

Device Network SDK

Developer Guide

Table of Contents

1 Overview.....	12
1.1 Introduction	12
1.2 SDK Component	12
1.3 System Requirements	12
2 Function Modules.....	13
2.1 SDK Initialization	13
2.1.1 Introduction	13
2.1.2 Process	13
2.1.3 Example Code.....	14
2.2 Real-time Monitoring.....	14
2.2.1 Introduction	14
2.2.2 Process	15
2.2.3 Example Code.....	17
2.3 Record Search	20
2.3.1 Introduction	20
2.3.2 Process	20
2.3.3 Example Code.....	21
2.4 Record Playback	23
2.4.1 Introduction	23
2.4.2 Process	24
2.4.3 Example Code.....	25
2.5 Record Download.....	29
2.5.1 Introduction	29
2.5.2 Process	31
2.5.3 Example Code.....	32
2.6 Talk Data Transparent Transmission.....	36
2.6.1 Introduction	36
2.6.2 Progress.....	37
2.6.3 Example Code.....	38
2.7 Talk and Broadcast	40
2.7.1 Introduction	40
2.7.2 Progress.....	41
2.7.3 Example Code.....	42
2.8 Alarm Listen	43
2.8.1 Introduction	43
2.8.2 Progress.....	44
2.8.3 Example Code.....	45
2.9 Device Search	47
2.9.1 Instroduction.....	47
2.9.2 Progress.....	47
2.9.3 Example Code.....	48
2.10 Device Config	50

2.10.1 Introduction	50
2.10.2 Progress.....	50
2.10.3 Example Code.....	52
2.11 Smart Stream	59
2.11.1 Introduction	59
2.11.2 Progress.....	60
2.11.3 Example Code.....	61
3 API Reference	64
3.1 SDK Initialization	64
3.1.1 SDK Initialization	64
3.1.2 SDK Cleanup	64
3.2 Real-time Monitoring.....	65
3.2.1 Create Stream Player	65
3.2.2 Delete Stream Player	65
3.2.3 Attach Callback to the Player	66
3.2.4 Detach Callback from the Player.....	66
3.2.5 Attach the Decoded Data Callback Function.....	67
3.2.6 Detach the Decoded Data Callback Function.....	67
3.2.7 Open Real-time Stream Player.....	68
3.2.8 Close the Stream Player	68
3.2.9 Start to Request the Stream.....	69
3.2.10 Stop Requesting the Stream	69
3.2.11 Audio Control of the Player.....	70
3.2.12 Switch the Real-time Stream Type.....	70
3.2.13 Show Smart Information.....	71
3.2.14 Video Snapshot	71
3.2.15 Start to Record	72
3.2.16 Stop Recording	72
3.2.17 Set Zoom Operation Window	73
3.2.18 Set Zoom Region	73
3.2.19 Set Zoom Region Being Selected	74
3.2.20 Get the Count of Smart Information.....	74
3.2.21 Get the Smart Information.....	75
3.3 Record Search	76
3.3.1 Search All Record File.....	76
3.3.2 Create Record Search Instance	77
3.3.3 Delete the Record Search Instance	77
3.3.4 Search the Next Set of Files	78
3.4 Record Playback	78
3.4.1 Create Stream Player	78
3.4.2 Delete Stream Player	79
3.4.3 Attach Callback to the Player	79
3.4.4 Detach Callback from the Player.....	80
3.4.5 Attach the Decoded Data Callback Function.....	80

3.4.6 Detach the Decoded Data Callback Function.....	81
3.4.7 Open Playback Stream Player by File	81
3.4.8 Close the Stream Player	82
3.4.9 Start to Request the Stream.....	82
3.4.10 Stop Requesting the Stream	83
3.4.11 Audio Control of the Player.....	83
3.4.12 Pause Request the Stream	84
3.4.13 Resume Request the Stream.....	84
3.4.14 Seek to the Specified Time.....	85
3.4.15 Seek to the Specified File	85
3.4.16 Play the Next File	86
3.4.17 Play the Previous File	86
3.4.18 Set the Speed of Playback.....	87
3.4.19 Single Frame Playback.....	87
3.4.20 Set the Transmission Bitrate of the Stream	88
3.4.21 Get the Time of Playback	88
3.4.22 Video Snapshot	89
3.4.23 Set the Playback Mode	89
3.4.24 Synchronous Playback.....	90
3.4.25 Set Zoom Operation Window	90
3.4.26 Set Zoom Region	91
3.4.27 Show Zoom Region Being Selected.....	91
3.5 Record Download	92
3.5.1 Create Download Instance	92
3.5.2 Delete the Download Instance.....	92
3.5.3 Attach Callback Function.....	93
3.5.4 Detach Callback Function.....	93
3.5.5 Open the Download Instance by Time.....	94
3.5.6 Open the Download Instance by File	95
3.5.7 Close the Download Instance	95
3.5.8 Start to Download	96
3.5.9 Stop Downloading	96
3.5.10 Get the Download Progress	97
3.5.11 Get the Download Bitrate	97
3.6 Talk Data Transparent Transmission.....	98
3.6.1 Create Talk Instance	98
3.6.2 Delete the Talk Instance	98
3.6.3 Attach Callback Function.....	99
3.6.4 Detach Callback Function.....	99
3.6.5 Open the Talk Instance.....	100
3.6.6 Send Audio Data to Device.....	101
3.6.7 Close the Talk Instance.....	101
3.7 Talk and Broadcast	102
3.7.1 Create Talk Instance	102

3.7.2 Delete the Talk Instance	102
3.7.3 Start to Talk	103
3.7.4 Stop Talking	103
3.8 Alarm Listen	104
3.8.1 Create Alarm Listen Instance	104
3.8.2 Delete the Alarm Listen Instance	104
3.8.3 Attach Callback Function.....	105
3.8.4 Detach Callback Function.....	105
3.8.5 Open the Alarm Listen Instance.....	106
3.8.6 Close the Alarm Listen Instance.....	106
3.9 Device Search	107
3.9.1 Search All Device in LAN	107
3.10 Device Remote Config.....	107
3.10.1 Create Device Instance.....	107
3.10.2 Delete the Device Instance	108
3.10.3 Attach Callback Function.....	108
3.10.4 Detach Callback Function.....	109
3.10.5 Open the Device Instance	109
3.10.6 Open the Device Instance (ex)	110
3.10.7 Close the Device Instance	110
3.10.8 Export Config.....	111
3.10.9 Import Config	111
3.10.10 Device Upgrade	112
3.10.11 Get the Device Information	112
3.10.12 Get the Device Time.....	113
3.10.13 Set the Device Time	113
3.10.14 Get the Device OSD Config	114
3.10.15 Set the Device OSD Config	114
3.10.16 Get the Device NTP Config.....	115
3.10.17 Set the Device NTP Config	115
3.10.18 Get the User Account Config	116
3.10.19 Add User Group	116
3.10.20 Modify User Group	117
3.10.21 Delete User Group	117
3.10.22 Add User Account	118
3.10.23 Modify User Account	118
3.10.24 Delete User Account	119
3.10.25 Modify User Password	119
3.10.26 Get Channel State	120
3.10.27 Get Face Detection Config	120
3.10.28 Set Face Detection Config.....	121
3.10.29 Get Line Crossing Config	121
3.10.30 Set Line Crossing Config.....	122
3.10.31 Get Area Intrusion Config	122

3.10.32 Set Area Intrusion Config	123
3.10.33 Get Unattended Object Config	123
3.10.34 Set Unattended Object Config	124
3.10.35 Get Object Missing Config	124
3.10.36 Set Object Missing Config	125
3.10.37 Get Region Entrance Config	125
3.10.38 Set Region Entrance Config	126
3.10.39 Get Region Exiting Config	126
3.10.40 Set Region Exiting Config	127
3.10.41 Get Fast Moving Config	127
3.10.42 Set Fast Moving Config	128
3.10.43 Get Loitering Detection Config	128
3.10.44 Set Loitering Detection Config	129
3.10.45 Get People Gathering Detection	129
3.10.46 Set People Gathering Detection	130
3.10.47 Get Parking Detection Config	130
3.10.48 Set Parking Detection Config	131
3.10.49 Get Scene Change Detection Config	131
3.10.50 Set Scene Change Detection Config	132
3.10.51 Get Blurred Detection Config	132
3.10.52 Set Blurred Detection Config	133
3.10.53 Get Audio Exception Detection	133
3.10.54 Set Audio Exception Detection	134
3.10.55 Get Crossing Line Statistics Config	134
3.10.56 Set Crossing Line Statistic Config	135
3.10.57 Clear Crossing Line Statistics Config	135
3.10.58 Get Crossing Line Statistics Report	136
3.10.59 Get Smart Ability	136
3.10.60 Get the Audio In Ability	137
3.10.61 Get Whether Storage Normal	137
3.10.62 Set Device TCP/IP Config	138
3.10.63 Modify the Device Network in LAN	138
3.10.64 Reboot Device	139
3.10.65 Restore the Factory Setting	139
3.10.66 Get Device Alarm In Config	140
3.10.67 Set Device Alarm In Config	140
3.10.68 Get device Alarm Out Config	141
3.10.69 Set Device Alarm Out Config	141
3.10.70 PTZ Control	142
3.10.71 Create Device Online Manage Instance	143
3.10.72 Query Device Whether Online	143
3.10.73 Delete the Device Online Manage Instance	144
3.10.74 Get Device Channel Number	144
3.10.75 Get Device Record Status	145

3.10.76	Get Log Config Ability.....	145
3.10.77	Query Log Information.....	146
3.10.78	Clear Log Information	146
3.10.79	Get Encode Config.....	147
3.10.80	Get FPS Config.....	147
3.10.81	Get Bitrate Config.....	148
3.10.82	Set Encode Config	148
3.10.83	Get Sub Stream Resolutuin	149
3.10.84	Release Encode Config Data	149
3.10.85	Get Area Cover Config.....	150
3.10.86	Set Area Cover Config	150
3.10.87	Get OSD Config.....	151
3.10.88	Set OSD Config	151
3.10.89	Get Device TCP/IP Config	152
3.10.90	Check IP Conflict	152
3.10.91	Set Device TCP/IP Config.....	153
3.10.92	Release TCP/IP Config Data	153
3.11	Smart Stream	154
3.11.1	Create Smart Stream Instance	154
3.11.2	Delete the Smart Stream Instance.....	154
3.11.3	Attach Callback Function.....	155
3.11.4	Detach Callback Function.....	155
3.11.5	Open the Smart Stream Instance.....	156
3.11.6	Close the Smart Stream Instance.....	156
3.11.7	Get Picture in the Smart Frame	157
3.11.8	Get Face Count.....	157
3.11.9	Get the Face Picture Instance	158
3.11.10	Get Face Information	158
3.11.11	Get Face Attribute	159
3.11.12	Get Face Picture	159
3.11.13	Get Face Picture with Shoulder	160
3.11.14	Get License Plate Count	160
3.11.15	Get the License Plate Picture Instance.....	161
3.11.16	Get License Plate Information.....	161
3.11.17	Get License Plate Picture	162
4	Callback.....	162
4.1	Create SDK Callback Function	162
4.2	Delete SDK Callback Function	163
4.3	The Format of the TDKCALLBACK.....	163
4.3.1	Preview and Playback	164
4.3.2	Record Download.....	164
4.3.3	Talk Data.....	164
4.3.4	Alarm Listen	164
4.3.5	Device Config	166

4.3.6 Smart Stream	166
4.4 Stream State Code	167
5 Structure Definition	168
5.1 TDK_FILE_INFO.....	168
5.2 TDK_SYSTEM_TIME	169
5.3 RECT	169
5.4 TDK_DOWNLOAD_PARAM_TIME.....	170
5.5 TDK_DOWNLOAD_PARAM_FILE	171
5.6 TDK_FILE_SEARCH_PARAM.....	172
5.7 TDK_FRAMEHEAD	173
5.8 TDK_PICTURE	174
5.9 TDK_ALARM_INFO	175
5.10 TDK_ALARM_INFO_CHN	175
5.11 TDK_SNAPSHOT_INFO	176
5.12 TDK_SNAPSHOT_PIC	176
5.13 TDK_PICTURE_DATA.....	177
5.14 TDK_FACE_SEARCH_INFO	177
5.15 TDK_DEVICE_SEARCH_PARAM	178
5.16 TDK_DEVICE_INFO	179
5.17 TDK_TIME_CONFIG	180
5.18 TDK_OSD_CONFIG	180
5.19 TDK_OSD_WIDGET.....	181
5.20 TDK_RECT32.....	181
5.21 TDK_NTP_CONFIG.....	182
5.22 TDK_NTP_INFO	182
5.23 TDK_USER_MANAGE_CONFIG.....	183
5.24 TDK_OPR_RIGHT	183
5.25 TDK_USER_GROUP_CONFIG.....	184
5.26 TDK_USER_CONFIG.....	185
5.27 TDK_CHANNEL_STATE	185
5.28 TDK_FACE_DETECTION_CONFIG.....	186
5.29 TDK_AREA	186
5.30 TDK_RULE_FACE_DETECT	187
5.31 TDK_RULE_FACE_DETECT_RANGE.....	187
5.32 TDK_SMART_ALARM	187
5.33 TDK_SCHEDULE	190
5.34 TDK_PTZ_LINK.....	191
5.35 TDK_PTZ_LINK_RANGE	191
5.36 TDK_LINE_CROSSING_CONFIG	192
5.37 TDK_LINE.....	192
5.38 TDK_RULE_LINE_CROSSING.....	193
5.39 TDK_RULE_LINE_CROSSING_RANGE	193
5.40 TDK_AREA_INTRUSION_CONFIG	194
5.41 TDK_RULE_AREA_INTRUSION.....	194

