK_CALL HA_StopSaveRealDate(HA_Cam *cam);
```

Description:
    Stop recording video stream

Note:
    No directory will be created

HA_SaveRealDate1

```
HASDK_API int HASDK_CALL HA_SaveRealDate1(
HA_Cam *cam ,
char *sFilePath
);
```

Description:
    Begin recording infrared video steam in avi format

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
sFileName	The file name	In

Note:
No directory will be created

HA_StopSaveRealDate1

```
   HASDK_API int HASDK_CALL HA_StopSaveRealDate1(HA_Cam
*cam);
```

Description:
Stop recording infrared video stream

4.11 Serial Port

4.11.1 RS485

HA_GetSerialConfigServiceEnable

```
   HASDK_API int HASDK_CALL HA_GetSerialConfigServiceEnable(
   structHA_Cam* cam,
   int index,
   char *enable
   );
```

Description:
Value that indicats whether serial port is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
index	The index of serial port HA_SERIA_RS485 HA_SERIA_RS232	In
enable	0: disabled non-zero: enabled	Out

HA_SetSerialConfigServiceEnable

```
    HASDK_API int HASDK_CALL HA_SetSerialConfigServiceEnable(
    structHA_Cam* cam,
    int index,
    charenable
    );
```
Description:

Enable serial port

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
index	Serial port number HA_SERIA_RS485 HA_SERIA_RS232	In
enable	0: disable Non-zero: enable	In

HA_GetRS485ProtocalNo

```
    HASDK_API int HASDK_CALL HA_GetRS485ProtocalNo(
    structHA_Cam* cam,
    char *rs485_protocal_no
    );
```
Description:

Get 485Output protocol number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rs485_protocal_no	Output protocol number	Out

HA_SetRS485ProtocalNo

```
    HASDK_API int HASDK_CALL HA_SetRS485ProtocalNo(
    structHA_Cam* cam,
    char rs485_protocal_no
    );
```

Description:
    Set 485 output protocol number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
rs485_protocal_no	Output protocol number	In

4.11.2 Transparent Serial Port

HA_OpenTSerial

```
HASDK_API int HASDK_CALL HA_OpenTSerial(
structHA_Cam* cam,
int index,
int baudrate,
int parity,
int databit,
int stopbit
);
```

Description:
    Open transparent serial port with specific parameters

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
index	The index of serial port HA_SERIA_RS485 HA_SERIA_RS232	In
baudrate	Baundrate One of the following value : 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000	In

parity	Parity 0:none, 1:odd, 2:even, 3:mark, 4:space	In
databit	Databit Can only be 5， 6， 7， 8	In
stopbit	Stopbit Can only be 1， 2	In

HA_GetTSerial

```
HASDK_API int HASDK_CALL HA_GetTSerial(
structHA_Cam* cam,
int index,
int* baudrate,
int* parity,
int* databit,
int* stopbit
);
```

Description:

Get transparent port parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
index	The index of transparent setial port HA_SERIA_RS485 HA_SERIA_RS232	In
baudrate	Baundrate Can only be： 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000	Out
parity	Parity 0:none, 1:odd,	Out

	2:even, 3:mark, 4:space	
databit	Databit Can only be 5, 6, 7, 8	Out
stopbit	Stopbit Can only be 1, 2	Out

4.11.3 Write Transparent Serial Port

HA_WriteTSerial

```
HASDK_API int HASDK_CALL HA_WriteTSerial(
structHA_Cam* cam,
int index,
constunsignedchar* data,
int size
);
```

Description:
Write data to transparent serial port

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
index	Index to the serial port HA_SERIA_RS485 HA_SERIA_RS232	In
data	Data to write	In
size	Size of the data	In

See also:
HA_RegReadTSerialCbEx
HA_RegReadTSerialCb

4.12 Upload

4.12.1 Upload Method

HA_GetUploadConfig

```
HASDK_API int HASDK_CALL HA_GetUploadConfig(
structHA_Cam* cam,
structClientParam*UploadParam
);
```

Description:
    Get upload configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
UploadParam	Upload parameter	Out

Note:
    Only one upload method can be enabled

HA_SetUploadConfig

```
HASDK_API int HASDK_CALL HA_SetUploadConfig(
structHA_Cam* cam,
structClientParam*UploadParam
);
```

Description:
    Set upload parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
UploadParam	Upload parameter	In

Note:
    Only one upload method can be enabled

4.12.2 Server Address

HA_GetExtranetParam

```
HASDK_API int HASDK_CALL HA_GetExtranetParam(
struct HA_Cam* cam,
ExtranetParam* Param
);
```

Description:
    Get server configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
Param	Server configure	Out

HA_SetExtranetParam

```
HASDK_API int HASDK_CALL HA_SetExtranetParam(
struct HA_Cam* cam,
ExtranetParam* Param
);
```

Description:
    Set server configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
Param	Server configure	In

Note:
    Only one upload method can be enabled, TCP or http

4.12.3 Break Point Resume

HA_GetRecorderResumeEnable

```
HASDK_API int HASDK_CALL HA_GetRecorderResumeEnable(
struct HA_Cam* cam,
char* enable
);
```

Description:
Get whether break point resume is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Whether break point resume is enabled Non-zero: enabled 0 : disabled	Out

HA_SetRecorderResumeEnable

```
HASDK_API int HASDK_CALL HA_SetRecorderResumeEnable(
struct HA_Cam* cam,
charenable
);
```

Description:
Enable or disable break point resume

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable or disable Non-zero: enable 0: disable	In

4.12.4 Encrypted Upload

HA_SetCamAESEncode

```
HASDK_API int HASDK_CALL HA_SetCamAESEncode(
struct HA_Cam* cam,
const unsigned char* oriKey,
unsigned  intoriKeyLen,
const char*newKey,
unsigned intnewKeyLen,
char enable
);
```

Description:

Enable or disable encryption of name and id when upload record

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
oriKey	Old encryption key, null for initialization for the first time	In
oriKeyLen	Old encryption key length	In
newKey	New encryption key, null to keep it unchanged	In
newKeyLen	New encryption key length	In
enable	Enable or disable -1: keep it unchanged 0: disable 1: enable	In

Note:

The encryption parameter, AES padd with 0X00, CBC mode VI vector 0X00

HA_GetCamAESEncode

```
HASDK_API int HASDK_CALL HA_GetCamAESEncode(
struct HA_Cam* cam,
char* enable
);
```

Description:
Get whether name and id encryption is enabled when uploading

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enabled or disabled Non-zero: enabled 0: disable	Out

4.13 Interfaces and Peripherals

4.13.1 Gate Access Control

Gate Access Schedule

HA_SetScheduleModeCfg

```
HASDK_API int HASDK_CALL HA_SetScheduleModeCfg(
    struct HA_Cam* cam,
    const struct KindSchedule* cfg,
    int ScheduleCount
    );
```

Description:
Set gate access control rules

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	Gate access control rules array	In
ScheduleCount	Rules count Max 5 rules	In

Note:

Rule are specified when register face

HA_GetScheduleModeCfg

```
HASDK_API int HASDK_CALL HA_SetScheduleModeCfg(
    struct HA_Cam* cam,
    struct KindSchedule* cfg,
    int* ScheduleCount
    );
```

Description:

Get gate access control rules

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	Gate access control rule array Max 5 rules	Out
ScheduleCount	Rules count Max 5 rules	Out

HA_GetScheduleModeCfgEx

```
HASDK_API int HASDK_CALL HA_SetScheduleModeCfg(
    struct HA_Cam* cam,
    struct KindSchedule* cfg,
    int* ScheduleCount
    );
```

Description:

Extended function to set gate access control rules

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	Gate access control rules array Max 15 rules	Out
ScheduleCount	Size of the rule array In: size of rule array Out: size of rule array	In/Out

Holiday Configure

HA_SetScheduleFestival

