Cloud Client Software Development Kit User Guide_V1.0.0

Preface

The Cloud Client Software Development Kit (abbreviated as ZWP2P Client) is based on Peer-to-Peer (P2P) communication technology and is used to access and control any device that supports cloud protocols. Typical applications include:

- Accessing and controlling cameras (IPC),
- Digital Video Recorders (DVR),
- Network Video Recorders (NVR),
- Hybrid Video Recorders (HVR), and other video surveillance devices.

This guide aims to help users quickly utilize the SDK to access and control devices. It describes the background knowledge, principles, potential difficulties, and important considerations you need to know while using the SDK. It is essential reading for developers.
We have tried to minimize the documentation, aiming to ensure that developers only need to refer to the comments in the header files to assist with their development work. However, for more complex topics, additional explanations are provided in this document.

Target Audience

- Software developers
- Software testers

Reader Requirements

Since this development guide and examples are provided in C language, familiarity with basic C language knowledge is required.

Tips

Readers can display the "Document Structure" to see the file structure on the left side of the software, which presents the main contents clearly and intuitively, making it easier to find the target content quickly.

Platform Support

Platform	Description
Win32	Windows X86 32-bit platform

Platform	Description
Win64	Windows X86 64-bit platform (not supported)
Linux 32bit	Linux X86 32-bit platform
Linux 64bit	Linux X86 64-bit platform
arm-hisiv100nptl-linux	Hisilicon V100 Toolchain
arm-hisiv300-linux	Hisilicon V300 Toolchain
arm-hisiv500-linux	Hisilicon V500 Toolchain

Version History

Modification Date	Version	Changes
2017/08/26	1.0.0	First creation

Main Features Overview

The SDK currently supports the following features:

1. Login, Logout
2. Real-Time Stream Data Transmission and Control
3. Historical Stream Data Query, Transmission, and Control
4. Two-Way Audio (not supported yet)
5. PTZ Control
6. Remote Desktop
7. Configuration Get and Set
8. HTTP API Command Forwarding
9. Runtime Parameter Adjustment
10. Others

API Documentation

Initialization, De-initialization, and Runtime Parameters API

Runtime Parameters

- **Interface Name**:

```
    int CALLSTACK ZWP2PVNClient_SetAttribute(IN eZWP2P_SDK_ATTRIBUTE_TYPE eType, IN ZWP2P_UINT32 dwAttribute);
```

- **Description**:
  You can adjust runtime parameters based on your application scenario for a better user experience.
- **Parameters**:
  - `eType` - The attribute type, such as `eZWP2P_ATTRIBUTE_USER_CUSTOM_STREAM_HEADER_LEN` (for custom media data formats), `eZWP2P_ATTRIBUTE_DEVICE_STATUS_NOTIFY_TIME` (device status update interval, default 10 seconds), etc.
  - `dwAttribute` - The value for the selected attribute.
- **Return Values**:
  - `eZWP2P_ERR_FAILED_INVALID_PARAMETER` - Invalid parameters.
  - `eZWP2P_ERR_SUCCESS` - Success.

Initialization

- **Interface Name**:

```
int CALLSTACK ZWP2PVNClient_Start(LPINIT_NETWORK_PARAM cInitNetworkPara);
```

- **Description**:
  Initializes the network parameters.
- **Parameters**:
  - `cInitNetworkPara` - Network initialization parameters. For example, `sRemoteServerAddress` for the P2P server address. If the user has their own P2P server, enter the server address. Otherwise, leave it empty.
- **Return Values**:
  - `eZWP2P_ERR_FAILED` - Failure. Check the debug message for more information.
  - `eZWP2P_ERR_SUCCESS` - Success.

De-initialization

- **Interface Name**:

```
int CALLSTACK ZWP2PVNClient_Stop();
```

- **Description**:
  Stops the service and de-initializes.
- **Return Values**:
  - `eZWP2P_ERR_FAILED` - Failure. Check the debug message.
  - `eZWP2P_ERR_SUCCESS` - Success.