5.42 TDK_RULE_AREA_INSTRUSION_RANGE	195
5.43 TDK_UNATTENDED_OBJECT_CONFIG.....	196
5.44 TDK_RULE_UNATTENDED_OBJECT	196
5.45 TDK_RULE_UNATTENDED_OBJECT_RANGE.....	197
5.46 TDK_OBJECT_MISSING_CONFIG	198
5.47 TDK_RULE_OBJECT_MISSING	198
5.48 TDK_RULE_OBJECT_MISSING_RANGE.....	199
5.49 TDK_REGION_ENTRANCE_CONFIG.....	200
5.50 TDK_RULE_REGION_ENTRANCE	200
5.51 TDK_RULE_REGION_ENTRANCE_RANGE.....	201
5.52 TDK_REGION_EXITING_CONFIG	201
5.53 TDK_RULE_REGION_EXITING.....	202
5.54 TDK_RULE_REGION_EXITING_RANGE	202
5.55 TDK_FAST_MOVING_CONFIG	203
5.56 TDK_RULE_FAST_MOVING.....	203
5.57 TDK_RULE_FAST_MOVING_RANGE	204
5.58 TDK_LOITERING_DETECTION	204
5.59 TDK_RULE_LOITERING_DETECTION.....	205
5.60 TDK_RULE_LOITERING_DETTECTION_RANGE	205
5.61 TDK_PEOPLE_GATHERING_DETECTION	206
5.62 TDK_RULE_PEOPLE_GATHERING_DETECTION.....	206
5.63 TDK_RULE_PEOPLE_GATHERING_DETECTION_RANGE	207
5.64 TDK_PARKING_DETECTION	208
5.65 TDK_RULE_PARKING_DETECTION.....	208
5.66 TDK_RULE_PARKING_DETECTION_RANGE	209
5.67 TDK_SCENE_CHANGE_DETECTION	210
5.68 TDK_RULE_SCENE_CHANGE_DETECTION.....	210
5.69 TDK_RULE_SCENE_CHANGE_DETECTION_RANGE	211
5.70 TDK_BLURRED_DETECTION	211
5.71 TDK_RULE_BLURRED_DETECTION	212
5.72 TDK_RULE_BLURRED_DETECTION_RANGE	212
5.73 TDK_AUDIO_EXCEPTION_DETECTION	213
5.74 TDK_STRONG_SOUND_INTENSITY.....	214
5.75 TDK_STRONG_SOUND_INTENSITY_RANGE	214
5.76 TDK_SOUND_INTENSITY_DROPPED_SHARPLY	215
5.77 TDK_SOUND_INTENSITY_DROPPED_SHARPLY_RANGE.....	215
5.78 TDK_CROSSING_LINE_STATISTICS	216
5.79 TDK_RULE_CROSSING_LINE_STATISTICS.....	217
5.80 TDK_CROSSING_LINE_STATISTICS_REPORT	218
5.81 TDK_CROSSING_LINE_STATISTICS_DATA.....	218
5.82 TDK_TCPIP	219
5.83 TDK_NETWORK	220
5.84 TDK_ALARM_IN.....	221
5.85 TDK_ALARM_OUT	221

5.86 TDK_PTZ_PARAM	222
5.87 TDK_SYS_RECORDSTATUS	223
5.88 TDK_SYS_LOGINFOCFG	223
5.89 TDK_SYS_LOGININFOQUERY	224
5.90 TDK_ENC_CHNCFG.....	224
5.91 TDK_ENC_CFG	225
5.92 TDK_ENC_RES	227
5.93 TDK_ENC_QUERYBITRATE	228
5.94 TDKENCONDcfgDATA	229
5.95 TDKSTREAMDATA	230
5.96 TDK_ENC_QUERYRES	231
5.97 TDK_ENC_COVER	232
5.98 TDK_ENC_OSD	232
5.99 TDK_DEVICE_TCPIP	233
5.100 TDK_NETWORK_TCP	234
5.101 TDK_NETWORK_RTSP	236
5.102 TDK_NETWORK_DNS	236
5.103 TDK_NETWORK_ADAPTER.....	237
5.104 TDK_SMART_PIC	238
5.105 TDK_FACEOBJ	239
5.106 TDK_FACE_ATTR.....	240
5.107 TDK_LICENCEPLATE_OBJ.....	241
5.108 TDK_SMART_OBJ	242
5.109 TDK_DEVICE_NET_INFO.....	242
6 Enumeration.....	245
6.1 TDK_IMAGE_FORMAT	245
6.2 TDK_RECORD_FORMAT	245
6.3 TDK_DEVICE_TYPE	245
6.4 TDK_ENUM_CHANNEL_STATE	246
6.5 ENUM_TDK_CROSSING_LINE_STATISTICS_REPORT.....	246
6.6 ENUM_TDK_SMART	247
6.7 ENUM_TDK_ALARM_IN_TYPE	247
6.8 ENUM_TDK_ALARM_OUT_TYPE.....	248
6.9 ENUM_TDK_ALARM_OUT_STATUS.....	248
6.10 ENUM_TDK_PTZ_CONTROL	249
6.11 ENUM_TDK_FACE_ATTR	250
6.12 SMART_FRAME_INFO_TYPE	251
7 Error Code	252
8 URL	254
8.1 Real-time Monitoring.....	254
8.2 Remote Config.....	255
8.3 Record Search and Playback	256
8.4 Talk and Broadcast	257
8.5 Alarm Listen	258

1 Overview

1.1 Introduction

The manual introduces SDK interfaces reference information that includes main function modules, interface functions, and data structures.

The following are the main functions:

SDK initialization, real-time monitoring, record playback, record download, PTZ control, voice talk, video snapshot, alarm upload, device search, smart event upload and snapshot, user management, device restart, device upgrade, device time setting, video parameter setting, channel name setting, and network parameter setting.

1.2 SDK Component

The SDK component may be different dependent on the environment.

Table 1-1 SDK component for Windows

Library type	Library file name	Library file description
Function library	TDKSDK.h	External API file
	TDKSDK.lib	
	TDKSDK.dll	
	NpcMpiMonTsp.dll	P2P auxiliary library file
	TDKCloud.dll	
	libcrypto-1_1-x64.dll	
	libssl-1_1-x64.dll	
	certs/ca.pem	
	certs/client.key	
	certs/client.pem	

1.3 System Requirements

- For Windows

Windows 11, Windows 10, Windows 8, Windows 7, Windows Server 2012/2008/2003

- For Linux

The common Linux system such as RedHat, Ubuntu, CentOS.

2 Function Modules

2.1 SDK Initialization

2.1.1 Introduction

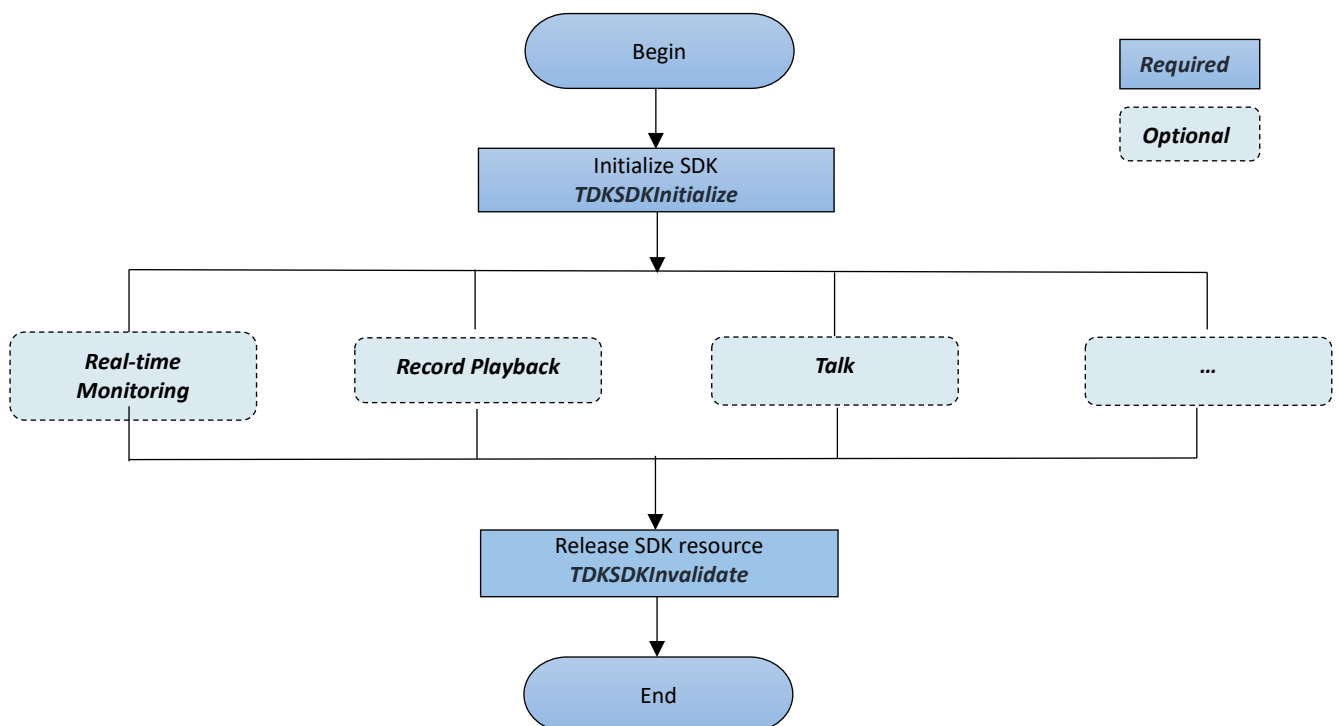
The initialization is required before conducting all the function modules.

- Initialization loads function modules and occupies some memory.
- It only needs to be called once.

Release resource is required by call `TDKSDKInvalidate` after integrating SDK.

2.1.2 Process

Figure 2.1-1 The process of initialize



Process Description

- Step 1 Call [`TDKSDKInitialize`](#) to initialize SDK.
- Step 2 (optional) Use the function modules as you needed.
- Step 3 After using SDK, call [`TDKSDKInvalidate`](#) to release SDK resource.

2.1.3 Example Code

```
#include <iostream>
#include "TDKSDK.h"
#pragma comment(lib, "TDKSDK.lib")

int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //TODO: Function module API

    //Release SDK resource
    TDKSDKInvalidate();
}
```

2.2 Real-time Monitoring

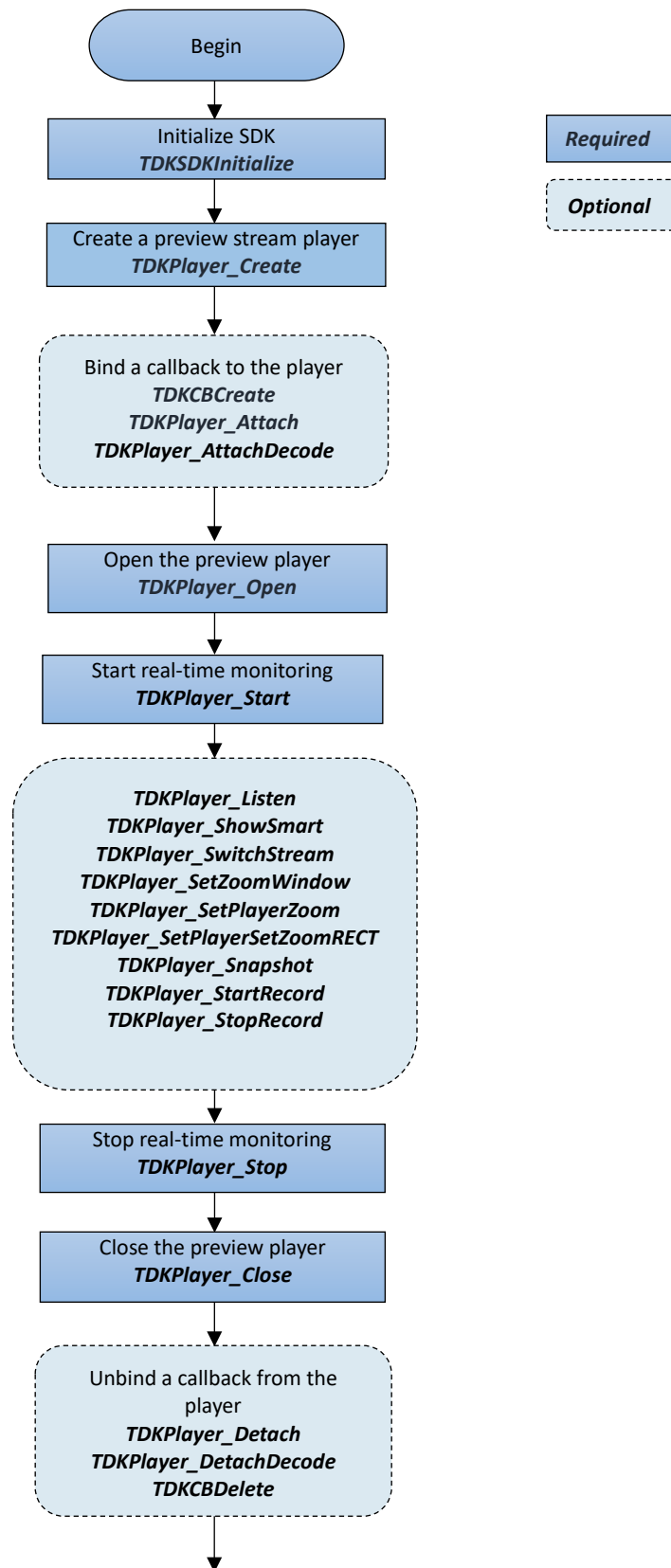
2.2.1 Introduction

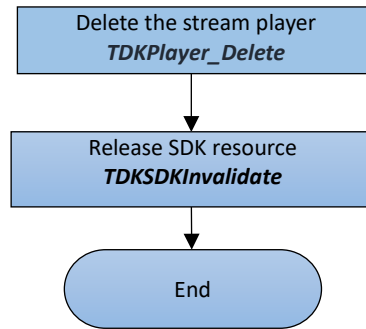
Real-time monitoring gets the live stream from the device.