```
HASDK_API int HASDK_CALL HA_SetScheduleFestival(
    struct HA_Cam* cam,
    const struct ScheduleFestival* cfg,
    );
```

Description:
    Set holiday, during which gate won’t open

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	Holiday parameter	In

HA_GetScheduleFestival

```
HASDK_API int HASDK_CALL HA_GetScheduleFestival(
    struct HA_Cam* cam,
    struct ScheduleFestival* cfg,
    );
```

Description:
    Get holiday configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cfg	Holiday configure	Out

GPIO Output

HA_GetGatewayControlType

```
HASDK_API int HASDK_CALL HA_GetGatewayControlType(
structHA_Cam* cam,
int *type
);
```

Description:

Get gate access control output method

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
type	0: electric relay 1: wiegand	Out

Note:

The setting only applies to automatic gate control when face is captured, which can be overwite through sdk.

HA_SetGatewayControlType

```
HASDK_API int HASDK_CALL HA_SetGatewayControlType(
structHA_Cam* cam,
int type
);
```

Description:

Set gate open method

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

type	0: electric relay 1: wiegand	In

HA_GetWiegandType

```
HASDK_API int HASDK_CALL HA_GetWiegandType(
structHA_Cam* cam,
int *type
);
```

Description:
Get access control device wiegandencoding

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
type	Wiegand encoding See GpioInType Only WG26, WG34 are supported	Out

Note:
The access control device type shoule be set to wiegand

HA_SetWiegandType

```
HASDK_API int HASDK_CALL HA_SetWiegandType(
structHA_Cam* cam,
int type
);
```

Description:
Set access control device wiegand encoding

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
type	Wiegand encoding See GpioInType Only WG26, WG34 are	In

	supported	

Note:
    Access control device type should be set to wiegand device

GPIO Input

HA_GetGpioInputEnable

```
HASDK_API int HASDK_CALL HA_GetGpioInputEnable(
structHA_Cam* cam,
char *enable
);
```

Description:
    Get whether GPIO input is enabled

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether gpio input is enabled 0: disabled 1: enabled	Out

See also:
HA_RegGpioInputCb

HA_SetGpioInputEnable

```
HASDK_API int HASDK_CALL HA_SetGpioInputEnable(
structHA_Cam* cam,
charenable
);
```

Description:
    Enable or disable GPIO input

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Value indicating whether GPIO input is enabled 0: disable 1: enable	Out

HA_GetGpioInputType

```
HASDK_API int HASDK_CALL HA_GetGpioInputType(
structHA_Cam* cam,
int *type
);
```

Description:
    Get GPIO input type

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
type	GPIO input type See GpioInType	Out

HA_SetGpioInputType

```
HASDK_API int HASDK_CALL HA_SetGpioInputType(
structHA_Cam* cam,
int type
);
```

Description:
    Set GPIO input type

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
type	GPIO input type See GpioInType	In

Electric Relay Configure

HA_GetGpioWorkState

```
    HASDK_API int HASDK_CALL HA_GetGpioWorkState(
    structHA_Cam* cam,
    unsignedchar *state
    );
```

Description:
    Get electric relay working mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
state	1: normally open 0: normally close	Out

HA_SetGpioWorkState

```
    HASDK_API int HASDK_CALL HA_SetGpioWorkState(
    structHA_Cam* cam,
    unsignedchar state
    );
```

Description:
    Set electric relay working mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
state	1: normally open 0: normally close	In

HA_GetAlarmDuration

```
HASDK_API int HASDK_CALL HA_GetAlarmDuration(
structHA_Cam* cam,
int *duration
);
```

Description:
Get electric relay close duration

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
duration	The duration (500-5000ms)	Out

HA_SetAlarmDuration

```
HASDK_API int HASDK_CALL HA_SetAlarmDuration(
structHA_Cam* cam,
int duration
);
```

Description:
Set electric relay close duration

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
duration	The duration (500-5000ms)	In

GPIO Output Control

HA_SetGPIO

```
HASDK_API int HASDK_CALL HA_SetGPIO(
structHA_Cam* cam,
int port,
int inout,
int onoff,
char *resv
);
```

Description:
    Set GPIO output

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
port	GPIO port number 0: wiegand device Non-zero: other GPIO device	In
inout	Fixed value: 2	In
onoff	Fixed value: 1	In
resv	Byte array indicating control method resv[0] control method 0: test (send open gate signal directly) 1: force 2 : cooperative (depends on whether device is enabled) resv[1-4]-->uint:wiegand card number resv[5-24]-->uint: face id	

Public Access Card

HA_GetWiegandPublicCardNO

```
    HASDK_API int HASDK_CALL HA_GetWiegandPublicCardNO(
    structHA_Cam* cam,
    unsignedint*cardNo
    );
```

Description:
    Get public access card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

cardNo	The card number	Out

HA_SetWiegandPublicCardNO

```
    HASDK_API int HASDK_CALL HA_SetWiegandPublicCardNO(
    structHA_Cam* cam,
    unsignedintcardNo
    );
```

Description:

Set public access card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cardNo	The card number	In

HA_GetWiegandAutoCardNoMin

```
    HASDK_API int HASDK_CALL HA_GetWiegandAutoCardNoMin(
    structHA_Cam* cam,
    unsignedint*cardNoMin
    );
```

Description:

Get the minimal value of automatically generated wiegand card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cardNoMin	The minimal value	Out

HA_SetWiegandAutoCardNoMin

```
    HASDK_API int HASDK_CALL HA_SetWiegandAutoCardNoMin(
    structHA_Cam* cam,
    unsignedintcardNoMin
    );
```

Description:
    Set the minimal value of automatically generated wiegand card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cardNoMin	The minimal value	In

HA_GetWiegandAutoCardNoMax

```
    HASDK_API int HASDK_CALL HA_GetWiegandAutoCardNoMax(
    structHA_Cam* cam,
    unsignedint*cardNoMax
    );
```

Description:
    Get the maximal value of automatically generated wiegand card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cardNoMax	The max value	Out

HA_SetWiegandAutoCardNoMax

```
    HASDK_API int HASDK_CALL HA_SetWiegandAutoCardNoMax(
    structHA_Cam* cam,
    unsignedintcardNoMax
    );
```

Description:
    Set the maximal value of automatically generated wiegand card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cardNoMax	The max value	In

Camera Work Mode

HA_GetCameraWorkMode

```
HASDK_API int HASDK_CALL HA_GetCameraWorkMode(
structHA_Cam* cam,
unsignedchar *work_mode
);
```

Description:
    Get working mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
work_mode	Working mode 1: automatic 2: online 3: offline	Out

HA_SetCameraWorkMode

```
HASDK_API int HASDK_CALL HA_SetCameraWorkMode(
structHA_Cam* cam,
unsignedcharwork_mode
);
```

Description:
    Get working mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

work_mode	Working mode 1: automatic 2: online 3: offline	In

Camera Matching Mode

HA_GetMatchMode

```
    HASDK_API int HASDK_CALL HA_GetMatchMode(
    structHA_Cam* cam,
    unsignedchar *mode
    );
```

Description:
    Get matching mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
mode	Matching mode See MatchMode	In

HA_SetMatchMode

```
    HASDK_API int HASDK_CALL HA_SetMatchMode(
    structHA_Cam* cam,
    unsignedchar mode
    );
```

Description:
    Set matching mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
mode	Matching mode See MatchMode	In

Force Gate Open

HA_WhiteListAlarm

```
HASDK_API int HASDK_CALL HA_WhiteListAlarm(
structHA_Cam *cam,
int inout,
int onoff
);
```

Description:
    Force gate open by whitelist

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
inout	Fixed value:2	In
onoff	Fixed value:1	In

HA_BlackListAlarm