Login, Logout, Device Status API

Login

- **Interface Name**:

```
int CALLSTACK ZWP2PVNClient_UserLogin(OUT ZWP2P_HUSER *hUser, IN const LPZWP2P_USER_LOGIN_PARAM cUserLoginPara);
```

- **Description**:
  Logs in to the device.
- **Parameters**:
  - `hUser` - Output parameter. After successful login, a handle is returned for subsequent device operations.
  - `cUserLoginPara` - User login parameters such as `sDeviceUID` (device P2P ID), `sUName` (username), `sUPass` (password), and `pServerString` (P2P server address).
- **Return Values**:
  - `eZWP2P_ERR_FAILED` - Failure.
  - `eZWP2P_ERR_SUCCESS` - Success.

Logout

- **Interface Name**:

```
int CALLSTACK ZWP2PVNClient_UserLogout(IN ZWP2P_HUSER hUser);
```

- **Description**:
  Logs out from the device. This reduces resource usage.
- **Parameters**:
  - `hUser` - User handle.
- **Return Values**:
  - `eZWP2P_ERR_FAILED` - Failure.
  - `eZWP2P_ERR_SUCCESS` - Success.

Device Status

- **Device Status Callback Function Definition**

```
typedef int(CALLBACK *ZWP2P_CB_DEVICE_STATUS)(IN ZWP2P_HUSER hUser, IN ZWP2P_UINT32 dwDeviceStatus, IN ZWP2P_UINT32 dwUserData);
```

- **Description**:
  Callback function for device status updates.
- **Parameters**:

- `hUser` - User handle.
- `dwDeviceStatus` - Device status (0: Offline, 1: Online).
- `dwUserData` - User data.

- **Return Values**:
  - None.

Setting Device Status Callback Function

Interface Name

```
int CALLSTACK ZWP2PVNClient_SetDeviceStatusCB(IN ZWP2P_HUSER hUser, IN ZWP2P_CB_DEVICE_STATUS cbDeviceStatus, IN ZWP2P_UINT32 dwUserData);
```

Description
Set the device status callback function.

Parameter List

- `hUser` - User handle
- `cbDeviceStatus` - Status callback function, see the callback function definition for details.
- `dwUserData` - User data

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE` - Invalid handle, indicating an illegal handle value was used.

Check if the Device is Online

Interface Name

```
int CALLSTACK ZWP2PVNClient_IsDeviceOnline(IN ZWP2P_CHAR * pStrUID);
```

Description
Check whether the device is online.

Real-Time Stream Data Transmission Interface

The main work of the library in terms of real-time streaming is to efficiently and quickly retrieve the device's real-time media data and forward it to the application layer. The application layer will receive basic information about the media data frames. The media data is in raw standard format, and the application layer should cache, decode, play, store, or process the data after receiving it.

Note: Only one stream can be transmitted at a time on a single channel for a client. Multiple streams, such as main stream, sub-stream, or others, cannot be transmitted simultaneously. This limitation is mainly due to bandwidth considerations in internet transmission.

Connection

Interface Name

```
int CALLSTACK ZWP2PVNClient_RealStreamConnect(OUT ZWP2P_HSTREAM *hStream, IN ZWP2P_HUSER hUser, IN const LPZWP2P_REALSTREAM_PARA cStreamPara);
```

Description

Connect to the real-time stream.
Once connected, you need to call the function to set the real-time stream callback function. After the stream starts, media data will be sent to the application layer.

Note: The media data's resolution, bitrate, and frame rate may not fully match the video quality level. If required, call the function to adjust the encoding parameters for video quality.

Parameter List

- `hUser` - User handle (input)
- `hStream` - Real-time stream handle (output), will be filled by the library upon successful connection. Note: The handle starts from 0, and the application layer must avoid treating 0 as an invalid or initial value.
- `cStreamPara`
  - `dwChannel` - Channel number, starting from 0.
  - `eMediaQuality` - Video quality level, defined in `eZWP2P_QUALITY_LEVEL` enum.

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE` - Invalid handle, indicating an illegal handle value.

Real-Time Stream Callback Function Definition

Interface Name

```
typedef int (CALLBACK *ZWP2P_CB_STREAMMEDIA)(IN ZWP2P_HSTREAM hStream, ZWP2P_STREAMMEDIA_FRAME_HEADER * pStreamMedia, IN ZWP2P_UINT32 dwUserData);
```

Description

Real-time stream media data callback function definition.