- Support getting main stream, sub stream and third stream.
- Support decoding and playing stream data directly.
- Support calling back stream data to user for processing.
- Support saving stream data to file, according to the specified file name and format.

2.2.2 Process

Figure 2.2-1 Process of Real-time Monitoring





Process Description

- Step 1 Call [TDKSDKInvalitade](#) to initialize SDK.
- Step 2 Call [TDKPlayer_Create](#) to create a preview stream player. The parameter **hWnd** is used for displaying video picture. SDK will only request stream data but not display if **hWnd** is NULL.
- Step 3 (optional) Call [TDKCBCreate](#) to create a callback function.
- Step 4 (optional) Call [TDKPlay_Attach](#) to bind the callback to receive general messge and data as needed, such as login state, disconnect notification, coded data frame.
Call [TDKPay_AttahDecode](#) to bind the callback to receive decoded frame data as needed.
- Step 5 Call [TDKPlayer_Open](#) to open the preview player.
- Step 6 Call [TDKPlayer_Start](#) to start real-time monitoring. SDK will decode stream data if user calls [TDKPay_AttahDecode](#) to bind the callback to receive decoded frame data. SDK will decode and play video data if user sets parameter **hWnd** is valid when call [TDKPlayer_Create](#).
- Step 7 (optional) Call [TDKPlayer_Listen](#) to play or stop playing audio.
Call [TDKPlayer_ShowSmart](#) to show or hide the specified smart information on video picture.
Call [TDKPlayer_SwitchStream](#) to switch between different video streams.
Call [TDKPlayer_Snapshot](#) to capture a picture to save.
Call [TDKPlayer_StartRecord](#) to save coded stream data to file.
Call [TDKPlayer_StopRecord](#) to stop saving stream data.
- Step 8 Call [TDKPlayer_Stop](#) to stop requesting stream, decoding and displaying.
- Step 9 After using the function module, call [TDKPlayer_Close](#) to close the preview player.
- Step 10 (optional) Call [TDKPlayer_Detach](#) to unbind the callback for receiving general message.
Call [TDKPlayer_DetachDecode](#) to unbind the callback for receiving decoded stream data.
Call [TDKCBDelete](#) to delete the callback function.
- Step 11 Call [TDKPlayer_Delete](#) to destroy the preview stream player.

Step 12 Call [TDKSDKInvalidade](#) to release SDK resource.

2.2.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")

int TDKAPI callback_monitor(void* context, TDKU32 cmd, void* param0, void*
param1, void* param2)
{
    if (cmd == TDKPLAYER_MSG_STATE)
    {
        printf("Live Stream State = %d, error = %d\n",
(TDKU32)(size_t)param1, (TDKU32)(size_t)param2);
    }
    else if (cmd == TDKPLAYER_MSG_PACKET)
    {
        TDK_FRAMEHEAD* head = (TDK_FRAMEHEAD*)param1;
        TDKU8* data = (TDKU8*)(head + 1);
#if 0
        printf("Frame(%04d-%02d-%02d %02d:%02d:%02d.%03u), codec
= %d, type = %d, fps = %.3f, len = %d, \
            (%02x %02x %02x %02x-%02x %02x %02x %02x) ----
(%02x %02x %02x %02x %02x %02x %02x %02x)\n",
            head->tms.year + 2000, head->tms.month, head->tms.day,
            head->tms.hour, head->tms.minute, head->tms.second,
head->ms,
            head->codec, head->prefix[3] - TDKFRAME_BASE,
(float)(head->fps / 4.0), head->length,
            data[0], data[1], data[2], data[3], data[4], data[5], data[6],
data[7],
            data[8], data[9], data[10], data[11], data[12], data[13], data[14],
data[15]);
#endif
    }

    return TDK_OK;
}

void startMonitor()
```

```

{
    // Get window handle of control unit
    HMODULE hKernel32 = GetModuleHandle("kernel32");
    HWND hWnd = GetConsoleWindow();

    //Create stream player
    TDKHANDLE hPlayer = TDKPlayer_Create(hWnd, 0);
    if (!hPlayer)
    {
        printf("Create player failed!\n");
        return;
    }

    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL,
(TDKCALLBACK)&callback_monitor);
    if (hCallback)
    {
        //Register callback
        TDKPlayer_Attach(hPlayer, hCallback);
        TDKPlayer_AttachDecode(hPlayer, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }

    //Open stream player
    string url = "tdk://192.168.13.152:34567/mode=real&idc=1&ids=1";
    int ret = TDKPlayer_Open(hPlayer, url.c_str(), "admin", "tdk123456");
    if (ret != TDK_OK)
    {
        goto err1;
    }

    //Start monitor
    ret = TDKPlayer_Start(hPlayer);
    if (ret != TDK_OK)
    {
        goto err2;
    }

    //TODO: Call optional api.
    //

```

```

    printf("Input any key to quit.\n");
    getchar();

    //Stop monitor
    TDKPlayer_Stop(hPlayer);
err2:
    //Close stream player
    TDKPlayer_Close(hPlayer);
err1:
    if (hCallback)
    {
        //Unregister callback
        TDKPlayer_Detach(hPlayer, hCallback);
        TDKPlayer_DetachDecode(hPlayer, hCallback);
        //Destroy callback
        TDKCBDelete(hCallback);
    }
    //Destroy stream player
    TDKPlayer_Delete(hPlayer);
    return;
}

int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startMonitor();

    //Release SDK resource
    TDKSDKInvalidate();

    return 0;
}

```

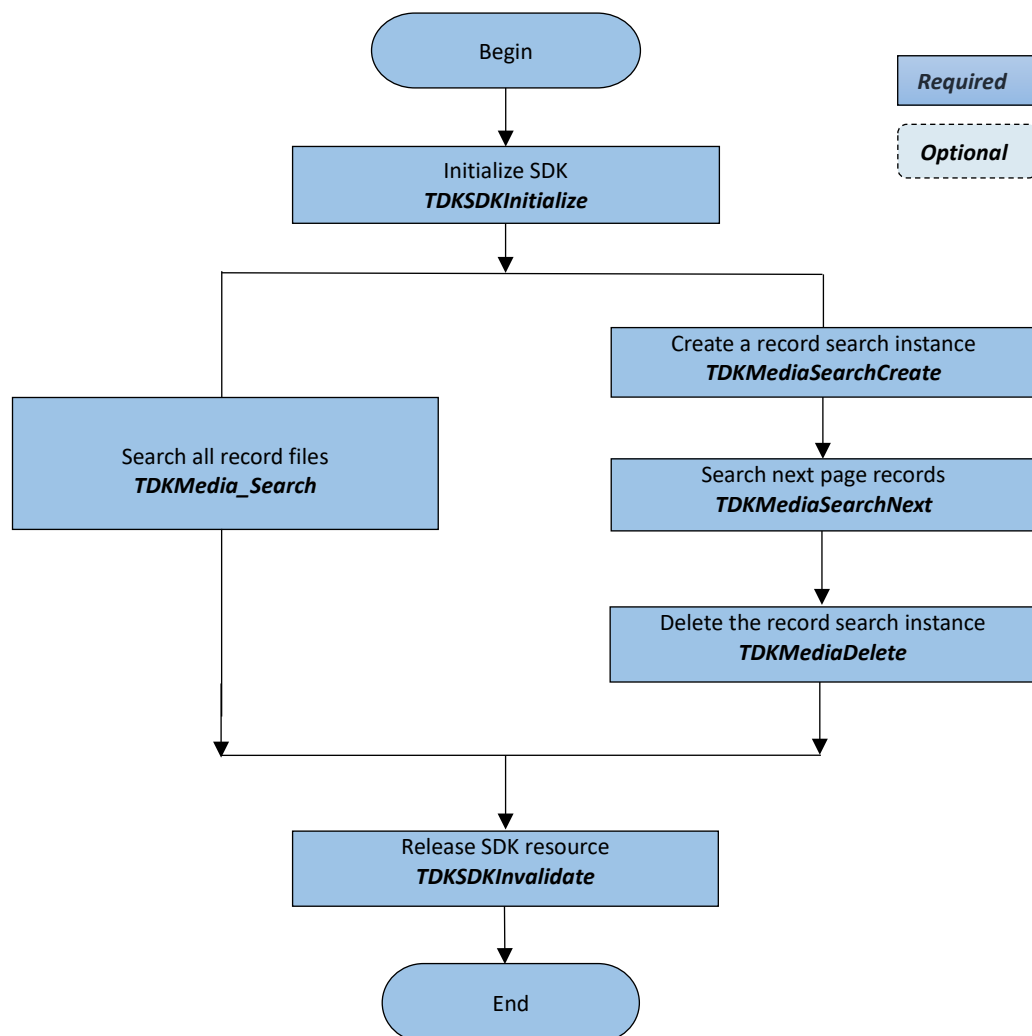
2.3 Record Search

2.3.1 Introduction

The record search function helps you find the records saved on the device. You can search all the records at once or page by page.

2.3.2 Process

Figure 2.3-1 Process of record search



Process Description

- Step 1 Call [TDKSDKInitialize](#) to initialize SDK.
- Step 2 Search the record files by one of the following two ways:
- Call [TDKMedia_Search](#) to get all records at once by maximum

quantity.

- Call [TDKMediaSearchCreate](#) to create a record search instance, then call [TDKMediaSearchNext](#) to search the next page records. You can specify the max request number of each page. After using, call [TDKMediaDelete](#) to delete the record search instance.

Step 3 Call [TDKSDKInvalidate](#) to release SDK resource.

2.3.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")

void startSearchRecord()
{
    string url = "tdk://192.168.13.197:34567/t";
    TDK_SYSTEM_TIME stime;    //start time
    stime.year = 2023;
    stime.month = 5;
    stime.day = 22;
    stime.hour = 10;
    stime.minute = 0;
    stime.second = 0;
    stime.isdst = 0;
    TDK_SYSTEM_TIME etime;    //end time
    etime.year = 2023;
    etime.month = 5;
    etime.day = 22;
    etime.hour = 12;
    etime.minute = 0;
    etime.second = 0;
    etime.isdst = 0;
    TDK_FILE_SEARCH_PARAM* pparam;
    TDK_FILE_SEARCH_PARAM param;
    param.filetype = 1;    //media
    param.rectype = 255;    //all
    param.chnmask0 = 1;
    param.chnmask1 = 0;
    param.streamtype = 0;    //main
    param.starttime = stime ;//
    param.endtime = etime;
```

```

pparam = &param;

//mode1
//Create media search handle
TDK_FILE_INFO* presult = new TDK_FILE_INFO[100];
memset(presult, 0, sizeof(TDK_FILE_INFO) * 100);

unsigned int resultcnt = 0;
int ret = TDKMedia_Search(url.c_str(), "admin", "", &param, 100, presult,
&resultcnt);
if (resultcnt > 0)
{
    for (unsigned int i = 0; i < resultcnt; i++)
    {

        TDK_FILE_INFO* pfileinfo = new TDK_FILE_INFO;
        memcpy(pfileinfo, &presult[i], sizeof(TDK_FILE_INFO));
        printf("%d.filename:%s\n", i+1, presult[i].filename);
    }
}
else
{
    //todo
}
delete[] presult;
}

int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startSearchRecord();

    //Release SDK resource