```
HASDK_API int HASDK_CALL HA_BlackListAlarm(
structHA_Cam *cam,
int inout,
int onoff
);
```

Description:
    Force gate open by blacklist

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
inout	Fixed value:2	In
onoff	Fixed value:1	In

HA_WiegandAlarm

```
    HASDK_API int HASDK_CALL HA_WiegandAlarm(
    structHA_Cam *cam,
    unsignedintwiegand_no
    );
```

Description:
    Force gate open by wiegand card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
wiegand_no	Wiegand card number	In

Trigger Gate Open

HA_WhiteListAlarmEx

```
    HASDK_API int HASDK_CALL HA_WhiteListAlarmEx(
    structHA_Cam *cam,
    int inout,
    int onoff,
    unsignedcharalarm_mode,
    unsignedchar *person_id
    );
```

Description:
    Force gate open by whitelist

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
inout	Fixed value:2	In
onoff	Fixed value:1	In
alarm_mode	0: test (send open gate signal directly) 1: force 2: cooperative (depends on whether device is enabled)	In
person_id	Face id	In

HA_BlackListAlarmEx

```
HASDK_API int HASDK_CALL HA_BlackListAlarmEx(
structHA_Cam *cam,
int inout,
int onoff,
unsignedcharalarm_mode,
unsignedchar *person_id
);
```

Description:
    Force gate open by blacklist

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
inout	Fixed value:2	In
onoff	Fixed value:1	In
alarm_mode	0: test (send open gate signal directly) 1: force 2: cooperative (depends on whether device is enabled)	In
person_id	Face id	In

HA_WiegandAlarmEx

```
HASDK_API int HASDK_CALL HA_WiegandAlarmEx(
structHA_Cam *cam,
unsignedintwiegand_no,
unsignedcharalarm_mode,
unsignedchar *person_id
);
```

Description:
    Force gate open by wiegand card number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
wiegand_no	Wiegand card number	In
alarm_mode	0: test (send open gate	In

	signal directly) 1: force 2: cooperative (depends on whether device is enabled)	
person_id	Face id	In

4.13.2 OSD

Get Display Title

HA_GetScreenOsdTitle

```
    HASDK_API int HASDK_CALL HA_GetScreenOsdTitle(
    structHA_Cam* cam,
    char *screen_title
    );
```

Description:
The title of OSD

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
screen_title	The title of OSD, max 64 bytes UTF8-encoding	Out

HA_SetScreenOsdTitle

```
    HASDK_API int HASDK_CALL HA_SetScreenOsdTitle(
    structHA_Cam* cam,
    char *screen_title
    );
```

Description:
Set OSD title

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
screen_title	The title of OSD, max 64 bytes UTF8-encoding	In

Display Bright Level

HA_GetLcdLightLevel

```
    HASDK_API int HASDK_CALL HA_GetLcdLightLevel(
    struct HA_Cam* cam,
    char* Level
    );
```

Description:
    Get the bright level of display

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
Level	The bright level	Out

HA_SetLcdLightLevel

```
    HASDK_API int HASDK_CALL HA_SetLcdLightLevel(
    struct HA_Cam* cam,
    char Level
    );
```

Description:
    Set bright level of display

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
Level	The bright level	In

Display CSC Parameter

HA_GetLayerCSC

```
HASDK_API int HASDK_CALL HA_GetLayerCSC(
struct HA_Cam* cam,
char* csc
);
```

Description:
    Get Video CSC parameters

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
csc	Array to receive value char[0]: brightness char[1]: contrast char[2]: chroma char[3]: saturation	Out

HA_SetLayerCSC

```
HASDK_API int HASDK_CALL HA_SetLayerCSC(
struct HA_Cam* cam,
char csc,
int index
);
```

Description:
    Set video CSC parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

value	The value to set (range: 0-100)	In
index	The target item to config 0: brightness 1: contrast 2: chroma 3: saturation	In

Configure Display Items

HA_GetLcdDisplayItems

```
    HASDK_API int HASDK_CALL HA_GetLcdDisplayItems(
    struct HA_Cam* cam,
    char* item
    );
```

Description:
    Get OSD display items configure

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
item	The items that is displayed, bitwise OR operator is supported. See LcdDisplayItem	Out

Privacy Configuration

HA_SetNamePrivacy

```
    HASDK_API int HASDK_CALL HA_SetNamePrivacy(
    struct HA_Cam* cam,
    charenable
    );
```

Description:

Enable/disable name masking

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
enable	Enable/disable 0: disable 1: enable	In

HA_SetLcdDisplayItems

```
HASDK_API int HASDK_CALL HA_SetLcdDisplayItems(
struct HA_Cam* cam,
char item
);
```

Description:
Set OSD display items

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
item	The items to display. Bitwise OR operator is supported, see LcdDisplayItem	In

4.13.3 LED Control

HA_GetLedMode

```
HASDK_API int HASDK_CALL HA_GetLedMode(
structHA_Cam* cam,
char *led_mode
);
```

Description:
Get led working mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
led_mode	1：always on 2：auto, based on the scene brightness 3：always off	Out

HA_SetLedMode

```
HASDK_API int HASDK_CALL HA_SetLedMode(
structHA_Cam* cam,
charled_mode
);
```

Description:
    Set led working mode

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
led_mode	1：always on 2：auto, based on the scene brightness 3：always off	In

HA_GetLedLevel

```
HASDK_API int HASDK_CALL HA_GetLedLevel(
structHA_Cam* cam,
char *led_level
);
```

Description:
    Get led bright level

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
led_level	The bright level (1-100)	Out

HA_SetLedLevel

```
HASDK_API int HASDK_CALL HA_SetLedLevel(
structHA_Cam* cam,
charled_level
);
```

Description:

Set led bright level

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
led_level	The bright level (1-100)	In

HA_GetLedThreshold

```
HASDK_API int HASDK_CALL HA_GetLedThreshold(
structHA_Cam* cam,
char *led_threshold
);
```

Description:

Get led threshold value, used in led auto mode. The higher the value, the more possible the led goes on.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
led_threshold	The threshold value (0-255)	In

HA_SetLedThreshold

```
    HASDK_API int HASDK_CALL HA_SetLedThreshold(
    structHA_Cam* cam,
    charled_threshold
    );
```

Description:
    Set led threshold value, used in led auto mode. The higher the value, the more possible the led goes on.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
led_threshold	The threshold value (0-255)	In

4.13.4 Audio

Play Audio

HA_PlayAudio

```
    HASDK_API int HASDK_CALL HA_PlayAudio(
    structHA_Cam* cam,
    constchar* audio,
    int len
    );
```

Description:
    Play audio

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
audio	Audio buffer	In
len	Audio buffer length	In

Note:
   Only support wave format (parameter: A-Law, 8000Hz,64kbps,mono)
   Generate audio file on windows:

```
ffmpeg.exe -iinput.wav -acodecpcm_alaw -ac 1 -ar 8000 output.wav
```

Default Audio Configuration

HA_GetAudioList

```
HASDK_API int HASDK_CALL HA_GetAudioList(
HA_Cam *cam,
structAudioItem* items,
int itemBufNum,
int* itemNum
);
```

Description:
   Get built-in audio list

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
items	The audio array	Out
itemBufNum	Capacity of storage for audios	In
itemNum	Size of audio array	Out

HA_TestAudioItemByName

```
HASDK_API int HASDK_CALL HA_TestAudioItemByName(
HA_Cam *cam,
conststructAudioItem* items
);
```

Description:
   Test playing build-in audio by name

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
items	Audio to play	In

HA_SetAudioDefault

```
HASDK_API int HASDK_CALL HA_SetAudioDefault(
HA_Cam *cam,
int audioId
);
```

Description:
    Set audio which will be played when triggered by whitelist

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
audioId	Audio id	In

4.14 Camera Control

Note: need hardware support, works only on some models

HA_FocusAndZoomCtl