Parameter List

- `hStream` - Real-time stream handle (input)
- `pStreamMedia` - Media data structure pointer (input), including basic media frame information and raw standard media data.
  Once the device sends media data, it will be passed back to the application layer through this function. The application layer should cache the data and immediately return, performing tasks like decoding, playback, storage, or processing in another thread.
  Do not perform time-consuming operations inside the callback function, as this may disrupt data transmission, leading to issues like frame loss, stuttering, or glitches.
- `dwUserData` - User data, passed back to the application layer when the callback is invoked.

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE` - Invalid handle, indicating an illegal handle value.

Set Real-Time Stream Media Data Callback Function

Interface Name

```
int CALLSTACK ZWP2PVNClient_StreamMediaCB(IN ZWP2P_HSTREAM hStream, IN ZWP2P_CB_STREAMMEDIA cbStreamMedia, IN ZWP2P_UINT32 dwUserData);
```

Description

Set the real-time stream media data callback function.

Parameter List

- `hStream` - Real-time stream handle (input)
- `cbStreamMedia` - Media data callback function (input), see its function definition for details.
- `dwUserData` - User data for callback (input).

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE` - Invalid handle, indicating an illegal handle value.

Real-Time Stream Control

Interface Name

```
int CALLSTACK ZWP2PVNClient_StreamMediaControl(IN ZWP2P_HSTREAM hStream, IN
eZWP2P_CMD_OPERATE eMediaControl);
```

Description

Start or stop real-time stream transmission.

Parameter List

- `hStream` - Real-time stream handle (input)
- `eMediaControl` - Start or stop.

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE` - Invalid handle, indicating an illegal handle value.

Disconnect the Real-Time Stream

Interface Name

```
int CALLSTACK ZWP2PVNClient_RealStreamDisconnect(IN ZWP2P_HSTREAM hStream);
```

Description

Disconnect from the real-time stream.
This will automatically stop real-time stream transmission.

Parameter List

- `hStream` - Real-time stream handle.

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE` - Invalid handle, indicating an illegal handle value.

Open and Close Audio

Interface Name

```
int CALLSTACK ZWP2PVNClient_OpenAudio(IN ZWP2P_HSTREAM hStream);
int CALLSTACK ZWP2PVNClient_CloseAudio(IN ZWP2P_HSTREAM hStream);
```

Description

Open and close audio.
By default, audio is not transmitted to save bandwidth. It can be enabled if necessary.

Parameter List

- `hStream` - Stream handle.

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE` - Invalid handle, indicating an illegal handle value.

History Stream Data Transmission Interface

Get the Video Distribution Overview for a Specific Month

Interface Name

```
int CALLSTACK ZWP2PVNClient_DataExistCHeck(IN ZWP2P_HUSER hUser, IN ZWP2P_UINT32 dwChannel, IN ZWP2P_CHAR *szYearMonth, OUT ZWP2P_CHAR *cResult);
```

Description

Get the video distribution overview for a specific month.

Parameter List

- `hUser` - User handle.
- `dwChannel` - Ignored, as the device currently does not support querying video distribution by channel.
- `szYearMonth` - Year and month string, e.g., “201708” for August 2017.
- `cResult` - Video distribution description string, with a length of 32 characters.
  Each character represents a day of the month. “1” means there is video, and “0” means there is no video. For example, "10101000000000000000000000000000" means video exists on the 1st, 3rd, and 5th of the month.

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.

Stop History Stream

Interface Name

```
int CALLSTACK ZWP2PVNClient_StopHistoryStream(IN ZWP2P_HUSER hUser);
```

Description

Stop history stream.

The stream handle will be invalidated after calling this function.

Return Values

- `eZWP2P_ERR_FAILED` - Failed. Check corresponding debug messages for the specific cause.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT` - Not initialized.
- `eZWP2P_ERR_SUCCESS` - Success.

History Stream Control

Interface Name

```
int CALLSTACK ZWP2PVNClient_HistoryStreamControl(IN ZWP2P_HSTREAM hHistory, IN eZWP2P_CMD_HISTORYSTREAM_OPERATE eHistoryStreamOperate);
```

Description

Start or stop history stream transmission.

Parameter List

- `hHistory`: Input parameter, history stream transmission handle.
- `eHistoryStreamOperate`: Operation type, either start or stop.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

Speed Transmission

Interface Name

```
int CALLSTACK ZWP2PVNClient_HistoryStreamSetSpeed(IN ZWP2P_HSTREAM hHistory, IN ZWP2P_UINT32 dwSpeed);
```

Description

Set the speed for history stream data transmission.