```

```
TDKSDKInvalidate();  
  
return 0;  
}
```

2.4 Record Playback

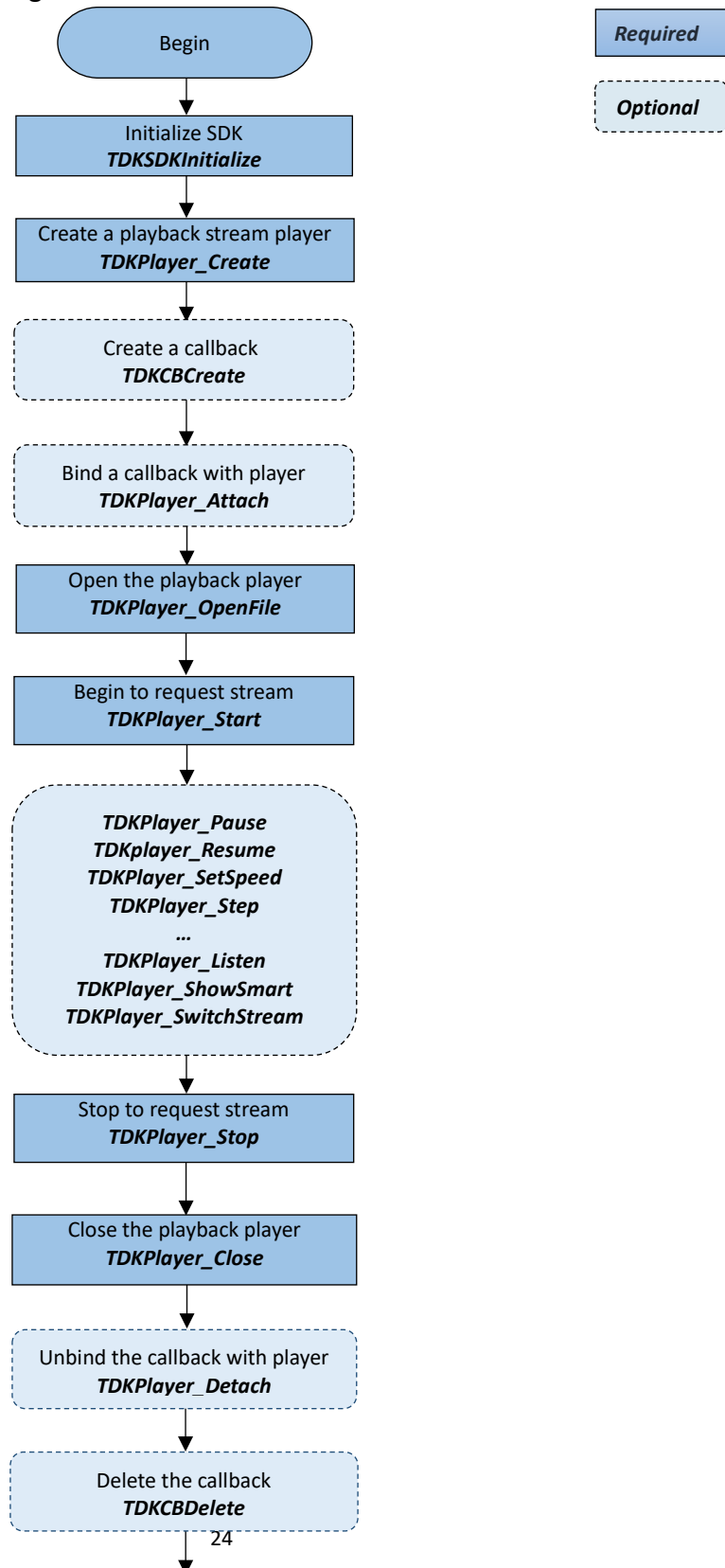
2.4.1 Introduction

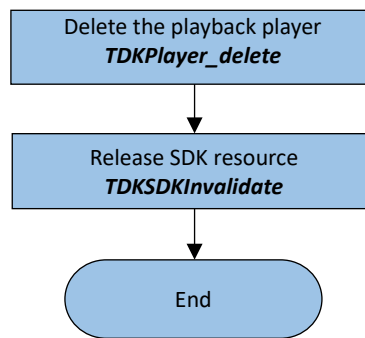
Record playback function plays the record files of certain channels by specific periods, to find target video for research.

The playback includes several operations, such as play, pause, fast play, slow play, backward play, frame-by-frame play and so on.

2.4.2 Process

Figure 2.4-1 Process of record





Process Description

- Step 1 Call [TDKSDKInitialize](#) to initialize SDK.
- Step 2 Call [TDKPlayer_Create](#) to create a playback stream player.
- Step 3 (optional) Call [TDKCBCreate](#) to create a callback.
- Step 4 (optional) After a callback created, call [TDKPlay_Attach](#) to bind the callback on player.
- Step 5 Call [TDKPlayer_OpenFile](#) to open the playback player.
- Step 6 Call [TDKPlayer_Start](#) to request stream.
- Step 7 (optional) Call [TDKPlayer_Listen](#) to play or stop audio.
Call [TDKPlayer_SetSpeed](#) to set the speed of playback.
- Step 8 Call [TDKPlayer_Stop](#) to stop requesting stream.
- Step 9 After using the function module, call [TDKPlayer_Close](#) to close the playback player.
- Step 10 (optional) Call [TDKPlayer_Detach](#) to unbind the callback.
- Step 11 (optional) After unbinding the callback, call [TDKCBDelete](#) to delete the callback.
- Step 12 Call [TDKPlay_Delete](#) to delete the playback stream player.
- Step 13 Call [TDKSDKInvalidate](#) to release SDK resource.

2.4.3 Example Code

```

#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")

int TDKAPI callback_playback(void* context, TDKU32 cmd, void* param0, void*
param1, void* param2)
{
    if (cmd == TDKPLAYER_MSG_STATE)

```

```

    {
        printf("Live Stream State = %d, error = %d\n", (TDKU32)(size_t)param1,
(TDKU32)(size_t)param2);
    }
    else if (cmd == TDKPLAYER_MSG_PACKET)
    {
        TDK_FRAMEHEAD* head = (TDK_FRAMEHEAD*)param1;
        TDKU8* data = (TDKU8*)(head + 1);
#ifdef 0
        printf("Frame(%04d-%02d-%02d %02d:%02d:%02d.%03u), codec = %d,
type = %d, fps = %.3f, len = %d, \
                (%02x %02x %02x %02x-%02x %02x %02x %02x) ----
(%02x %02x %02x %02x %02x %02x %02x %02x)\n",
                head->tms.year + 2000, head->tms.month, head->tms.day,
                head->tms.hour, head->tms.minute, head->tms.second, head->ms,
                head->codec, head->prefix[3] - TDKFRAME_BASE, (float)(head->fps
/ 4.0), head->length,
                data[0], data[1], data[2], data[3], data[4], data[5], data[6], data[7],
                data[8], data[9], data[10], data[11], data[12], data[13], data[14],
data[15]);
#endif
    }

    return TDK_OK;
}

void startPlayback()
{
    // Get window handle of control unit
    HMODULE hKernel32 = GetModuleHandle(L"kernel32");
    HWND hWnd = GetConsoleWindow();

    //Create stream player
    TDKHANDLE hPlayer = TDKPlayer_Create(hWnd, 1);
    if (!hPlayer)
    {
        printf("Create player failed!\n");
        return;
    }

    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&
callback_playback);
    if (hCallback)
    {

```

```

        //Register callback
        TDKPlayer_Attach(hPlayer, hCallback);
        TDKPlayer_AttachDecode(hPlayer, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }

    //search playback list
    string url = "tdk://192.168.13.197:34567/t";
    TDK_SYSTEM_TIME stime;    //start time
    stime.year = 2023;
    stime.month = 5;
    stime.day = 22;
    stime.hour = 10;
    stime.minute = 0;
    stime.second = 0;
    stime.isdst = 0;
    TDK_SYSTEM_TIME etime;    //end time
    etime.year = 2023;
    etime.month = 5;
    etime.day = 22;
    etime.hour = 12;
    etime.minute = 0;
    etime.second = 0;
    etime.isdst = 0;
    TDK_FILE_SEARCH_PARAM* pparam;
    TDK_FILE_SEARCH_PARAM param;
    param.filetype = 1;    //media
    param.rectype = 255;    //all
    param.chnmask0 = 1;
    param.chnmask1 = 0;
    param.streamtype = 0;    //main
    param.starttime = stime;
    param.endtime = etime;
    pparam = &param;

    //mode1
    //Create media search handle
    TDK_FILE_INFO* presult = new TDK_FILE_INFO[100];
    memset(presult, 0, sizeof(TDK_FILE_INFO) * 100);

```

```

    unsigned int resultcnt = 0;
    TDKMedia_Search(url.c_str(), "admin", "", &param, 100, presult,
&resultcnt);

    if (resultcnt > 0)
    {
        for (unsigned int i = 0; i < resultcnt; i++)
        {

            TDK_FILE_INFO* pfileinfo = new TDK_FILE_INFO;
            memcpy(pfileinfo, &presult[i], sizeof(TDK_FILE_INFO));

        }
    }
    else
    {
        return;
        //todo
    }

    //Open stream player

    int ret = TDKPlayer_OpenFile(hPlayer, url.c_str(), "admin",
"",presult,resultcnt);
    if (ret != TDK_OK)
    {
        goto err1;
    }

    //Start monitor
    ret = TDKPlayer_Start(hPlayer);
    if (ret != TDK_OK)
    {
        goto err2;
    }

    //TODO: Call optional api.
    //

    printf("Input any key to quit.\n");
    getchar();

    //Stop monitor

```

```

        TDKPlayer_Stop(hPlayer);
err2:
    //Close stream player
    TDKPlayer_Pause(hPlayer);
err1:
    if (hCallback)
    {
        //Unregister callback
        TDKPlayer_Detach(hPlayer, hCallback);
        TDKPlayer_DetachDecode(hPlayer, hCallback);
        //Destroy callback
        TDKCBDelete(hCallback);
    }
    delete[] presult;
    //Destroy stream player
    TDKPlayer_Delete(hPlayer);
    return;
}
int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startPlayback();

    //Release SDK resource
    TDKSDKInvalidate();

    return 0;
}

```

2.5 Record Download

2.5.1 Introduction

The record download function helps you obtain the records saved on the

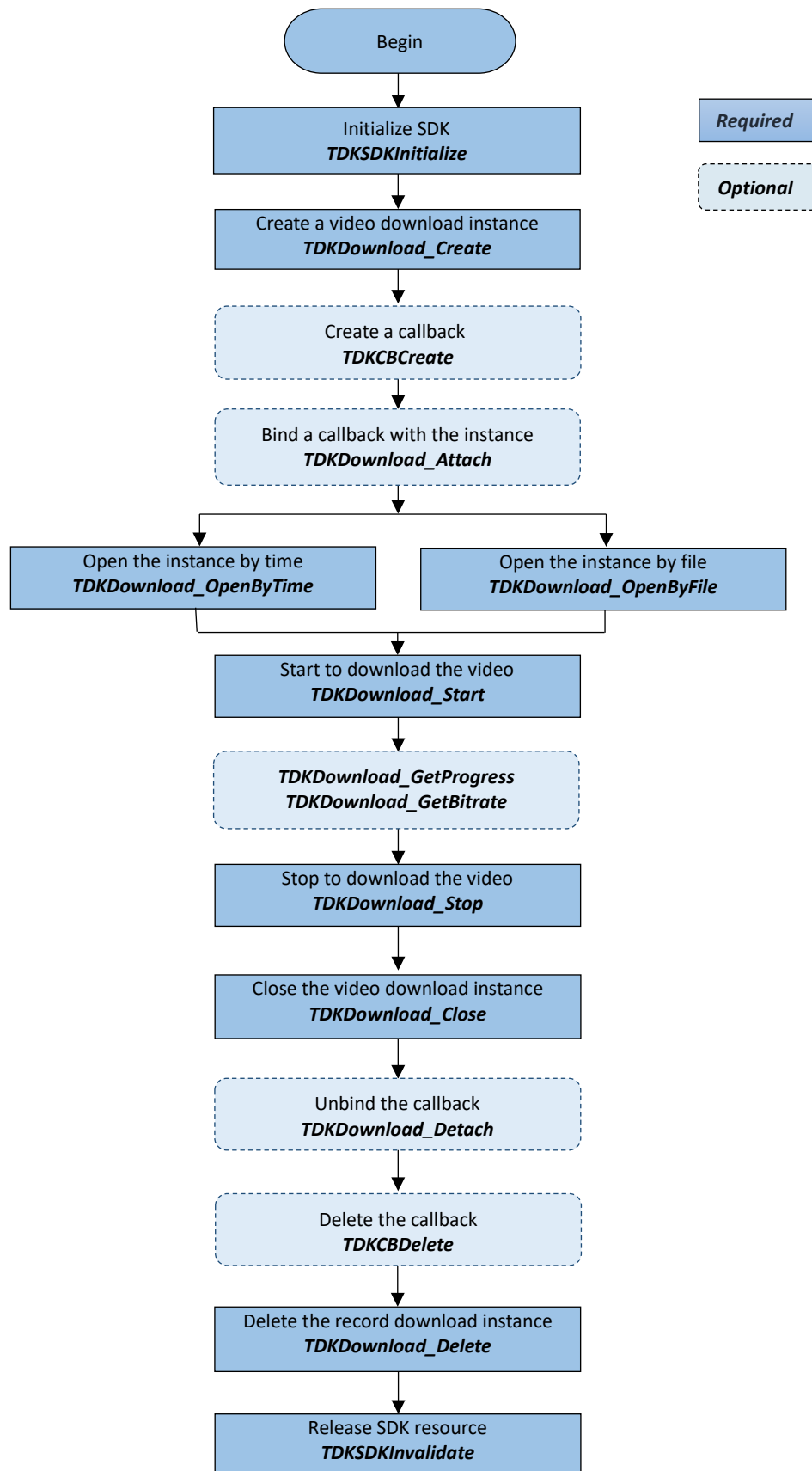
device and save into the local.

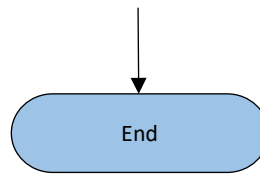
It allows you to download from the selected channels and export to the local disk or external USB flash drive.

Record download have two ways: download by file and download by time.