```
HASDK_API int HASDK_CALL HA_FocusAndZoomCtl(
structHA_Cam* cam,
int ptzCtl,
int ctlMode
);
```

Description:
    Focus and zoom control

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ptzCtl	The action, see PTZCTL	In
ctlMode	Action parameter, see PTZMODE	In

4.15 Customer Authentication Code

HA_WriteCustomerAuthCode

```
HASDK_API int HASDK_CALL HA_WriteCustomerAuthCode(
structHA_Cam* cam,
char* auth,
int size
);
```

Description:
    Set user-defined authentication code

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
auth	Authentication code	In
size	Size of authentication code	In

HA_ReadCustomerAuthCode

```
HASDK_API int HASDK_CALL HA_ReadCustomerAuthCode(
structHA_Cam* cam,
char* auth,
int* size
);
```

Description:
    Get user-defined authenticaton code

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
auth	Authentication code	Out
size	Size of authentication code	Out

4.16 Reboot Camera

HA_SystemReboo

```
HASDK_API int HASDK_CALL HA_SystemReboot(HA_Cam* cam);
```

Description:
    Reboot camera

HA_AppReboot

```
HASDK_API int HASDK_CALL HA_AppReboot(HA_Cam* cam);
```

Description:
    Restart camera app

4.17 Feature Authorization

HA_FunctionAuth

```
HASDK_API int HASDK_CALL HA_FunctionAuth(
    struct HA_Cam* cam,
    short number,
    const char* data,
    short size
    );
```

Description:
    Functionality authorization

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
number	The functionality to authorize 0: liveness detection 2: TTS	In
data	The authorization code	In
size	The size of the code	In

```
    HASDK_API int HASDK_CALL HA_AuthState(
        struct HA_Cam* cam,
        short number
        );
```

Description:
    Get authorization state

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
number	The functionality to query 0: liveness detection	In

Returns:
    0: unauthorized
    Non-zero: authorized

4.18 Platform Integration

HA_GetPlatformAccessParam

```
    HASDK_API int HASDK_CALL HA_GetPlatformAccessParam(
        struct HA_Cam* cam,
        struct PlatformAccess* param
        );
```

Description:
    Get platform integration parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
param	The parameter	Out

HA_SetPlatformAccessParam

```
HASDK_API int HASDK_CALL HA_SetPlatformAccessParam(
    struct HA_Cam* cam,
    struct PlatformAccess* param
    );
```

Description:
  Set platform integration parameter

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
param	The parameter	In

4.19 4G Module Status

HA_Get4GInfo

```
int HASDK_CALL HA_Get4GInfo(
    struct HA_Cam* cam,
    FourthGInfo* info
    );
```

Description:
  Query 4G status
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
info	4G info	Out

4.20 Perform ping from Camera

HA_CamPing

```
HASDK_API int HASDK_CALL HA_CamPing(
struct HA_Cam* cam,
char* url_ip,
int timeout
);
```

Description:

Perform ping from camera

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
url_ip	ip	In
timeout	Timeout	In

See also:

HA_RegCamPingCb

4.21 wifi

HA_SearchWifi

```
HASDK_API int HASDK_CALL HA_SearchWifi(
struct HA_Cam* cam,
const char* ssid,
);
```

Description:

Search wifi or get connected wifissid

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ssid	Value indicating what action to perform Null: search wifi Non-null: buffer to receive connected wifissid	In

Note:

For searching wifi, result is passed through callback function

See also:

HA_RegSearchWiFiCb

HA_ConnectWifi

```
HASDK_API int HASDK_CALL HA_ConnectWifi(
struct HA_Cam* cam,
const char*ssid,
const char* password,
char enable
);
```

Description:

Connect to or disconnect from wifi

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
ssid	ssid to connect to, null for disconnect	In
password	Password of the wifi, null for disconnect	In
enable	Action to perform 0:disconnect 1:connect	In

Note:

Return value 0 indicates the camera has accept the request, the connection result is passed through callback function

See also:

HA_RegWifiConnectCb

4.22 Send Json Command to Camera

HA_SendJson

```
HASDK_API int HASDK_CALL HA_SendJson(
    struct HA_Cam* cam,
    const char*cmd,
        const char* json,
        unsigned intjsonSize,
        char*replyJson,
        int buffSize
        );
```

Description:
    Send command in json format to camera.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cmd	Json command string	In
json	Json data	In
jsonSize	Length of json data	In
replyJson	The buffer to receive reply in json format	Out
buffSize	The length of receiving buffer	In

4.23 Sip

HA_CamSipCall

```
    HASDK_API int HASDK_CALL HA_CamSipCall(
    struct HA_Cam* cam,
    const char* sip_num
    );
```

Description:
    Make a sip call

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
sip_num	Sip address Following format is supported: Ip address: e.g.,sip:192.168.0.188	In

	house number: e.g., num:0001	

Note:
    Before making sip call, sip account must be registered

See also:
  HA_RegSIPKeyEventCb
  HA_RegSIPCallEventCb

HA_SetVoipRegister

```
HASDK_API int HASDK_CALL HA_SetVoipRegister(
struct HA_Cam* cam,
const char* username,
const char* domain,
unsigned short port,
const char* password
);
```

Description:
    Set sip account
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
username	User name of sip account	In
domain	Sip server domain or ip	In
port	Sip server port number	In
password	Password of sip account	In

HA_CamSipHangup

```
HASDK_API int HASDK_CALL HA_CamSipHangup(
struct HA_Cam* cam,
);
```

Description:
    Hang up an ongoing outgoing sip call

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In

Note:
    Hang up an ongoing sip call

HA_CamSipAnswer

```
HASDK_API int HASDK_CALL HA_CamSipAnswer (
struct HA_Cam* cam,
int call_answer
);
```

Description:
    Answer or reject an incoming sip call

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
call_answer	Answer or reject 0:reject 1:answer	In

See also:
  HA_RegSIPKeyEventCb
  HA_RegSIPCallEventCb

HA_CamSipAddRoomId

```
HASDK_API int HASDK_CALL HA_CamSipAddRoomId(
const struct SipRoomIdGroup*roomid_group,
unsigned int count
);
```

Description:
    Add the sip account corresponding to the room number
Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
roomid_group	The sip account info array to add	In
count	The size of the array Up to 100 at a time	In

Note:
    Sip account with the same house number will be overwiten

HA_CamSipQureRoomId

```
HASDK_API int HASDK_CALL HA_CamSipQureRoomId(
struct HA_Cam* cam,
const char* room_id,
char fuzzy,
struct SipRoomIdGroup* roomid_group,
int* group_count,
int page_no,
int page_size
);
```

Description:
    Query the sip account of specific room number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
room_id	The room number, null for all room number	In
fuzzy	Whether perform fuzzy match 0:Exact match 1:Fuzzy match	In
roomid_group	Array of room numbers	Out
group_count	In: size of room number array Out: number of query result array	In/Out
page_no	1-based page number (fuzzy match only)	In
page_size	Page size, max 500 (fuzzy match only)	In

HA_CamSipDelRoomId

```
HASDK_API int HASDK_CALL HA_CamSipDelRoomId(
struct HA_Cam* cam,
    const char* room_id[],
    unsigned int count
    );
```

Description:
    Delete sip account by room number

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
room_id	The room number array to delete corresponding sip account, null to clear all sip account	In
count	The size of the array	In

Callback Functions

HA_ClearAllCallbacks

```
        HASDK_API int HASDK_CALL
    HA_ClearAllCallbacks(structHA_Cam* cam);
```

Description:
    Clear all callbacks of specified camera

HA_ClearAllCallbacksEx