Note: Due to differences in device performance, the device may only send key frames at higher speeds. For example, most devices only send key frames at speeds above 4x.

Parameter List

- `hHistory`: Input parameter, history stream handle (will be filled after a successful connection).
- `dwSpeed`: Speed setting, options are 1, 2, 4, 8, and 16.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

Positioning

Interface Name

```
int CALLSTACK ZWP2PVNClient_HistoryStreamPosition(IN ZWP2P_HSTREAM hHistory, IN ZWP2P_DATETIME *cTime);
```

Description

Position the history stream to a specific time.
Note: It is important to remember the current value of `cPositionID` in the history stream header before calling this interface (denoted as A). After positioning, if the value of `cPositionID` in the media data returned to the application (denoted as B) is the same as A, the frame should be discarded. If B is different, it means the data after positioning has arrived, and the frame should be processed.

Parameter List

- `hHistory`: Input parameter, history stream transmission handle.
- `cTime`: Position time point.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

Pause/Resume Playback

The functionality is the same as the history stream control interface described above.

Destroy History Stream

Interface Name

```
int CALLSTACK ZWP2PVNClient_HistoryStreamDestroy(IN ZWP2P_HSTREAM hHistory);
```

Description

Destroy history stream transmission resources.
It is crucial to destroy the resources when no longer needed to prevent memory leaks.

Parameter List

- `hHistory`: Input parameter, history stream transmission handle.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

PTZ Control Interface

Interface Name

```
int CALLSTACK ZWP2PVNClient_PTZControl(IN ZWP2P_HUSER hUser, IN ZWP2P_UINT32 uiChannel, IN eZWP2P_PTZ_CONTROL_CODE eCommandCode, IN ZWP2P_UINT32 uiParam1, IN ZWP2P_UINT32 uiParam2, IN ZWP2P_UINT32 uiParam3, ZWP2P_UINT32 uiParam4);
```

Description

Control PTZ operations, including direction control, preset points, trajectories, and cruise settings.

Parameter List

- `hUser`: Input parameter, user handle.
- `uiChannel`: Channel number.
- `eCommandCode`: PTZ operation command code, refer to the definition of `eZWP2P_PTZ_CONTROL_CODE`.
- `uiParam1`, `uiParam2`, `uiParam3`, `uiParam4`: Parameters specific to the PTZ command, refer to the documentation for details.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

Command Processing Callback

Interface Name

```
typedef int(CALLBACK *ZWP2P_CB_USER_CMD_REPLY)(IN ZWP2P_HUSER hUser,
ZWP2P_USER_CMD_REPLY * pstruReply, IN ZWP2P_UINT32 dwUserData);
```

Description

Command processing callback.

Parameter List

- `hUser`: Input parameter, user handle.
- `pstruReply`: Device response content.
- `dwCmd`: Command word, refer to the `ZWP2P_CMD_CODE` definition for details.
- `dwResult`: Result value, refer to `eZWP2P_ERR_CODE` for details.
- `cResponse`: Response content, the application layer should copy the corresponding structure data to its structure or use the structure pointer to access it.
- `dwUserData`: User data.

Return Values

- `int`: Ignored.

Set Command Processing Callback

Interface Name

```
int CALLSTACK ZWP2PVNClient_SetCmdReplyCB(IN ZWP2P_HUSER hUser,
ZWP2P_CB_USER_CMD_REPLY cbUserCmdReply, ZWP2P_UINT32 dwUserData);
```

Description

Set command processing callback function.

Parameter List

- `hUser`: Input parameter, user handle.
- `cbUserCmdReply`: Command processing callback function.
- `dwUserData`: User data, which will be passed back to the application layer when the callback is triggered.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

Device Information

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetDeviceInfo(IN ZWP2P_HUSER hUser);int CALLSTACK ZWP2PVNClient_GetDeviceInfoBlock(IN ZWP2P_HUSER hUser, OUT ZWP2P_DEVICE_INFO * pDeviceInfo);
```

Description

- `ZWP2PVNClient_GetDeviceInfo`: Asynchronously fetch device information.
- `ZWP2PVNClient_GetDeviceInfoBlock`: Synchronously fetch device information.

Parameter List

- `hUser`: Input parameter, user handle.
- `pDeviceInfo`: Output parameter, includes device model, name, serial number, etc.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

Device Capabilities

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetDeviceCapBlock(IN ZWP2P_HUSER hUser, ZWP2P_DEVICE_CAP * pDeviceCap);
```

Description

Get the capabilities of the device.
Note: The client should present certain features based on the device's capabilities. For instance, if the device does not support PTZ, the PTZ control interface should not be shown.