2.5.2 Process

Figure 2.5-1 Process of record download





Process Description

- Step 1 Call [TDKSDKInitialize](#) to initialize SDK.
- Step 2 Call [TDKDownload Create](#) to create a record download instance.
- Step 3 (optional) Call [TDKCBCreate](#) to Create a callback.
- Step 4 (optional) After a callback created, call [TDKDownload Attach](#) to bind the callback on the instance.
- Step 5 Download the record have the following two ways:
- Download by time, apply to you know the start and end times you need to download. You can call [TDKDownload OpenByTime](#) to open the record download instance.
 - Download by file name, apply to you know the record file name. You can call [TDKDownload OpenByFile](#) to open the record download instance.
- Step 6 Call [TDKDownload Start](#) to start download record files.
- Step 7 (optional) Call [TDKDownload GetProgress](#) to get the download progress, call [TDKDownload getBitrates](#) to get the download bitrate.
- Step 8 Call [TDKDownload Stop](#) to stop downloading the record files.
- Step 9 After using the function module, call [TDKDownload Close](#) to close the instance.
- Step 10 (optional) Call [TDKDownload Detach](#) to unbind the callback.
- Step 11 (optional) After detaching, call [TDKCBDelete](#) to delete the callback.
- Step 12 Call [TDKDownload Delete](#) to delete the record download instance.
- Step 13 Call [TDKSDKInvalidate](#) to release SDK resource.

2.5.3 Example Code

```

#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")
int TDKAPI callback_download(void* context, TDKU32 cmd, void* param0,
void* param1, void* param2)
{
    if (cmd == TDKDOWNLOAD_MSG_PACKET)
  
```



```

    {
        TDK_FRAMEHEAD* head = (TDK_FRAMEHEAD*)param1;
        TDKU8* data = (TDKU8*)(head + 1);
    }
    return TDK_OK;
}

void startDownloadRec()
{
    //Create downloader
    TDKHANDLE hDown = TDKDownload_Create();
    if (!hDown)
    {
        printf("Create download handle failed!\n");
        return;
    }

    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&
callback_download);
    if (hCallback)
    {
        //Register callback
        TDKDownload_Attach(hDown, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }
    //search playback file list
    string url = "tdk://192.168.13.197:34567/mode=real&idc=1&ids=1";
    TDK_SYSTEM_TIME stime; //start time
    stime.year = 2023;
    stime.month = 5;
    stime.day = 22;
    stime.hour = 18;
    stime.minute = 41;
    stime.second = 0;
    stime.isdst = 0;
    TDK_SYSTEM_TIME etime; //end time
    etime.year = 2023;
    etime.month = 5;
    etime.day = 22;
    etime.hour = 18;
    etime.minute = 45;

```

```

etime.second = 0;
etime.isdst = 0;

TDK_FILE_SEARCH_PARAM param;
param.filetype = 1; //media
param.rectype = 255; //all
param.chnmask0 = 1;
param.chnmask1 = 0;
param.streamtype = 0; //main
param.starttime = stime;
param.endtime = etime;

TDK_DOWNLOAD_PARAM_TIME dwparam;
dwparam.channel = 0; //this channel is begin from 0;
dwparam.filetype = TDK_FILE_MEDIA;
dwparam.recordtype = TDKREC_ALL;
dwparam.streamtype = TDK_STREAM_MAIN; //main
dwparam.starttime = stime;
dwparam.endtime = etime;

const char* str = "D:\\tdk\\";
strcpy_s(dwparam.destpath, str);
dwparam.format = TDK_RECORD_DAV;

//Open download player by time
int ret = TDKDownload_OpenByTime(hDown, url.c_str(), "admin",
"", &dwparam);
if (ret != TDK_OK)
{
    if (hCallback)
    {
        //Unregister callback
        TDKDownload_Detach(hDown, hCallback);

        //Destroy callback
        TDKCBDelete(hCallback);
    }

    //Destroy stream player
    TDKDownload_Delete(hDown);
    return;
}
//Start download
ret = TDKDownload_Start(hDown);

```

```

    if (ret != TDK_OK)
    {
        TDKDownload_Close(hDown);
    }
    //TODO: Call optional api.
    //
    UINT state;
    int error;
    UINT progress;
    int ret2 = TDKDownload_GetProgress(hDown, &state, &error, &progress);
    if (ret2 == TDK_OK) {
        while (progress != 100) {
            TDKDownload_GetProgress(hDown, &state, &error, &progress);
            printf("%d\n", progress);
        }
    }

    TDKDownload_Stop(hDown);
    TDKDownload_Detach(hDown, hCallback);

    //Destroy callback
    TDKCBDelete(hCallback);
    TDKDownload_Close(hDown);

}
int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startDownloadRec();

    //Release SDK resource
    TDKSDKInvalidate();
}

```

```
    return 0;  
}
```

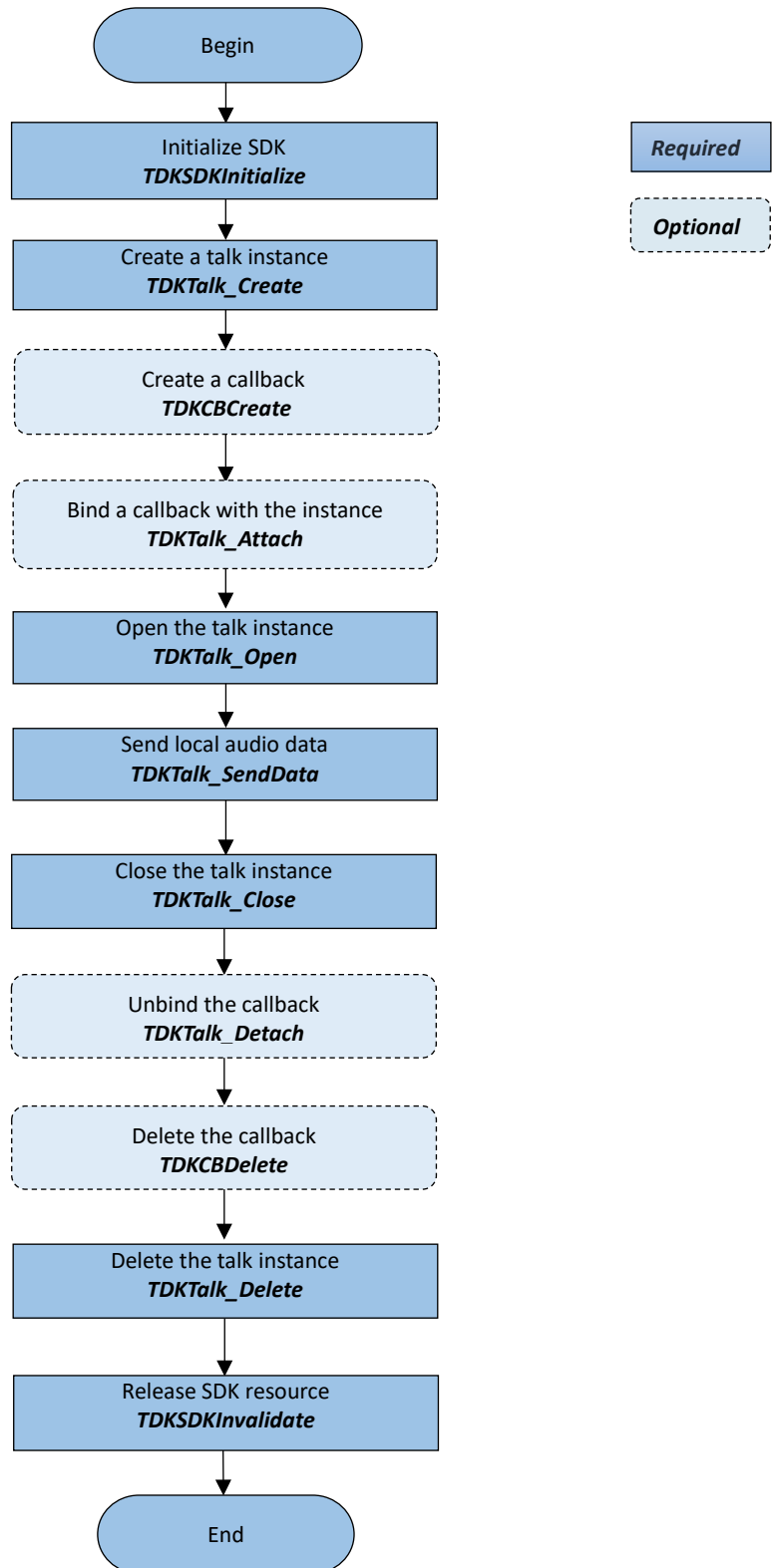
2.6 Talk Data Transparent Transmission

2.6.1 Introduction

Talk data transparent transmission realizes the talk data interaction between the local platform and the device. You can send talk data to the device, and attach a callback function to get the talk data from device.

2.6.2 Progress

Figure 2.6-1 Process of talk data transparent



Process Description

- Step 1 Call [TDKSDKInitialize](#) to initialize SDK.
- Step 2 Call [TDKTalk Create](#) to create a talk instance.
- Step 3 (optional) Call [TDKCBCreate](#) to create a callback.
- Step 4 (optional) After a callback created, call [TDKTalk Attach](#) to bind the callback to the talk instance.
- Step 5 Call [TDKTalk Open](#)(set the uri, username, password, audio coding type, audio sampling rate) to open the talk instance.
- Step 6 Call [TDKTalk SendData](#) to transmit audio data to the device.
- Step 7 After transferring the data, call [TDKTalk Close](#) to close the talk instance.
- Step 8 (optional) After closing, call [TDKTalk Detach](#) to unbind the callback.
- Step 9 (optional) After detaching, call [TDKCBDelete](#) to delete the callback.
- Step 10 Call [TDKTalk Delete](#) to delete the talk instance.
- Step 11 Call [TDKSDKInvalidate](#) to release SDK resource.

2.6.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")
int TDKAPI callback_talk(void* context, TDKU32 cmd, void* param0, void* param1, void* param2)
{
    if (cmd == TDKTALK_MSG_PACKET)
    {
        TDK_FRAMEHEAD* head = (TDK_FRAMEHEAD*)param1;
        TDKU8* data = (TDKU8*)(head + 1);
        #if 1
            printf("Frame(%04d-%02d-%02d %02d:%02d:%02d.%03u), codec = %d, type = %d, fps = %.3f, len = %d, \n",
                (%02x %02x %02x %02x-%02x %02x %02x %02x) ----
                (%02x %02x %02x %02x %02x %02x %02x %02x)\n",
                head->tms.year + 2000, head->tms.month, head->tms.day,
                head->tms.hour, head->tms.minute, head->tms.second, head->ms,
                head->codec, head->prefix[3] - TDKFRAME_BASE, (float)(head->fps
```

```

/ 4.0), head->length,
    data[0], data[1], data[2], data[3], data[4], data[5], data[6], data[7],
    data[8], data[9], data[10], data[11], data[12], data[13], data[14],
data[15]);

#endif
}
if (cmd == TDKTALK_MSG_STATE)
{
    printf("Live Stream State = %d, error = %d\n", (TDKU32)(size_t)param1,
(TDKU32)(size_t)param2);
}
return TDK_OK;
}

void startSendTalkData() {
    //Create talk handle
    TDKHANDLE hTalk = TDKTalk_Create();
    if (!hTalk)
    {
        printf("Create talk handle failed!\n");
        return;
    }

    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&callback_talk);
    if (hCallback)
    {
        //Register callback
        TDKTalk_Attach(hTalk, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }

    //open talk handle
    string url = "tdk://192.168.13.196:34567/t";
    int ret = TDKTalk_Open(hTalk, url.c_str(), "admin",
"",TDKCODEC_PCMA,8000);
    if (ret != TDK_OK) {
        goto err1;
    }
    //after open success, you can use this api to send data

```

```

//TDKTalk_SendData(hTalk, data, data.lenth);

printf("Input any key to quit.\n");
getchar();
TDKTalk_Close(hTalk);
err1:
if (hCallback)
{
    //Unregister callback
    TDKTalk_Detach(hTalk, hCallback);
    //Destroy callback
    TDKCBDelete(hCallback);
}
//Destroy talk handle
TDKTalk_Delete(hTalk);
return;
}
int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startSendTalkData();