```
    HASDK_API int HASDK_CALL HA_ClearAllCallbacksEx();
```

Description:
    Clear all callbacks of all cameras

HA_RegAlarmRecordCb

```
    HASDK_API void HASDK_CALL HA_RegAlarmRecordCb(
    HA_Cam* cam,
    HA_AlarmRecordCb_t  cb,
    void* usrParam
    );
```

Description:
Register callback for gate open event, every time the gate is opened a new record will be generated

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	Pointer to the callback function	In
usrParam	User defined parameter	In

HA_RegAlarmRequestCb

```
HASDK_API void HASDK_CALL HA_RegAlarmRequestCb(
HA_Cam* cam,
HA_AlarmRequestCb_t cb,
void* usrParam
);
```

Description:
Register callback for gate open request event. Camera in online mode will not open gate automatically, it will send an gate open request instead.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegFaceReRegistProgressCb

```
HASDK_API void HASDK_CALL HA_RegFaceReRegistProgressCb(
HA_Cam* cam,
HA_FaceReRegistProgressCb_tcb,
void* usrParam
);
```

Description:
Re-register face progress callback. In the case of the algorithm upgrade, the face will be re-registered, during the process of

upgrading, the camera is not operational until the upgrading process is finished, the purpose of the callback is to notify the progress of re-register process.

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegDiscoverIpscanCb

```
HASDK_API void HASDK_CALL HA_RegDiscoverIpscanCb(
discover_ipscan_cb_tcb,
int usrParam
);
```

Description:
    Camera discovery result callback

Arguments:

Argument	Description	In/Out
cb	The callback function	In
usrParam	User defined parameter	In

See also:
    HA_DiscoverIpscan

HA_RegLiveStreamCb

```
HASDK_API void HASDK_CALL HA_RegLiveStreamCb(
HA_LiveStreamCb_tcb,
int usrParam
);
```

Description:
    Register streaming callback for all connected cameras, the data received is not decoded

Arguments:

Argument	Description	In/Out
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegLiveStreamCbEx

```
HASDK_API void HASDK_CALL HA_RegLiveStreamCbEx(
HA_Cam* cam,
HA_LiveStreamCb_tcb,
int usrParam
);
```

Description:

Register streaming callback for specified cameras, the data received is not decoded

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback	In
usrParam	User defined parameter	In

HA_RegFaceRectCb

```
HASDK_API void HASDK_CALL HA_RegFaceRectCb(
HA_Cam* cam,
HA_FaceRectCb_t cb,
void* usrParam
);
```

Description:

Register face rect detected event callback

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback	In
usrParam	User defined parameter	In

Note:
    Works only if debug mode is enabled

See also:
    HA_SetDebugEnable

HA_RegConnectEventCb

```
HASDK_API void HASDK_CALL HA_RegConnectEventCb(
HA_ConnectEventCb_tcb,
int usrParam
);
```

Description:
    Register global connect event callback

Arguments:

Argument	Description	In/Out
cb	The callback	In
usrParam	User defined parameter	In

Note:
    The callback works only if it’s enabled by calling HA_SetNotifyConnected(1)

HA_RegConnectEventCbEx

```
HASDK_API void HASDK_CALL HA_RegConnectEventCbEx(
HA_Cam* cam,
HA_ConnectEventCb_tcb,
int usrParam
);
```

Description:
    Register connects event callback for specified camera

Arguments:

Argument	Description	In/Out

cam	Handle to camera	In
cb	The callback	In
usrParam	User defined parameter	In

Note:
    The callback works only if it is enabled by calling HA_SetNotifyConnected(1)

HA_RegVerifyStatusCbEx

```
HASDK_API void HASDK_CALL HA_RegVerifyStatusCbEx(
HA_Cam* cam,
HA_VerifyStatusCb_tcb,
void* usrParam
);
```

Description:
    Login authentication status callback

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegFaceRecoCb

```
HASDK_API void HASDK_CALL HA_RegFaceRecoCb(
HA_Cam* cam,
HA_FaceRecoCb_t cb,
void* usrParam
);
```

Description:
    Register callback for face recognition event

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegFaceQueryCb

```
HASDK_API void HASDK_CALL HA_RegFaceQueryCb(
HA_Cam* cam,
HA_FaceQueryCb_tcb,
void* usrParam
);
```

Description:
    Register callback for face registration query result

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

See also:
    Query Face Registration

HA_RegFaceDeleteProgressCb

```
HASDK_API void HASDK_CALL HA_RegFaceDeleteProgressCb(
HA_Cam* cam,
HA_FaceDelProgressCb_tcb,
void* usrParam
);
```

Description:
    Register face registration deletion progress callback

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

See also:

Delete Face Registration

HA_RegFaceDebugImageCb

```
HASDK_API void HASDK_CALL HA_RegFaceDebugImageCb(
HA_Cam* cam,
HA_FaceDebugImageCb_tcb,
void* usrParam
);
```

Description:

Register face debug image callback

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegFaceDebugInfraredImageCb

```
HASDK_API void HASDK_CALL HA_RegFaceDebugInfraredImageCb(
HA_Cam* cam,
HA_FaceDebugInfraredImageCb_tcb,
void* usrParam
);
```

Description:

Register infrared debug image callback

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegFaceRecordQueryCb

```
    HASDK_API void HASDK_CALL HA_RegFaceRecordQueryCb(
    HA_Cam* cam,
    HA_FaceRecordCb_tcb,
    void* usrParam
    );
```

Description:
    Register callback for query face capture record

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

See also:
    HA_QueryFaceRecord

HA_RegSnapshotCb

```
    HASDK_API void HASDK_CALL HA_RegSnapshotCb(
    HA_Cam* cam,
    HA_SnapshotCb_tcb,
    void* usrParam
    );
```

Description:
    Register snapshot image callback

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

See also:
    HA_Snapshot

HA_RegReadTSerialCb

```
    HASDK_API int HASDK_CALL HA_RegReadTSerialCb(
    HA_ReadTSerialCb_tcb,
    int usrParam
    );
```
Description:

Register global callback for read data from transparent serial port data

Arguments:

Argument	Description	In/Out
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegReadTSerialCbEx

```
    HASDK_API int HASDK_CALL HA_RegReadTSerialCbEx(
    HA_Cam* cam,
    HA_ReadTSerialCb_tcb,
    int usrParam
    );
```
Description:

Register callback to specified camera from which to read data from transparent serial port data

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegGpioInputCb

```
    HASDK_API int HASDK_CALL HA_RegGpioInputCb(
    HA_Cam* cam,
    HA_GpioInputCb_tcb,
    void* usrParam
    );
```
Description:

Register callback for GPIO/Wiegand data input event

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegWGInputCb

```
HASDK_API int HASDK_CALL HA_RegWGInputCb(
HA_Cam* cam,
HA_WGInputCb_tcb,
void* usrParam
);
```

Description:
Register callback for Wiegand device data input event, only WG66 is supported

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegCamPingCb

```
HASDK_API void HASDK_CALL HA_RegCamPingCb(
   HA_Cam* cam,
   HA_CamPingCb_tcb,
   void* usrParam
   );
```

Description:
Register callback for ping result

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegWifiConnectCb

```
HASDK_API void HASDK_CALL HA_RegWifiConnectCb(
   HA_Cam*   cam,
   HA_WifiConnectCb_tcb,
   void* usrParam
   );
```

Description:
   Register callback for wifi connect status

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

See also:
   HA_ConnectWifi

HA_RegSearchWiFiCb

```
HASDK_API void HASDK_CALL HA_RegSearchWiFiCb(
   HA_Cam*   cam,
   HA_SearchWiFiCb_tcb,
   void* usrParam
   );
```

Description:
   Register callback for wifi searching result

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

See also:
    HA_SearchWifi

HA_RegQRCodeCb