Parameter List

- `hUser`: Input parameter, user handle.
- `pDeviceCap`: Output parameter, includes device capabilities such as PTZ support, SD card support, and video channel count.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.

- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

Video Quality Level

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetDeviceVideoQulityBlock(IN ZWP2P_HUSER hUser, OUT ZWP2P_VIDEO_QUALITY_LEVEL * pVideoQulity);
```

Description

Get the video quality level.
Note: The application should present video quality options according to the returned video quality levels.

Parameter List

- `hUser`: Input parameter, user handle.
- `pVideoQulity`: Output parameter, includes the video quality level.

Return Values

- `eZWP2P_ERR_FAILED`: Failure, refer to corresponding debug messages for details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle, this means an illegal handle value was used.

System Time

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetDeviceSystemTimeBlock(IN ZWP2P_HUSER hUser, OUT ZWP2P_DEVICE_SYSTEM_TIME * pDeviceSystemTime);int CALLSTACK ZWP2PVNClient_SetDeviceSystemTimeBlock(IN ZWP2P_HUSER hUser, OUT ZWP2P_DEVICE_SYSTEM_TIME * pDeviceSystemTime);
```

Description

Get and set the device system time.

Parameter List

- `hUser`: Input parameter, user handle.
- `pDeviceSystemTime`: System time.

Return Values

- `eZWP2P_ERR_FAILED`: Failure. Check the corresponding debug messages for more details.

- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle. This means an illegal handle value was used.

Real-Time Stream Audio Enable

Interface Name

```
int CALLSTACK ZWP2PVNClient_AudioEnableBlock(IN ZWP2P_HUSER hUser, IN ZWP2P_UINT32 uiChannel, IN ZWP2P_UINT32 uiEnable);
```

Description

Enable or disable the transmission of real-time stream audio data.
This function is the same as the audio control interface in real-time streams, which is omitted here.

Parameter List

- `hUser`: Input parameter, user handle.
- `uiChannel`: Channel number.
- `uiEnable`:
  - `0`: Stop real-time stream audio data transmission.
  - `1`: Enable real-time stream audio data transmission.

Return Values

- `eZWP2P_ERR_FAILED`: Failure. Check the corresponding debug messages for more details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle. This means an illegal handle value was used.

Encoding Parameters

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetCodecBlock(IN ZWP2P_HUSER hUser, ZWP2P_UINT32 uiChannel, ZWP2P_UINT32 uiStreamType, ZWP2P_CODEC_CFG * pCodecCfg);int CALLSTACK ZWP2PVNClient_SetCodecBlock(IN ZWP2P_HUSER hUser, ZWP2P_UINT32 uiChannel, ZWP2P_UINT32 uiStreamType, ZWP2P_CODEC_CFG * pCodecCfg);
```

Description

Get and set encoding parameters.
Unlike getting and setting video quality levels, this interface provides more detailed settings for encoding parameters, such as resolution, frame rate, and bitrate. It is mainly used for setting encoding

parameters for the primary stream.
Once the video quality level is set, the corresponding stream will have its encoding parameters fixed.

Parameter List

- `hUser`: Input parameter, user handle.
- `uiChannel`: Channel number.
- `uiStreamType`: Stream type (also referred to as stream ID):
  - `0`: Primary stream.
  - `1`: Sub-stream.
  - `2`: High-definition sub-stream (mobile stream).
- `pCodecCfg`: Includes encoding type, resolution, frame rate, bitrate, and other encoding parameters.

Return Values

- `eZWP2P_ERR_FAILED`: Failure. Check the corresponding debug messages for more details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle. This means an illegal handle value was used.

Device Network Configuration

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetNetcardListBlock(IN ZWP2P_HUSER hUser, ZWP2P_NETCARD_LIST * pNetcardList);int CALLSTACK ZWP2PVNClient_SetNetcardListBlock(IN ZWP2P_HUSER hUser, ZWP2P_NETCARD_LIST * pNetcardList);
```

Description

Get and set network configuration.

