AIIPC SDK

API 1.0.x

Version History

verson	date	author	Change log
1.0.0	2020.12.30	Duke zuo	First version

1：SDK C/C++ API

All interface header file(.h) are in directory inc, It include cliSdk.h、Fp16Convert.h 2 files.

1.1 get_sdk_version()

function	Param	Param description
get_sdk_version()	**char* version**	Version Information **(len>=50)**

Returns: void

example: Omitted

Description:
Gets SDK version information(include compiled date).

1.2 ScanIPCAddr()

function	Param	Param description
GetMtPoints()	**int nsecond**	**Max scan times**
	vector<string>& rtsp	**Vectory of rtsp url**

Returns: ipc number
example: Omitted

Description:
Auto scan ipc rtsp url in the same LAN with client.

1.3 ConnectToDevice()

function	Param	Param description
ConnectToDevice()	**const char* device_ip**	**Device ip**
	const char* filepath="./"	**If need record,record path**

Returns:>0 if successful, return handle of client endpoint session which can been used as others API input param, 0 otherwise.

example:

```
CLI_HANDLE hDevice = ConnectToDevice( “192.168.0.8” );
If (hCli==0)
{
        printf( “connect to device error! \n” );
}
```

```
else
{
        printf( “connect to device success! \n” );
}
```

Description:
Connect to our device,you can get device ip by Onvif tools or ret from ScanIPCAddr()

1.4 CloseDevice()

function	Param	Param description
CloseDevice()	CLI_HANDLE handle	Handle of device

Returns:0 if successful, -1 otherwise

example: Omitted

Description:
Close the conected session with device and free resources about this session

1.5 GetDeviceVer()

function	Param	Param description
GetDeviceVer()	CLI_HANDLE handle	Handle about device
	char* version	Version Information **(len>50)**

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
Get device firware version.

1.6 GetAIMode()

function	Param	Param description
GetAIMode()	CLI_HANDLE handle	Handle of device
	char* modename	Current ai mode name(len >50)

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
Get the currnet ai mode which used in device

1.7 DelAIMode()

function	Param	Param description
GetAlermTemp()	CLI_HANDLE handle	Handle of device

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
Delete ai mode in device

1.8 RebootByRemote()

function	Param	Param description
RebootByRemote()	CLI_HANDLE handle	Handle of device
	int action	Device action

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
Reboot device by client, action:1 device restart 2: restart rtsp server. if client can't play rtsp stream media over times if met problem.

1.9 StartUpgrade()

function	Param	Param description
StartUpgrade()	CLI_HANDLE handle	Handle of device
	const char* up_file	Updated file

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
You can build a update package refed readme.md in release diretory.

1.10 UpdateAiModel()

function	Param	Param description
UpdateAiModel()	CLI_HANDLE handle	Handle of device
	const char* ai_blob	Blob file of ai mode
	const char* ai_xml	Xml file of ai mode

	const char* mode_name	**Mode name**

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
Changed the ai mode in device.

1.11 QueryUpStep()

function	**Param**	**Param description**
DelMtArea()	CLI_HANDLE handle	**Handle of device**
	int step	**[0-100]**

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:if you used StartUpgrade() or UpdateAiModel(),it will cost some time to transmite file,so you can used QueryUpStep() to query transmission of progress.
Notes:because StartUpgrade() will block,you'd better used it in a alone thread.

1.12 ReadMetaData()

function	**Param**	**Param description**
ReadMetaData()	CLI_HANDLE handle	**Handle of device**
	BYTE* pbuf	**Point to data buffer**
	int& size	**Buf size in/out**

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
Read AI meta data from device.if size length is less meta data legth,will copy part data(the size input param),so you can used 1M buffer to read ai meta data,it is long enough.

1.13 SetEncParam()

function	**Param**	**Param description**
SetEncParam()	CLI_HANDLE handle	**Handle of device**
	struct video_enc_param* enc	**Enc param**

Returns:=0 if successful, -1 otherwise

example: Omitted

```
struct video_enc_param
{
  int enc_type;//264=h264;265=h265
  int enc_bps;   //unit KB
  int res[6];   //resverd
};
```

Description:
    Set video encode param

1.14 GetEncParam()

function	Param	Param description
GetEncParam()	CLI_HANDLE handle	Handle of device
	struct video_enc_param* enc	Enc param

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
    Get video encode param

1.15 SendCustomData()

function	Param	Param description
SendCustomData()	CLI_HANDLE handle	Handle of device
	char* in_outbuf	Inpu/output buffer
	int& in_outlen	Inpu/output length

Returns:=0 if successful, -1 otherwise

example: Omitted

Description:
    Send use customed data

2: video stream

Our device support H264/H265 NAL transmit method and support onvif too.

2.1 Onvif

Our device support standard onvif workflow which it work as a onvif server.

If the onvif client need find device (chat with the server) ,It work just like:

Discover Device>> Get Device Capabilities >> Get Device media info >> Get device video encoder info (media profile)>> Get media stream url >> ONVIF finish

If all work flow ok, The client will get media stream urls like:
rtsp://192.168.0.11:8554/liveRGB For RGB video

Default device IP is not fixed which used dhcp and get itself IP from router.if you want used fixed ip,you can used set ip address of onvif protocol by onvif tools.

2.2:RTSP

The device support stand rtsp work flow which it worked as a media server.
(C abbreviated client, S abbreviated device media server.)

Step 1: Query server methods

1. C->S:OPTION request //C: query all methods whicd media server supported.

2. S->C:OPTION response //S: response methods which ared included in public field .

Step 2: Get SDP information

1. C->S:DESCRIBE request //C ask server supported media description

2. 2. S->C:DESCRIBE response //S response sdp information

Step 3: Set up transmit session

1. C->S:SETUP request //C asked rtp channel transmit(just like tcp or udp) method which used Transport field to ask server to set up media transmit session.

2. 2. S->C:SETUP response //S if ok，return Session ID;

Step 4: Ask server start transmit rtp data

1 .C->S:PLAY request //C ask media data

2 .S->C:PLAY response //S return play start ok.

Step 5: RTP media data tranmit

S->C: send media data used RTP protocal C need receive and decode and render .

Step 6: Close session,exit

C->S:TEARDOWN request //C ask close these session

S->C:TEARDOWN response //S responsed and close media channel.