```
    HASDK_API void HASDK_CALL HA_RegQRCodeCb(
        HA_Cam*   cam,
        HA_QRCodeCb_tcb,
        void* usrParam
        );
```

Description:
    Register callback for QR code scan result

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegSIPCallEventCb

```
    HASDK_API void HASDK_CALL HA_RegSIPCallEventCb(
        HA_Cam*   cam,
        HA_SIPCallEventCb_tcb,
        void* usrParam
        );
```

Description:
    Register callback for sip call event

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

HA_RegSIPKeyEventCb

```
    HASDK_API void HASDK_CALL HA_RegSIPKeyEventCb(
        HA_Cam*   cam,
        HA_SIPKeyEventCb_tcb,
        void* usrParam
        );
```

Description:
Register callback for sip key press event

Arguments:

Argument	Description	In/Out
cam	Handle to camera	In
cb	The callback function	In
usrParam	User defined parameter	In

Misc Functions

Logging

HA_LogEnable

```
    HASDK_API void HASDK_CALL HA_LogEnable(
        char enable
        );
```

Description:
Enable/disable logging

Arguments:

Argument	Description	In/Out
enable	Value indicating whether to enable logging 0: disable Non-zero: enable	In

Note:
By default, logging is enabled and rotated with a max size of 1 MB. The log file is D:/HA_log.txt.

Decode Jpeg Data

HA_DecodeJpeg

```
HASDK_API int HASDK_CALL HA_DecodeJpeg(
constunsignedchar* srcJpg,
int srcJpgLen,
unsignedchar* desRgb,
unsignedint* jpgW,
unsignedint* jpgH
);
```

Description:
Jpeg decoding function

Arguments:

Argument	Description	In/Out
srcJpg	Jpg data	In
srcJpgLen	Jpg data length	In
desRgb	Decoded data in BGR format	Out
jpgW	The with of image	Out
jpgH	The height of image	Out

HA_DecodeJpgSize

```
HASDK_API int HASDK_CALL HA_DecodeJpgSize(
constunsignedchar* srcJpg,
int srcJpgLen,
unsignedint* jpgW,
unsignedint* jpgH
);
```

Description:
Get jpeg image width and height

Arguments:

Argument	Description	In/Out
srcJpg	Jpg data	In
srcJpgLen	Jpg data length	In
jpgW	The width	Out
jpgH	The height	Out

HA_SaveBMP

```
    HASDK_API void HASDK_CALL HA_SaveBMP(
    constchar *filename,
    constunsignedchar* imgData,
    int width,
    int height,
    int cn
    );
```

Description:

Save image in bmp format

Arguments:

Argument	Description	In/Out
filename	The destination file name	In
imgData	Image data inBGR format	In
width	The width	In
height	The height	In
cn	Channel count	In

HA_SaveJpg

```
    HASDK_API int HASDK_CALL HA_SaveJpg(
    constchar *filename,
    constunsignedchar* jpgBuf,
    int len
    );
```

Description:

Save image in jpg format

Arguments:

Argument	Description	In/Out
filename	Destination file name	In
jpgBuf	Jpgdata	In
len	Jpg data length	In

Convert Between GB2312 and Utf8

HA_Gb23122Utf8

```
HASDK_API int HASDK_CALL HA_Gb23122Utf8(
char *inbuf,
int inlen,
char *outbuf,
int outlen
);
```

Description:

Convert from GB2312 to UTF8

Arguments:

Argument	Description	In/Out
inbuf	GB2312-encoded string buffer	In
inlen	GB2312-encoded string length	In
outbuf	UTF8	Out
outlen	UTF8 buffer length	In

HA_Utf82Gb2312

```
HASDK_API int HASDK_CALL HA_Utf82Gb2312(
char *inbuf,
int inlen,
char *outbuf,
int outlen
);
```

Description:

Convert from UTF8to GB2312

Arguments:

Argument	Description	In/Out
inbuf	UTF8-encoded string buffer	In
inlen	UTF8-encoded string length	In
outbuf	Gb2312 buffer	Out
outlen	gb2312 buffer length	In

Extract Face Feature and/or Normalized Image

HA_GetJpgFeatureImageNew

```
HASDK_API int HASDK_CALL HA_GetJpgFeatureImageNew(
const unsigned char *jpg,
int len,
unsigned char *feature_image,
int *feature_size,
unsigned char *faceImgJpg,
int *faceJpgLen,
const char* twist_version
);
```

Description:

For EV200, to be compatible with new algorithm

Arguments:

Argument	Description	In/Out
jpg	The jpg data	In
len	Length of jpg data	In
feature_image	The buffer to receive normalized image (must be > 110KB)	Out
feature_size	In: length of feature_image buffer Out: the actual length of normalized image	In/Out
faceImgJpg	The extracted facial image in jpg format (must be > 10KB)	Out
faceJpgLen	In: length of faceImgJpg buffer Out: the actual lengthofthe facial image	In/Out
twist_version	Algorithm version. if null, all versions of normalized image will be extracted	In

Note:

The algorithm version can be get through

HA_GetFaceSystemVersionExfunction. Call HA_FeatureConvert to convert to normalized image that can be used to register

HA_FeatureConvert

```
HASDK_API int HASDK_CALL HA_FeatureConvert(
HA_Cam* cam,
const unsigned char *feature_image,
int feature_size,
FaceImage* twist_image,
const char* twist_version
);
```

Description:

Convert the normalized data extracted to registerable format

Arguments:

Argument	Description	In/Out
cam	Handle to camera. If null, the target algorithm version is specified by twist_version	In
feature_image	The new normalized data	In
feature_size	The new normalized data length	In

twist_image	Normalized data
	struct FaceImage
	{
	int img_seq; //fixed to 0
	int img_fmt; //fixed to 1
	unsigned char *img; //normalized image buffer(must be > 70KB)
	int img_len; //in:img buffer length out: actual length of normalized data
	int width; //width of image (Out)
	int height;//height of image(Out)
	};
		In/Out

twist_version	The algorithm version. If null, the target algorithm will be detected from camera handle	Out

Extract by BRG Data

HA_GetFeatureImage

```
    HASDK_API int HASDK_CALL HA_GetFeatureImage(
    constunsignedchar *bgrimage,
    int width,
    int height,
    unsignedchar *twist_image,
    int *twist_size,
    int *twist_width,
    int *twist_height,
    unsignedchar *faceImgJpg,
    int *faceJpgLen
    );
```

Description:

Extract normalized image and facial image.

Arguments:

Argument	Description	In/Out
bgrimage	The image, must be BGR format	In
width	The width of the image	In
height	The height of the image	In
twist_image	Normalized image in BGR format (must be > 70KB)	Out
twist_size	Normalized image length	Out
twist_width	Width of normalized image	Out
twist_height	Height of normalized image	Out
faceImgJpg	Facial image in jpg format	Out
faceJpgLen	Length of facial image	Out

Extract by Jpeg Data

HA_GetJpgFeatureImage

```
    HASDK_API int HASDK_CALL HA_GetJpgFeatureImage(
    constunsignedchar *jpg,
    int len,
    unsignedchar *twist_image,
    int *twist_size,
    int *twist_width,
    int *twist_height,
    unsignedchar *faceImgJpg,
    int *faceJpgLen
    );
```

Description:
    Extract normalized image and facial image

Arguments:

Argument	Description	In/Out
jpg	The image, support jpg, png, bmp	In
len	The length of the image	In
twist_image	Normalized image in BGR format (must be > 70KB)	Out
twist_size	Normalized image length	Out
twist_width	Width of normalized image	Out
twist_height	Height of normalized image	Out
faceImgJpg	Facial image in jpg format	Out
faceJpgLen	Length of facial image	Out

Extract by Jpeg Image Path

HA_GetJpgPathFeatureImage