Parameter List

- `hUser`: Input parameter, user handle.
- `pNetcardList`: The network configuration of all network cards supported by the current device, including basic network parameters like DHCP, DNS, MAC address, etc. (MAC address cannot be configured).

Return Values

- `eZWP2P_ERR_FAILED`: Failure. Check the corresponding debug messages for more details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle. This means an illegal handle value was used.

OSD Configuration

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetOSDBlock(IN ZWP2P_HUSER hUser, ZWP2P_UINT32 uiChannel, ZWP2P_OSD_CFG * pOSDCfg);int CALLSTACK ZWP2PVNClient_SetOSDBlock(IN ZWP2P_HUSER hUser, ZWP2P_UINT32 uiChannel, ZWP2P_OSD_CFG * pOSDCfg);
```

Description

Get and set OSD configuration (channel name, time OSD).
Note: Channel names do not support Chinese characters.

Parameter List

- `hUser`: Input parameter, user handle.
- `pOSDCfg`: OSD configuration, including:
  - Position of the channel name and whether it is displayed.
  - Position and display format of the time OSD.

Return Values

- `eZWP2P_ERR_FAILED`: Failure. Check the corresponding debug messages for more details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle. This means an illegal handle value was used.

SD Card

Interface Name

```
int CALLSTACK ZWP2PVNClient_GetSDCardBlock(IN ZWP2P_HUSER hUser, ZWP2P_SDCARD_INFO* pSDCardInfo);int CALLSTACK ZWP2PVNClient_FormatSDCardBlock(IN ZWP2P_HUSER hUser);
```

Description

Get basic SD card information and format the SD card.

Parameter List

- `hUser`: Input parameter, user handle.
- `pSDCardInfo`: Includes SD card brand, capacity, status, and other basic SD card information.

Return Values

- `eZWP2P_ERR_FAILED`: Failure. Check the corresponding debug messages for more details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.

- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle. This means an illegal handle value was used.

Restore Factory Configuration

Interface Name

```
int CALLSTACK ZWP2PVNClient_RestoreFactoryBlock(IN ZWP2P_HUSER hUser);
```

Description

Restore factory configuration.

Reboot

Interface Name

```
int CALLSTACK ZWP2PVNClient_RemoteRebootBlock(IN ZWP2P_HUSER hUser);
```

Description

Reboot the device.

Modify User Password

Interface Name

```
int CALLSTACK ZWP2PVNClient_ModifyPasswordBlock(IN ZWP2P_HUSER hUser,
ZWP2P_DEVICE_MODIFY_USER_PASSWORD * pModifyUserPassword);
```

Description

Modify the user password.

HTTP API Request Forwarding

Interface Name

```
int CALLSTACK ZWP2PVNClient_TransferCGICommand(IN ZWP2P_HUSER hUser, char*
pHttpRequest, int dwRequestLen, char** ppResponse, int* pResponseLen);int CALLSTACK
ZWP2PVNClient_ReleaseTransferCGIBuf(char* pHttpResponse);
```

Description

For devices that support HTTP API access, this interface sends an HTTP API request, and the device will reply with a standard HTTP response. After obtaining the response, the client needs to parse the reply content, which is usually in HTTP+XML format.

This interface is generally used when the SDK does not support certain configurations, but the client wishes to quickly support it.

Parameter List

- `hUser`: Input parameter, user handle.
- `pHttpRequest`: Complete HTTP request string.
- `dwRequestLen`: Length of the HTTP request string.
- `ppResponse`: HTTP response content.
- `pResponseLen`: Length of the HTTP response content.

Return Values

- `eZWP2P_ERR_FAILED`: Failure. Check the corresponding debug messages for more details.
- `eZWP2P_ERR_FAILED_SDK_NOT_INIT`: Not initialized.
- `eZWP2P_ERR_SUCCESS`: Success.
- `eZWP2P_ERR_FAILED_INVALID_HANDLE`: Invalid handle. This means an illegal handle value was used.

Data Structure Detailed Explanation

Video Quality Level

Definition