    //Release SDK resource
    TDKSDKInvalidate();

    return 0;
}

```

2.7 Talk and Broadcast

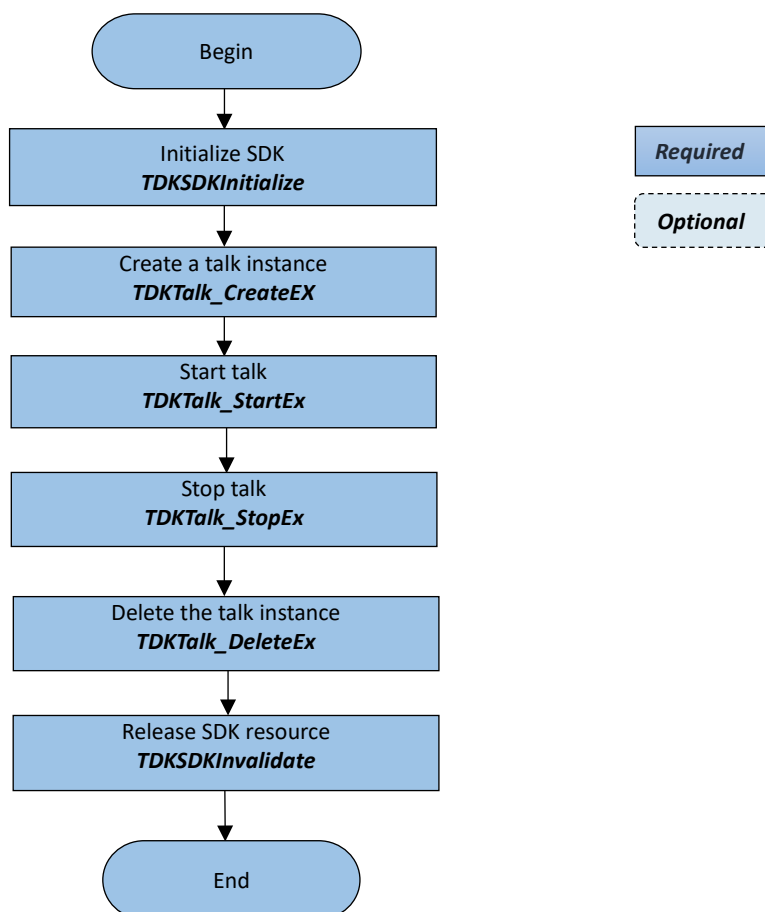
2.7.1 Introduction

Talk and broadcast realizes the voice interaction between the local platform

and the environment where front-end devices are located.
This section introduces how to use SDK to realize the voice talk with the front-end devices.

2.7.2 Progress

Figure 2.7-1 Process of talk and broadcast



Process Description

- Step 1 Call [TDKSDKInitialize](#) to initialize SDK.
- Step 2 Call [TDKTalk_CreateEx](#) to create a talk instance.
- Step 3 Call [TDKTalk_StartEx](#)(set the uri, username, password) to start talk.
- Step 4 Call [TDKTalk_StopEx](#)(set the uri, username, password) to stop talk.
- Step 5 After transferring the data, call [TDKTalk_DeleteEx](#) to delete the talk instance.
- Step 6 Call [TDKSDKInvalidate](#) to release SDK resource.

2.7.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")

void startTalkBroadcast() {
    //Create downloader
    TDKHANDLE hTalkb = TDKTalk_CreateEx();
    if (!hTalkb)
    {
        printf("Create talk handle failed!\n");
        return;
    }
    string url = "tdk://192.168.13.196:34567/talk";
    int ret = TDKTalk_StartEx(hTalkb, url.c_str(), "admin", "");
    if (ret != TDK_OK) {
        return;
    }
    printf("Input any key to quit.\n");
    getchar();
    TDKTalk_StopEx(hTalkb, url.c_str(), "admin", "");
    TDKTalk_DeleteEx(hTalkb);
}

int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startTalkBroadcast();

    //Release SDK resource
```

```
    TDKSDKInvalidate();  
  
    return 0;  
}
```

2.8 Alarm Listen

2.8.1 Introduction

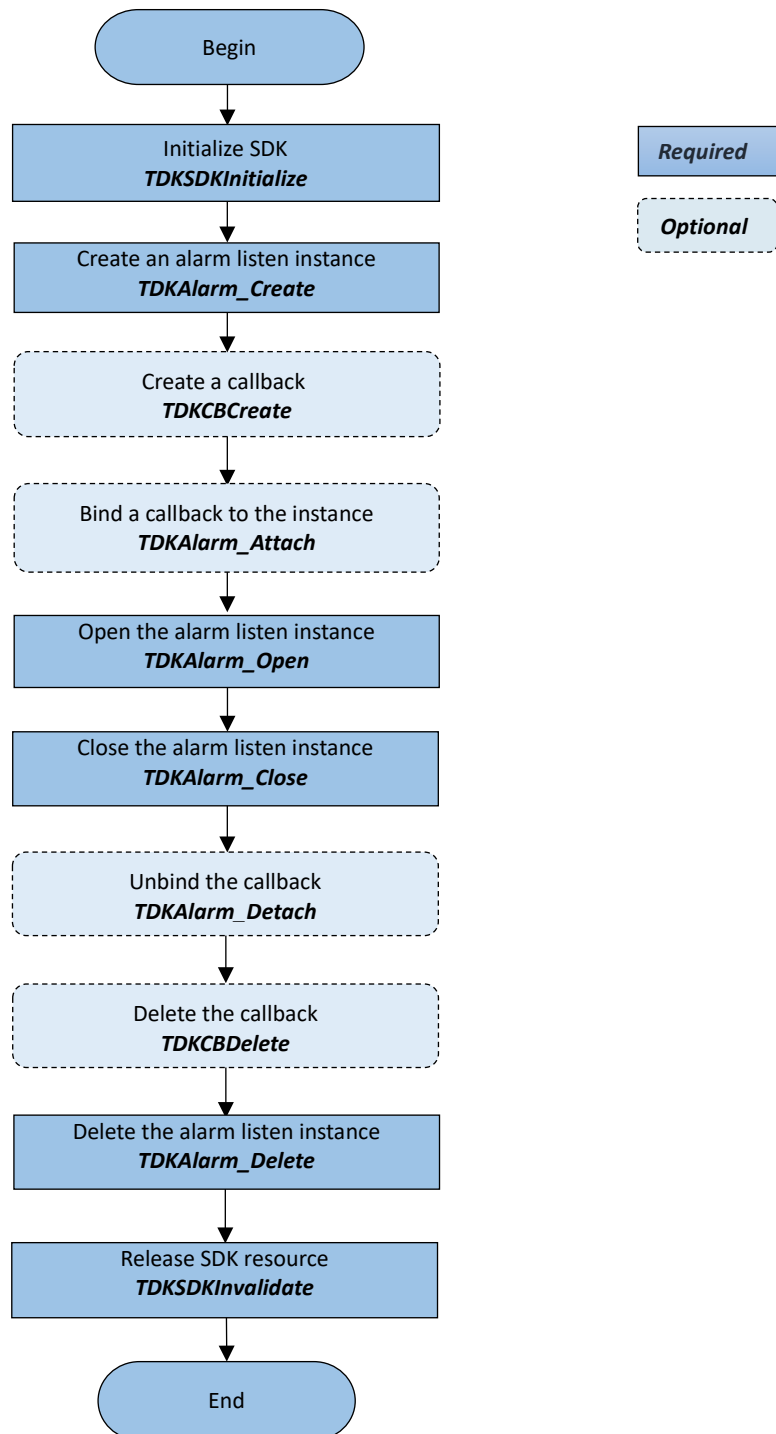
Alarm listen, is to send alarm to SDK and notify the user, when device detects special event set previously. The platform may receive video signal lost alarm, tampering alarm and motion detection alarm uploaded by device.

The method of alarm listen is that SDK actively connects device and subscribes alarm function from device.

When device detects alarm event, it will immediately send a message to SDK, then it will call the callback function. User can handle the alarm by setting the callback function.

2.8.2 Progress

Figure 2.8-1 Process of alarm listen



Process Description

Step 1 Call [TDKSDKInitialize](#) to initialize SDK.

- Step 2 Call [TDKAlarm Create](#) to create an alarm listen instance.
- Step 3 (optional) Call [TDKCBCreate](#) to Create a callback.
- Step 4 (optional) After a callback created, call [TDKAlarm Attach](#) to bind the callback to the alarm listen instance.
- Step 5 Call [TDKAlarm Open](#)(set the uri, username, password) to open the alarm listen instance.
- Step 7 Call [TDKAlarm Close](#) to close the alarm listen instance.
- Step 8 (optional) After closing, call [TDKAlarm Detach](#) to unbind the callback.
- Step 9 (optional) After detaching, call [TDKCBDelete](#) to delete the callback API.
- Step 10 Call [TDKAlarm Delete](#) to delete the alarm listen instance.
- Step 11 Call [TDKSDKInvalidate](#) to release SDK resource.

2.8.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")

int TDKAPI callback_alarm(void* context, TDKU32 cmd, void* param0, void*
param1, void* param2)
{
    if (cmd == TDKALARM_MSG_ALARM)
    {
        TDK_ALARM_INFO* alarmInfo = (TDK_ALARM_INFO*)param1;
        //printf("channel %d receive an alarm info,the type is %d\n",
alarmInfo->info->chn+1,alarmInfo->type);
        switch (alarmInfo->type)
        {
            case 2:    //motion detect
                printf("channel %d receive motion detect\n",
alarmInfo->info->chn + 1);
            case 3:    //vedio lost
                printf("channel %d receive video lost\n", alarmInfo->info->chn
+ 1);

            //case xx
            //you can get more alarm type in our document.
            default:
                break;
        }
    }
}
```

```

    }
    return TDK_OK;
}

void startAlarm()
{
    //create alarm listen handle
    TDKHANDLE hAlarm = TDKAlarm_Create();
    if (!hAlarm)
    {
        printf("Create alarm handle failed!\n");
        return;
    }
    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&
callback_alarm);
    if (hCallback)
    {
        //Register callback
        TDKTalk_Attach(hAlarm, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }
    //open alarm
    string url = "tdk://192.168.13.197:34567/alarm";
    int ret = TDKAlarm_Open(hAlarm, url.c_str(), "admin", "");
    if (ret != TDK_OK) {
        if (hCallback) {
            TDKAlarm_Detach(hAlarm, hCallback);
            TDKCBDelete(hCallback);
        }
        return;
    }
}

printf("Input any key to quit.\n");
getchar();

//close alarm
TDKAlarm_Close(hAlarm);
//Unregister callback
TDKAlarm_Detach(hAlarm, hCallback);
//Destroy callback
TDKCBDelete(hCallback);

```

```

        //Delete the alarm listen handle
        TDKAlarm_Delete(hAlarm);
    }
int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startAlarm();

    //Release SDK resource
    TDKSDKInvalidate();

    return 0;
}

```

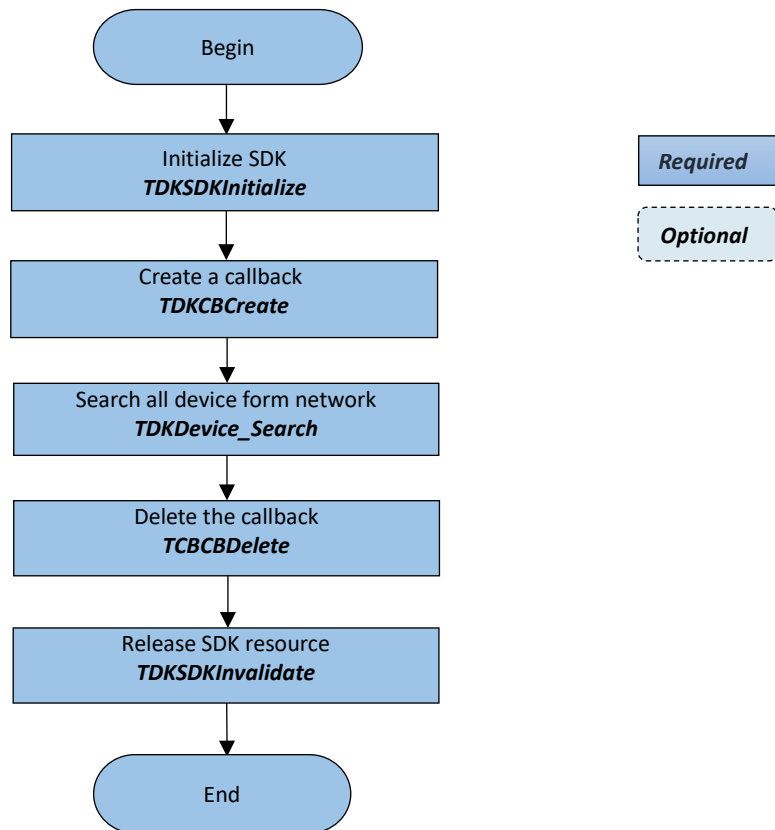
2.9 Device Search

2.9.1 Introduction

Device search is mainly used to help user to get device info in the local area network, you need to give right param, and if the device is online, it will be return in the callback.

2.9.2 Progress

Figure 2.9-1 Process of device search



Process Description

- Step 1 Call [TDKSDKInitialize](#) to initialize SDK.
- Step 2 Call [TDKCBCreate](#) to create a callback.
- Step 3 Call [TDKDevice_Search](#) (set the callback and search time) to start searching device.
- Step 4 Call [TDKCBDelete](#) to delete the callback.
- Step 5 Call [TDKSDKInvalidate](#) to release SDK resource.

2.9.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")
int TDKAPI callback_devicessearch(void* context, TDKU32 cmd, void* param0,
void* param1, void* param2)
{
```



```

    if (cmd == TDKDEVICE_MSG_DEV_SEARCH) {
        TDK_DEVICE_NET_INFO* pInfo = (TDK_DEVICE_NET_INFO*)param1;
        //todo,you can get device info in here

    }
    return TDK_OK;
}

void startSearchDevice()
{
    TDK_DEVICE_SEARCH_PARAM sparam;
    //create callback
    TDKHANDLE hCallback = TDKCBBCreate(NULL, (TDKCALLBACK)&
callback_devicessearch);
    if (hCallback)
    {
        //Register callback
        sparam.callback = hCallback;

    }
    sparam.timeout = 3000;
    //search device
    int ret = TDKDevice_Search(&sparam);

    if (ret != TDK_OK) {
        return;
    }
}

int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    startSearchDevice();

    //Release SDK resource
    TDKSDKInvalidate();
}