```
    HASDK_API int HASDK_CALL HA_GetJpgPathFeatureImage(
    constunsignedchar *img_path,
    unsignedchar *twist_image,
    int *twist_size,
    int *twist_width,
    int *twist_height,
    unsignedchar *faceImgJpg,
    int *faceJpgLen
    );
```

Description:
    Extract normalized image and facial image.

Arguments:

Argument	Description	In/Out
img_path	The path of the image, support jpg, png, bmp	In
twist_image	Normalized image in BGR format (must be > 70KB)	Out
twist_size	Normalized image length	Out
twist_width	Width of normalized image	Out
twist_height	Height of normalized image	Out
faceImgJpg	Facial image in jpg format	Out
faceJpgLen	Length of facial image	Out

Test If Image Is Qualified for Registration

HA_FaceDetect

```
HASDK_API int HASDK_CALL HA_FaceDetect(
constunsignedchar *rgbimage,
int width,
int height,
structha_rect *rect,
structHA_Point*oripoint
);
```

Description:
    Test if image is qualified for registration

Arguments:

Argument	Description	In/Out
rgbimage	The image data in BGR format	In
width	The width of the image	In
height	The height of the image	In
rect	The face bound in the image	Out
oripoint	The five feature point detected	Out

HA_FaceJpgLegal

```
    HASDK_API int HASDK_CALL HA_FaceJpgLegal(
    constunsignedchar *jpg,
    int len
    );
```

Description:

    Test if jpeg image is qualified for registration

Arguments:

Argument	Description	In/Out
jpg	Jpg image data	In
len	Jpg image data length	In

Returns:

    0: passes the test
    Non-zero: error code

Callback Functions

discover_ipscan_cb_t

Description:

    Camera searching callback function

```
    typedef void (HASDK_CALL *discover_ipscan_cb_t)(
    conststructipscan_t* ipscan,
    int usr_param
    );
```

Arguments:

Argument	Description
ipscan	The discovered camera ip
usr_param	User defined parameter

See also:

HA_RegDiscoverIpscanCb
HA_DiscoverIpscan

HA_ConnectEventCb_t

Description:

Connect event callback function

```
typedef void (HASDK_CALL *HA_ConnectEventCb_t)(
structHA_Cam* cam,
constchar* ip,
unsignedshort port,
int event,
int usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
ip	IP address
port	Port number
event	event 1：connect 2：disconnect
usrParam	User defined parameter

See also:

HA_RegConnectEventCbEx
HA_SetNotifyConnected

HA_ServerConnectCb_t

Description:

Sever connect event callback

```
typedef void (HASDK_CALL *HA_ServerConnectCb_t)(
structHA_Cam* cam,
DeviceInfor devInfor,
unsignedshort port,
int event,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera, null if fail to connect
ip	Ip address

port	Port number
event	event -1: device No. duplicated -2: device No. not configured 1: connect 2: disconnect
usrParam	User defined parameter

HA_LiveStreamCb_t

Description:

Live stream callback (H264)

```
typedef void (HASDK_CALL *HA_LiveStreamCb_t)(
structHA_Cam* cam,
constchar* ip,
conststructHA_LiveStream* stream,
int usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
ip	Ip address
stream	The stream data
usrParam	User defined parameter

See also:

HA_RegLiveStreamCbEx

HA_HA_DecodeImageExCb_t

Description:

Live stream data callback (rgb)

```
typedef void (HASDK_CALL *HA_DecodeImageCbEx_t)(
structHA_Cam* cam,
HA_LiveStream* stream,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera

stream	Live stream data
width	Width of the image
usrParam	User defined parameter

See also:
HA_StartStreamEx1

HA_ReadTSerialCb_t

Description:
Transparent serial port data read callback

```
typedef void (HASDK_CALL *HA_ReadTSerialCb_t)(
structHA_Cam* cam,
int index,
constunsignedchar* data,
int size,
int usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
index	Serial port type 1:485 2:232
data	Data read
size	Data length
usrParam	User defined parameter

See also:
HA_RegReadTSerialCbEx

HA_GpioInputCb_t

Description:
GPIO data input callback

```
typedef void (HASDK_CALL *HA_GpioInputCb_t)(
structHA_Cam* cam,
int type,
unsignedintdata,
    void* usrParam);
```

Arguments:

Argument	Description
cam	Handle to camera
type	GPIO signal type 255:gpio 0:26 bit Wiegand 1:34 bit Wiegand
data	When type is gpio, the gpio port number; When type is Wiegand, the Wiegand card number
usrParam	User defined parameter

- **HA_WGInputCb_t**

Description:
GPIO input callback

```
typedef void (HASDK_CALL *HA_WGInputCb_t)(
structHA_Cam* cam,
int type,
unsignedlong long data,
void* usrParam);
```

Arguments:

Argument	Description
cam	Handle to camera
type	Input signal type 4:WG66
data	The Widgand card number
usrParam	User defined parameter

HA_FaceRecoCb_t

Description:
    Face recognition event callback

```
typedef void (HASDK_CALL* HA_FaceRecoCb_t)(
structHA_Cam* cam,
conststructFaceRecoInfo* faceRecoInfo,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
faceRecoInfo	The face recognition info
usrParam	User defined parameter

See also:
    HA_RegFaceRecoCb

HA_FaceQueryCb_t

Description:
    Face query result callback

```
typedef void (HASDK_CALL* HA_FaceQueryCb_t)(
structHA_Cam* cam,
conststructQueryFaceInfo* faceQueryInfo,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
faceQueryInfo	The face registration info
usrParam	User defined parameter

See also:
    HA_RegFaceQueryCb

HA_FaceRectCb_t

Description:
    The face bound callback

```
typedef void (HASDK_CALL*HA_FaceRectCb_t)(
structHA_Cam* cam,
conststructFaceRect* faceRects,
int rectNum,
int showIdFlag,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
faceRects	The face bound array
rectNum	The size of face bound array
showIdFlag	Value indicating whether show face bound
usrParam	User defined parameter

See also:
HA_RegFaceRectCb

HA_FaceDebugImageCb_t

Description:
Face debug image callback

```
typedef void (HASDK_CALL* HA_FaceDebugImageCb_t)(
structHA_Cam* cam,
structDebugImageInfo *debugImageInfo,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
debugImageInfo	Debug image info
usrParam	User defined parameter

See also:
HA_RegFaceDebugImageCb

HA_FaceDebugInfraredImageCb_t

Description:
Infrared image debug callback

```
typedef void (HASDK_CALL* HA_FaceDebugInfraredImageCb_t)(
structHA_Cam* cam,
structBebugInfraredImage *debugImageInfo,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
debugImageInfo	Infrared debug image data
usrParam	User defined parameter

See also:

HA_RegFaceDebugInfraredImageCb

HA_FaceDelProgressCb_t

Description:

Face delete progress callback

```
typedef void (HASDK_CALL* HA_FaceDelProgressCb_t)(
structHA_Cam* cam,
structFaceDelProgressInfo *delProgress,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
delProgress	The progress info
usrParam	User defined parameter

See also:

HA_RegFaceDeleteProgressCb

Delete Face

HA_SnapshotCb_t

Description:

Snapshot callback

```
typedef void (HASDK_CALL* HA_SnapshotCb_t)(
structHA_Cam* cam,
structSnapshotImage *snapImage,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
snapImage	The snapshot image
usrParam	User defined parameter

See also:

HA_RegSnapshotCb

HA_Snapshot

HA_VerifyStatusCb_t

Description:

Verify status callback

```
typedef void (HASDK_CALL *HA_VerifyStatusCb_t)(
structHA_Cam* cam,
constchar* ip,
unsignedshort port,
int status,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
ip	Ip address
port	Port number
status	Verification status 0: success Non-zero: see error code
usrParam	User defined parameter