```
typedef enum
{
    eZWP2P_QUALITY_UNKNOWN = 0x00, // Use the current actual quality, do not change current encoding parameters
    eZWP2P_QUALITY_MAX = 0x01,     // Maximum quality
    eZWP2P_QUALITY_HIGH = 0x02,    // High quality
    eZWP2P_QUALITY_MIDDLE = 0x03,  // Medium quality
    eZWP2P_QUALITY_LOW = 0x04,     // Low quality
    eZWP2P_QUALITY_MIN = 0x05,     // Minimum quality
    eZWP2P_QUALITY_SUPER = 0x06,   // Super quality
    eZWP2P_QUALITY_DECIDE_BY_DEVICE = 0x20, // Device decides adaptively
} ZWP2P_QUALITY_LEVEL;
```

Description

This enum defines the video quality levels.

- `eZWP2P_QUALITY_SUPER`: Typically, the SUPER quality level corresponds to the main stream.
  Important: It is not recommended to use `eZWP2P_QUALITY_UNKNOWN` and `eZWP2P_QUALITY_DECIDE_BY_DEVICE` as these will be gradually deprecated.

Real-Time Stream Media Frame Header Structure

Definition

```
typedef struct tagZWP2P_STREAMMEDIA_FRAME_HEADER
{
    ZWP2P_UINT32 uiFrameType;  //!< Frame type, refer to the definition of
ZWP2P_FRAMETYPE
    ZWP2P_UINT32 uiFrameNo;    //!< Frame number
    ZWP2P_UINT32 uiChannel;    //!< Channel number, starts from 0
    ZWP2P_CHAR cStreamID;      //!< Stream ID. 0 for main stream, 1 for sub-stream
    ZWP2P_CHAR cCodecID;       //!< Codec type, refer to the definition of
ZWP2P_CODECID
    ZWP2P_UINT32 uiSec;        //!< Absolute timestamp in seconds (since 1970)
    ZWP2P_UINT32 uiMSec;       //!< Absolute timestamp in milliseconds (for display or
control playtime)
    ZWP2P_CHAR cPositionID;    //!< Used when positioning or switching video quality
    ZWP2P_CHAR cReserve[3];    //!< Reserved space
    ZWP2P_CHAR* pBuffer;       //!< Pointer to the data buffer
    ZWP2P_UINT32 dwBufferLen;  //!< Length of the data
} ZWP2P_STREAMMEDIA_FRAME_HEADER, *LPZWP2P_STREAMMEDIA_FRAME_HEADER;
```

Description

This structure is used for real-time stream media data callbacks. Each callback corresponds to one frame of media data, including basic frame information and raw media data.

Member List

- `uiFrameType`: Frame type
- `uiFrameNo`: Frame number. The frame numbers for all video frames are continuous. If frames are not continuous, frame loss should be handled by discarding frames until the next keyframe, otherwise, it may cause visual artifacts or crashes. All audio frames have continuous frame numbers.
- `uiChannel`: Channel number
- `cStreamID`: Stream ID
- `cCodecID`: Codec type, e.g., H264, H265, G711U
- `uiSec`: Absolute timestamp in seconds since 1970
- `uiMSec`: Absolute timestamp in milliseconds (Note: divide by 1000 for milliseconds). Be careful when processing in the application layer.
- `cPositionID`: This field is invalid for real-time stream media data.
  For historical stream media data, this field is valid. Each time the position is adjusted, this value will change. Since the device, network protocol stack, and application layer may all cache media data during transmission, it is necessary to differentiate between pre-positioning and post-positioning data. If the `cPositionID` matches the value at the time of positioning, it should be discarded. Otherwise, it represents fresh data that should be processed.
- `cReserve`: Reserved space

- `pBuffer`: Media data buffer, consisting of two parts: reserved space and standard raw media data. The reserved space is by default 28 bytes. The reserved space length can be set by the `ZWP2PVNClient_SetAttribute` interface. This reserved space is for scenarios where the application layer needs to add private frame headers before the raw data to avoid unnecessary memory copies and facilitate integration into existing software systems for playback, storage, forwarding, etc.
  After the reserved space, the raw standard media data follows, which can be decoded using any compatible decoder.
- `dwBufferLen`: Length of the standard raw media data (excluding the reserved space).