```

```
    return 0;  
}
```

2.10 Device Config

2.10.1 Introduction

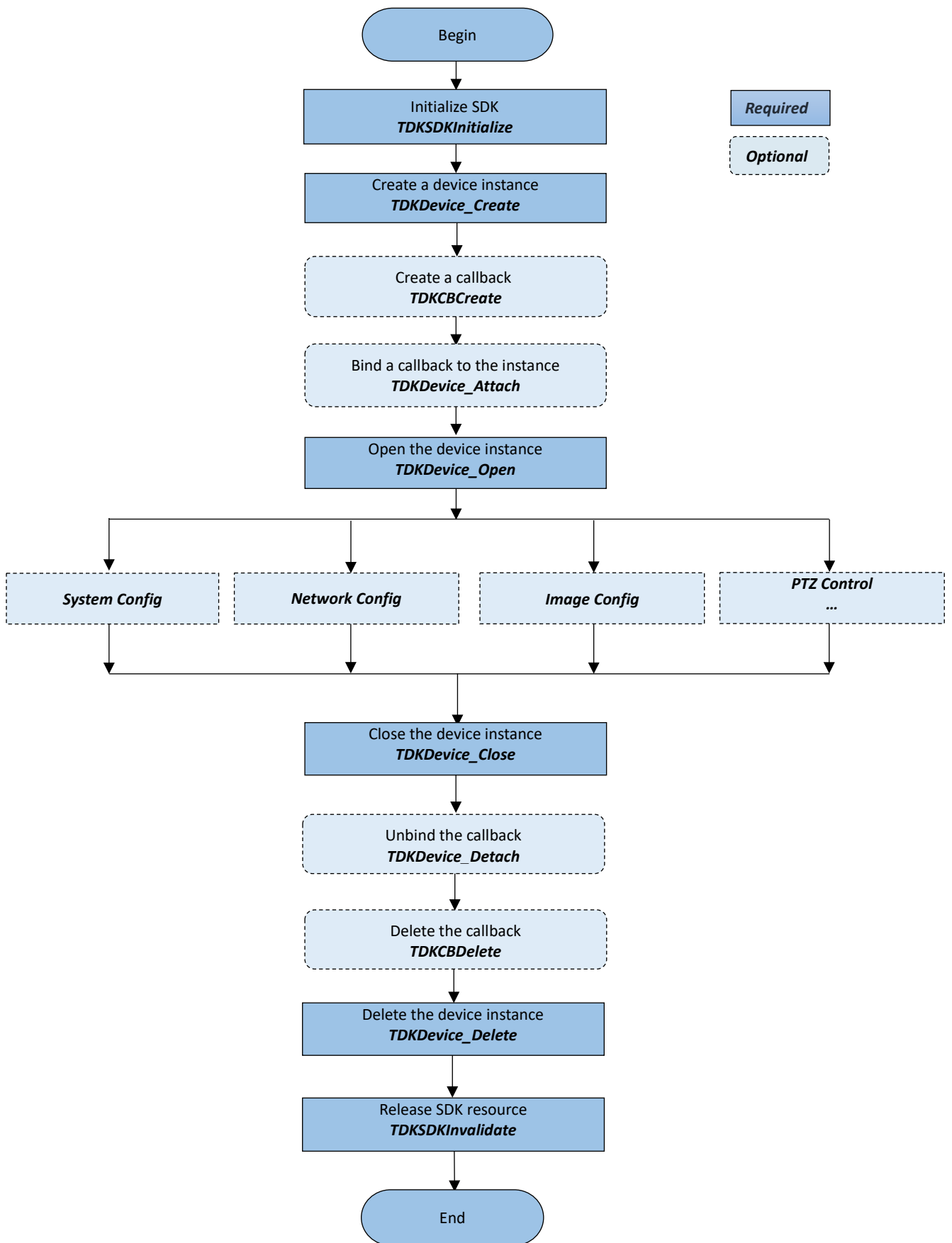
Device config is the most important part of the SDK.

Before config the device, successfully SDK initialization should be done.

You can get/set device system config, network config, image config, encode config, storage config, event config, recording config, PTZ control and so on.

2.10.2 Progress

Figure 2.10-1 Process of device config



Process Description

- Step 1 Call [TDKSDKInitialize](#) to initialize SDK.
- Step 2 Call [TDKDevice Create](#) to create a device instance.
- Step 3 (optional) Call [TDKCBCreate](#) to create a callback.
- Step 4 (optional) After a callback is created, you can call [TDKDevice Attach](#) to bind the callback to the device instance.
- Step 5 Call [TDKDevice Open](#)(set the uri, username, password) to open the device instance.
- Step 6 After open device:
- You can get/set image config
 - You can get/set system config.
 - You can get/set network config
 - ...
- Step 7 Call [TDKDevice Close](#) to close the device instance.
- Step 8 (optional) After closing, call [TDKDevice Detach](#) to unbind the callback.
- Step 9 (optional) After detaching, call [TDKCBDelete](#) to delete the callback.
- Step 10 Call [TDKDevice Delete](#) to delete the device instance.
- Step 11 Call [TDKSDKInvalidate](#) to release SDK resource.

2.10.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")
int TDKAPI callback_deviceconfig(void* context, TDKU32 cmd, void* param0,
void* param1, void* param2)
{
    if (cmd == TDKDEVICE_MSG_UPGRADE_PROGRESS) {
        int progress = (int)param1;
        if (progress == 100)
        {
            printf("upgreade success\n");
            return 0;
        }
    }

    if (cmd == TDKDEVICE_MSG_STATE) {
        printf("Live Stream State = %d, error = %d\n", (TDKU32)(size_t)param1,
(TDKU32)(size_t)param2);
    }
}
```

```

    if (cmd == TDKDEVICE_MSG_ONLINE) {
        printf("this device is online\n");
    }

    return TDK_OK;
}
//Device config
//example 1,get device infomation
void startDeviceConfig1()
{
    //create device handle
    TDKHANDLE hDev = TDKDevice_Create();
    if (!hDev)
    {
        printf("Create alarm handle failed!\n");
        return;
    }
    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&
callback_deviceconfig);
    if (hCallback)
    {
        //Register callback
        TDKDevice_Attach(hDev, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }

    //open the device
    string strurl = "tdk://192.168.13.197:34567/t";
    int ret = TDKDevice_OpenEx(hDev, strurl.c_str(), "admin", "");
    //int ret= TDKDevice_Open(hDev, "192.168.13.197/t", 34567, "admin", "");
    if (ret != TDK_OK) {
        return;
    }
    //TODO: Call optional api.
    //example 1
    //get device infomation
    TDK_DEVICE_INFO info = { 0 };
    ret = TDKDevice_GetDeviceInfo(hDev, &info);
    if (ret != TDK_OK) {

```

```

        return;
    }
    printf("Device info:the channel number is %d,the version is %s\n",info.analogchnnum + info.digitalchnnum,
        info.devversion);
    printf("-----\n");
    //example 2
    //get device time config
    TDK_TIME_CONFIG tconfig = {0};
    TDKDevice_GetTime(hDev, &tconfig);
    if (ret != TDK_OK) {
        return;
    }
    printf("Device time zone is %d,time is%d,%d,%d,%d,%d,%d\n",
tconfig.timezone,tconfig.time.year,tconfig.time.month,

tconfig.time.day,tconfig.time.hour,tconfig.time.minute,tconfig.time.second
    );
    printf("-----\n");

    //example 3
    //set device time config
    //TDK_TIME_CONFIG tconfig;
    TDK_SYSTEM_TIME tempTime;
    tempTime.year = 2023;
    tempTime.month = 5;
    tempTime.day = 5;
    tempTime.hour = 5;
    tempTime.minute = 5;
    tempTime.second = 5;
    tconfig.time = tempTime;
    tconfig.timezone = 13;
    ret = TDKDevice_SetTime(hDev,&tconfig);
    if (ret != TDK_OK) {
        return;
    }
    printf("set success!\n");
    //get device time config after change;
    TDKDevice_GetTime(hDev, &tconfig);
    if (ret != TDK_OK) {
        return;
    }
    printf("Device time zone is %d,time is%d,%d,%d,%d,%d,%d\n",
tconfig.timezone, tconfig.time.year, tconfig.time.month,

```

```

        tconfig.time.day, tconfig.time.hour, tconfig.time.minute,
tconfig.time.second
    );
    printf("-----\n");
    //example 4
    //Get device TCPIP config
    TDK_DEVICE_TCPIP cfg;
    memset(&cfg, 0, sizeof(TDK_DEVICE_TCPIP));
    ret = TDKDevice_Network_GETTCPIP(hDev, &cfg);
    if (ret != TDK_OK) {
        return;
    }
    printf("TCP infomation:HS_download_sup=%d,HS_download=%d\n"

    "transfer_mode_sup=%d,transfer_CheckBox_enable=%d,transfer_CheckBox
_checked=%d\n"
        "transfer_comboBox_enable=%d,transfer_comboBox_value=%d\n"
        "http_port=%d,http_port_sup=%d\n"

    "max_con_max=%d,max_con_val=%d,max_con_min=%d,max_con_sup=%d\
n"
        "moblie_port=%d,moblie_port_sup=%d\n"
        "tcp_port=%d,tcp_port_sup=%d\n"
        "https_port_sup=%d,https_port=%d\n"
        , cfg.tcp.HS_download_sup, cfg.tcp.HS_download,
        cfg.tcp.transfer_mode_sup, cfg.tcp.transfer_CheckBox_enable,
cfg.tcp.transfer_CheckBox_checked,
        cfg.tcp.transfer_comboBox_enable, cfg.tcp.transfer_comboBox_value,
        cfg.tcp.https_port, cfg.tcp.https_port_sup,
        cfg.tcp.max_con_max, cfg.tcp.max_con_val, cfg.tcp.max_con_min,
cfg.tcp.max_con_sup,
        cfg.tcp.moblie_port, cfg.tcp.moblie_port_sup,
        cfg.tcp.tcp_port, cfg.tcp.tcp_port_sup,
        cfg.tcp.https_port_sup, cfg.tcp.https_port);
    printf("-----\n");
    printf("RTSP information:rtsp_sup=%d,rtsp_port=%d,rtsp_url=%s\n"
        , cfg.rtsp.rtsp_sup, cfg.rtsp.rtsp_port, cfg.rtsp.rtsp_url);
    printf("-----\n");
    printf("DNS
information:major_dns_sup=%d,major_dns=%s,minor_dns_sup=%d,minor_dns
=%s\n",
        cfg.dns.major_dns_sup, cfg.dns.major_dns, cfg.dns.minor_dns_sup,
cfg.dns.minor_dns);
    printf("-----\n");

```

```

printf("network_adapter_count= %d\n", cfg.network_adapter_count);
printf("-----\n");
printf("first adapter information :ip=%s,mac_addr=%s,submask=%s\n"
      "gateway=%s,inner_ip=%s,inner_ip_sup=%d\n"
      "dhcp_sup=%d,dhcp_checked=%d,ethType=%d,ethType_sup=%d\n"
      ,cfg.adapter[0].ip, cfg.adapter[0].mac_addr, cfg.adapter[0].submask,
      cfg.adapter[0].gateway, cfg.adapter[0].inner_ip,
cfg.adapter[0].inner_ip_sup,
      cfg.adapter[0].dhcp_sup, cfg.adapter[0].dhcp_checked,
cfg.adapter[0].ethType, cfg.adapter[0].ethType_sup);
printf("-----\n");
printf("network_adapter_default=%d,pcfg=%d\n",
cfg.network_adapter_default,(int)cfg.pcfg);
printf("-----\n");

printf("Input any key to quit.\n");
getchar();

//close device handle
TDKDevice_Close(hDev);

if (hCallback)
{
    //Unregister callback
    TDKDevice_Detach(hDev, hCallback);

    //Destroy callback
    TDKCBDelete(hCallback);
}

TDKDevice_Delete(hDev);
}
//example 2,get online device
void startDeviceConfig2() {

    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&
callback_deviceconfig);
    if (!hCallback)
    {
        printf("Create callback function failed!\n");
    }
    //create online query handle
    TDKHANDLE hOnline = TDKDeviceOnline_Create(hCallback);

```



```

//query online device
int ret = TDKDeviceOnline_Query(hOnline, "tdk://192.168.13.196");

if (ret != TDK_OK) {
    TDKCBDelete(hCallback);
    return;
}
Sleep(15000);
printf("Querying,please wait 15s,Input any key to quit.\n");
getchar();
//Destroy callback
TDKCBDelete(hCallback);
//Delete online query handle
TDKDeviceOnline_Delete(hOnline);
}
//example 3,upgrade device
void startDeviceConfig3() {
    //create device handle
    TDKHANDLE hDev = TDKDevice_Create();
    if (!hDev)
    {
        printf("Create alarm handle failed!\n");
        return;
    }
    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&
callback_deviceconfig);
    if (hCallback)
    {
        //Register callback
        TDKDevice_Attach(hDev, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }
}

//open the device
string strurl = "tdk://192.168.13.197:34567/t";
int ret = TDKDevice_OpenEx(hDev, strurl.c_str(), "admin", "");
//int ret= TDKDevice_Open(hDev, "192.168.13.197/t", 34567, "admin", "");
if (ret != TDK_OK) {
    return;
}

```

```

//TODO: Call optional api.
string file = "D:\\QV-IPC-7N816_ZX_V2.10.6.R11413.20220907.upf";
ret = TDKDevice_Upgrade(hDev, file.c_str(), callback_deviceconfig);
if (ret != TDK_OK) {
    return;
}
printf("Upgrading, please wait\n");
printf("Input any key to quit.\n");

getchar();

//close device handle
TDKDevice_Close(hDev);

if (hCallback)
{
    //Unregister callback
    TDKDevice_Detach(hDev, hCallback);