See also:

HA_RegVerifyStatusCbEx

HA_AlarmRecordCb_t

Description:

Gate open event callback

```
typedef void (HASDK_CALL *HA_AlarmRecordCb_t)(
structHA_Cam* cam,
structAlarmInfoRecord*alarmRecord,
void* usrParam)
;
```

Arguments:

Argument	Description
cam	Handle to camera
alarmRecord	Gate opening info
usrParam	User defined parameter

Note:

Each time a white-listed person matched triggers the callback

See also:

HA_RegAlarmRecordCb

HA_AlarmRequestCb_t

Description:

Open gate request callback

```
typedef void (HASDK_CALL *HA_AlarmRequestCb_t)(
structHA_Cam* cam,
structAlarmRequest*alarmRequest,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
alarmRequest	The request info
usrParam	User defined parameter

See also:

HA_RegAlarmRequestCb

HA_FaceReRegistProgressCb_t

Description:

Reregister face progress callback

```
typedef void (HASDK_CALL* HA_FaceReRegistProgressCb_t)(
structHA_Cam* cam,
structFaceReRegistProgressInfo *registProgress,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
registProgress	The reregister progress
usrParam	User defined parameter

See also:

HA_RegFaceReRegistProgressCb

HA_FaceRecordCb_t

Description:

Face record query callback

```
typedef void (HASDK_CALL* HA_FaceRecordCb_t)(
structHA_Cam* cam,
conststructRecordData* recordInfo,
void* usrParam
);
```

Arguments:

Argument	Description
cam	Handle to camera
recordInfo	The record data
usrParam	User defined parameter

See also:
  HA_RegFaceRecordQueryCb
  HA_QueryFaceRecord

HA_SearchWiFiCb_t

Description:
  The wifi searching result callback

```
typedef void (HASDK_CALL *HA_SearchWiFiCb_t)(
   struct HA_Cam* cam,
   int num ,
   WifiSignal* signal,
   void*usrParam
   );
```

Arguments:

Argument	Description
cam	Handle to camera
num	Number of wifi info array
signal	wifi info array
usrParam	User defined parameter

See also:
  HA_RegSearchWiFiCb

HA_CamPingCb_t

Description:
  Camera ping result callback

```
typedef void (HASDK_CALL *HA_CamPingCb_t)(
   struct HA_Cam* cam,
   char* Infor,
   int size,
   void*usrParam
   );
```

Arguments:

Argument	Description
cam	Handle to camera

Infor	Ping result
size	The lengh of ping result
usrParam	User defined parameter

See also:
    HA_RegCamPingCb
    HA_CamPing

HA_WifiConnectCb_t

Description:
    Wifi connect result callback

```
typedef void (HASDK_CALL *HA_WifiConnectCb_t)(
    struct HA_Cam* cam,
    int state,
    char* ssid,
    char* ip,
    void* usrParam
    );
```

Arguments:

Argument	Description
cam	Handle to camera
state	0: connect succeed 1: disconnected 2:wifi not found 3: password error 4: connection rejected
ssid	The wifissid when connect succeed, otherwise, null
ip	Wifi interface ip address when connect succeed, othersize, null
usrParam	User defined parameter

See also:
    HA_RegWifiConnectCb

HA_QRCodeCb_t

Description:
    QR code scan result

```
typedef void (HASDK_CALL *HA_QRCodeCb_t)(
    struct HA_Cam* cam,
    unsigned char* code,
    void* resv,
    void* usrParam
    );
```

Arguments:

Argument	Description
cam	Handle to camera
code	The qr code string
resv	reserved
usrParam	User defined parameter

See also:

HA_RegQRCodeCb

HA_SIPCallEventCb_t

Description:

SIP call event callback

```
typedef void (HASDK_CALL *HA_SIPCallEventCb_t)(
    struct HA_Cam* cam,
    int sipEvent,
    char* resv,
    void* userParam
    );
```

Arguments:

Argument	Description
cam	Handle to camera
sipEvent	Sip call event, see enumSIPEventCall
resv	reserved
usrParam	User defined parameter

See also:

HA_RegSIPCallEventCb

HA_SIPKeyEventCb_t

Description:
    SIP key press event callback

```
    typedef void (HASDK_CALL *HA_SIPKeyEventCb_t)(
        struct HA_Cam* cam,
        int keyId,
        char* resv,
        void* userParam
        );
```

Arguments:

Argument	Description
cam	Handle to camera
keyId	The key code keyId 0-9 == 0-9 keyId 10 == * keyId 11 == #
resv	reserved
usrParam	User defined parameter

See also:
    HA_RegSIPKeyEventCb

5 Sample Code

5.1 Connect Camera

```
    void __stdcallConnectEventCb(struct HA_Cam* cam, const char* ip,
unsigned short port, int    event, int usrParam)
    {
        If(event==1)
            printf("connect succeed");
        else if(evet==2)
            printf("disconnected");
    }//define the connect event callback function


    HA_Init();                //SDK initialization
    HA_InitFaceModel(NULL); //face extraction module initialization
    HA_SetNotifyConnected(1);  //enable connect event callback
    HA_RegConnectEventCb(ConnectEventCb,0);  //register global
connect event
    char* p_ip="192.168.0.111"; //ip address of camera
    int port =9527;  //port, fixed to 9527;
    int erroNum=0;  /*the error code, for some old camera, the
connection can be successful even if the connect function return
error, thus, the HA_Connected function is recommended to test if
connect is successful*/
    HA_Cam* cam = HA_Connect(p_ip, port, NULL, NULL, &erroNum);
//connect to the camera
    If(HA_Connected(cam))  //test connect status
        printf("succeed");
    else
        printf("failed");
    /**************************************
    Call other functions here
    **************************************/
        HA_DeInit();  // SDK deinitialization
```

5.2 Register Face

```
    FaceFlags faceID; //the person info struct
    char* patch[5];
    patch[0]="./xxx.jpg"; //the face image path
    faceID.effectTime = 0xFFFFFFF;  //valid to time: never expire
in this case
    faceID.effectStartTime = 0;     //valid from time
    faceID.role = 1;    //person category, normal
    strcpy(faceID.faceID, "xxxxxxxxx"); //face id, must be unique
    strcpy(faceID.faceName, "xxxxxxxxx");//person name
  int ret=HA_AddJpgPaths(cam, &faceID, patch, 1, 1); //do the
     registration
    if(ret==ERR_NONE)
        printf("register succeed");
    else
        printf("register failed, error code=%d\n",ret);//failed
    register, show error code
```

5.2 Registration Query

```
    void __stdcallfaceQueryCb_t(struct HA_Cam* cam, const struct
QueryFaceInfo* faceQueryInfo, void* usrParam){
    printf("record_count=%d record_no=%d personID=%s version=%d \n", faceQueryInfo->record_count,
faceQueryInfo->record_no, faceQueryInfo->personID, faceQueryInfo->version);//the callback
function, print info here
  } /*define the callback for the query result, record_no == 0
     indicates no more query result*/

    HA_RegFaceQueryCb(cam, faceQueryCb_t,NULL); //register face
query callback function
    char flags=0;
    short mode=1;//fuzzy query
    QueryCondition conditon;//the criteria
    strcpy(conditon.faceID, "XXXXX"); //face id
    strcpy(conditon.faceName, "XXXXX"); //person name
    flags |= QUERY_BY_ID; //enable id field
    flags |= QUERY_BY_NAME;//enable name field
    int ret=HA_QueryFaceEx(g_cam[Using_cam].cam, -1, 1, 100, 1, 1, flags, mode,
&conditon);//do the query in pagination
    if (ret == ERR_NONE)
        printf("query succeed");
    else
        printf("query failed, error code =%d\n", ret);
```