    //Destroy callback
    TDKCBDelete(hCallback);
}
TDKDevice_Delete(hDev);
}

int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startDeviceConfig1();
    //startDeviceConfig2();
    //startDeviceConfig3();

    //Release SDK resource
    TDKSDKInvalidate();
}

```

```
    return 0;  
}
```

2.11 Smart Stream

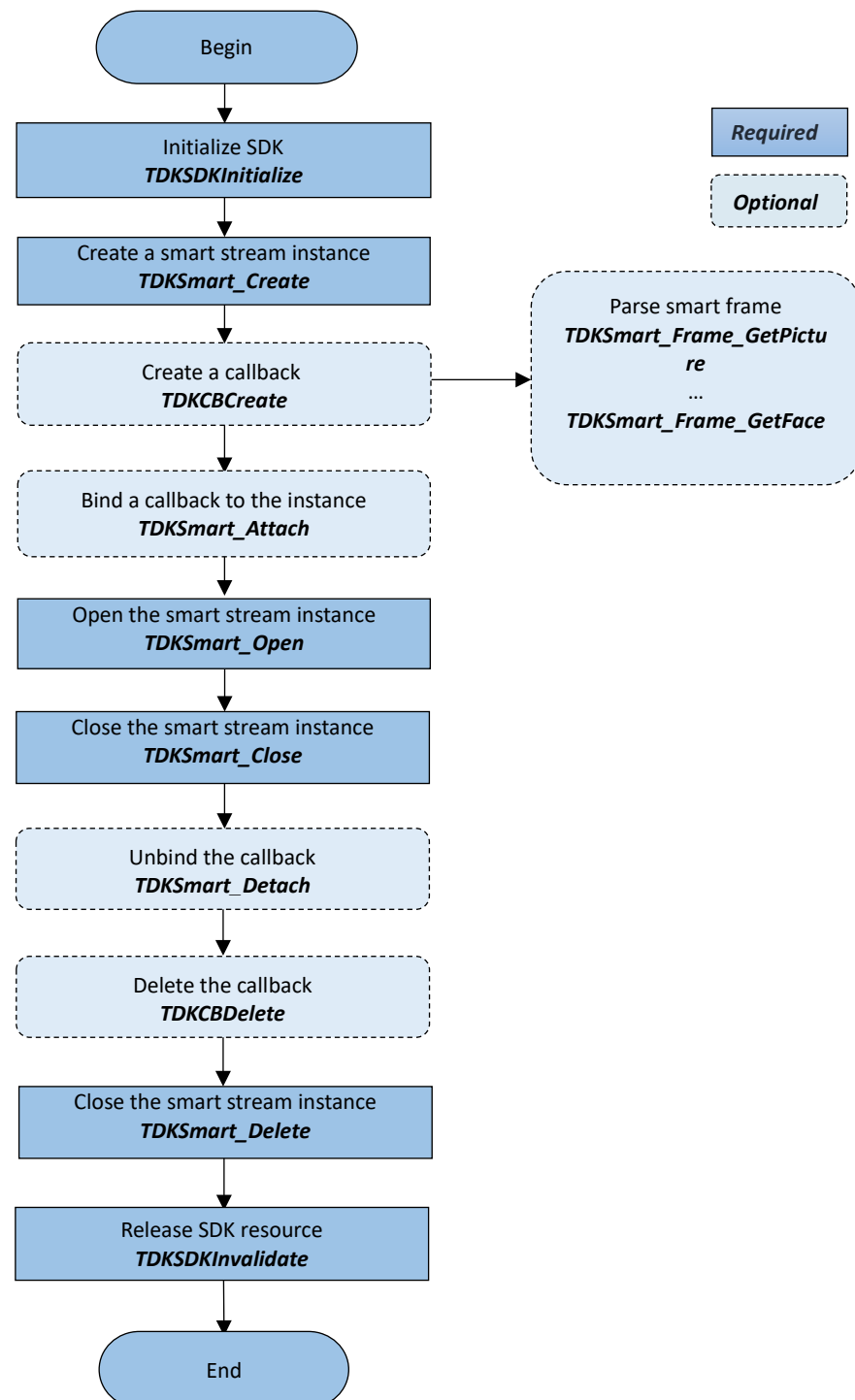
2.11.1 Introduction

Devices make smart analysis by real-time stream, such as face recognition, license plate recognition.

You can call the smart frame analysis api in the callback to get the face picture, face information, license plate picture and license plate.

2.11.2 Progress

Table 2-1 Process of smart stream



Process Description

Step 1 Call [TDKSDKInitialize](#) to initialize SDK.

- Step 2 Call [TDKSmart Create](#) to create a smart stream instance.
- Step 3 (optional) Call [TDKCBCreate](#) to create a callback, in this callback function, you can use Intelligent frame parse interface to parse the frame to get information in the frame(such as face picture, face information, license plate picture, license plate).
- Step 4 (optional) After a callback is created, call [TDKSmart Attach](#) to bind the callback to the smart stream instance.
- Step 5 Call [TDKSmart Open](#)(set the address, port, username, password) to open the smart stream instance.
- Step 6 Call [TDKSmart Close](#) to close the smart stream instance.
- Step 7 (optional) After closing, call [TDKSmart Detach](#) to unbind the callback.
- Step 8 (optional) After detaching, call [TDKCBDelete](#) to delete the callback.
- Step 9 Call [TDKSmart Delete](#) to delete the smart stream instance.
- Step 10 Call [TDKSDKInvalidate](#) to release SDK resource.

2.11.3 Example Code

```
#include <Windows.h>
#include <iostream>
#include "TDKSDK.h"
using namespace std;

#pragma comment(lib, "TDKSDK.lib")
int TDKAPI callback_smart(void* context, TDKU32 cmd, void* param0, void* param1, void* param2)
{
    if (cmd == TDKSMART_MSG_STATE) {
        printf("Live Stream State = %d, error = %d\n", (TDKU32)(size_t)param1, (TDKU32)(size_t)param2);
    }
    if (cmd == TDKSMART_MSG_FRAME) {
        TDKHANDLE hFrame = (TDKHANDLE*)param1;

        int facenum = TDKSmart_Frame_GetFaceNum(hFrame);
        printf("face num = %d\n", facenum);
        TDK_SMART_PIC pic = { 0 };
        TDKSmart_Frame_GetPicture(hFrame, &pic);
        printf("picture width = %d,height = %d,codec=%d,picture name= %s", pic.w, pic.h, pic.codec, pic.name);
    }

    return TDK_OK;
}
```

```

}
void startSmart() {
    //create smart handle
    TDKHANDLE hSmart = TDKSmart_Create();
    if (!hSmart)
    {
        printf("Create Smart handle failed!\n");
        return;
    }
    //Create callback function
    TDKHANDLE hCallback = TDKCBCreate(NULL, (TDKCALLBACK)&
callback_smart);
    if (hCallback)
    {
        //Register callback
        TDKSmart_Attach(hSmart, hCallback);
    }
    else
    {
        printf("Create callback function failed!\n");
    }
    //Open smart
    int ret = TDKSmart_Open(hSmart,
"192.168.13.152",34567,"admin","tdk123456");
    if (ret != TDK_OK) {
        return;
    }

    printf("Input any key to quit.\n");
    getchar();

    //close smart handle
    TDKSmart_Close(hSmart);
    if (hCallback)
    {
        //Unregister callback
        TDKSmart_Detach(hSmart, hCallback);

        //Destroy callback
        TDKCBDelete(hCallback);
    }
    //Delete the smart handle
    TDKSmart_Delete(hSmart);
}

```

```

int main()
{
    //SDK initialize
    int ret = TDKSDKInitialize();
    if (ret != TDK_OK)
    {
        printf("Initialize SDK failed!\n");
        return 0;
    }

    //Function module implementation
    startSmart();

    //Release SDK resource
    TDKSDKInvalidate();

    return 0;
}

```

3 API Reference

3.1 SDK Initialization

3.1.1 SDK Initialization

Table 3-1 TDKSDKInitialize

Item	Description
Name	SDK initialization interface. Call it when initializing program.
Pre-condition	None.
Function	int TDKAPI TDKSDKInitialize(void); int TDKAPI TDKSDKInitializeEx(IN TDK_GLOBAL_PARAM* param);
Parameter	[in] param Global init param, can be NULL
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .
Note	By default, all video and audio streams are received in a single thread. If you want to receive streams in a thread pool, you can use the TDKSDKInitializeEx interface and specify the number of threads in the thread pool with a valid param. If the incoming param is NULL, the effect is the same with the TDKSDKInitialize interface. [warning] After using this API, you must to call TDKSDKInvalidate to release the SDK in the end.

3.1.2 SDK Cleanup

Table 3-2 TDKSDKInvalidate

Item	Description
Name	SDK cleanup interface. Call it when program end.
Pre-condition	None.
Function	int TDKAPI TDKSDKInvalidate(void);
Parameter	None.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .
Note	Used in pairs with TDKSDKInitialize.

3.2 Real-time Monitoring

3.2.1 Create Stream Player

Table 3-3 TDKPlayer_Create

Item	Description	
Name	Create stream player.	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKPlayer_Create(IN TDKHANDLE hWnd, IN TDKS32 type);	
Parameter	[in] hWnd	The handle of a display view that may be the NSOpenGLView or NSOpenGLLayer in MACOS.
	[in] type	Reserved parameter, default value is 0.
Return value	Success: Non-zero. Failure: 0.	
Note	[warning] Must to call TDKPlayer_Delete in the end.	

3.2.2 Delete Stream Player

Table 3-4 TDKPlayer_Delete

Item	Description	
Name	Delete stream player.	
Pre-condition	Get a stream player handle by calling TDKPlayer_Create.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Delete(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Create.	

3.2.3 Attach Callback to the Player

Table 3-5 TDKPlayer_Attach

Item	Description	
Name	Attach callback to the player.	
Pre-condition	Make sure the stream player and callback have been created.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Attach(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	After using, call TDKPlayer_Detach to detach it.	

3.2.4 Detach Callback from the Player

Table 3-6 TDKPlayer_Detach

Item	Description	
Name	Detach callback from the player.	
Pre-condition	Make sure the stream player has attached the callback.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Detach(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hcallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Attach.	

3.2.5 Attach the Decoded Data Callback Function

Table 3-7 TDKPlay_AttachDecode

Item	Description	
Name	Attach the decode data callback to the player.	
Pre-condition	Make sure the stream player and callback have been created.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_AttachDecode(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning]After using, call TDKPlayer_DetachDecode to detach it.	

3.2.6 Detach the Decoded Data Callback Function

Table 3-8 TDKPlayer_DetachDecode

Item	Description	
Name	Detach the decode data callback from the player.	
Pre-condition	Make sure the stream player has attached the decode data callback.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_DetachDecode(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_AttachDecode.	

3.2.7 Open Real-time Stream Player

Table 3-9 TDKPlayer_Open

Item	Description	
Name	Open real-time stream player.	
Pre-condition	The stream player must be created.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Open(IN TDKHANDLE hPlayer, IN const char* uri, IN const char* username, IN const char* password);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] uri	Stream request uri .
	[in] username	Username for logging in to the device.
	[in] password	Password for logging in to the device.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, call TDKPlayer_Close to close the player in the end.	

3.2.8 Close the Stream Player

Table 3-10 TDKPlayer_Close

Item	Description	
Name	Close the real-time/playback stream player.	
Pre-condition	The stream player must be opened.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Close(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Open/TDKPlayer_OpenFile.	

3.2.9 Start to Request the Stream

Table 3-11 TDKPlayer_Start

Item	Description	
Name	Start to request the stream.	
Pre-condition	The stream player must be created and opened.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Start(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	After using, call TDKPlay_Stop to stop the stream.	

3.2.10 Stop Requesting the Stream

Table 3-12 TDKPlayer_Stop

Item	Description	
Name	Stop requesting the stream.	
Pre-condition	The stream player must be started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Stop(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Start.	

3.2.11 Audio Control of the Player

Table 3-13 TDKPlayer_Listen

Item	Description	
Name	Audio control of the player	
Pre-condition	The stream player must be started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Listen(IN TDKHANDLE hPlayer, IN TDKBOOL enable);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] enable	1-open audio play, 0-stop audio play.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.12 Switch the Real-time Stream Type

Table 3-14 TDKPlayer_SwitchStream

Item	Description	
Name	Switch the real-time stream type.	
Pre-condition	The stream player must be started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SwitchStream(IN TDKHANDLE hPlayer, IN TDKS32 type);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] type	0-Main Stream, 1-Sub Stream, 2-Third Stream
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.13 Show Smart Information

Table 3-15 TDKPlayer_ShowSmart

Item	Description	
Name	Display the specified smart information on play picture.	
Pre-condition	The stream player must be started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_ShowSmart(IN TDKHANDLE hPlayer, IN TDKS32 typemask);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] typemask	0-close, 1-open.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.14 Video Snapshot

Table 3-16 TDKPlayer_Snapshot

Item	Description	
Name	Capture the picture data of the video.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Snapshot(IN TDKHANDLE hPlayer, IN const char* picname, IN TDK_IMAGE_FORMAT format);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] picname	The path to save the picture, it can include the name of the picture, if not, SDK will generate the picture name by the rule.
	[in] format	The format of the picture, reference the TDK_IMAGE_FORMAT
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.15 Start to Record

Table 3-17 TDKPlayer_StartRecord

Item	Description	
Name	Start to record.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_StartRecord(IN TDKHANDLE hPlayer, IN const char* filename, IN const char* picname, IN TDK_IMAGE_FORMAT picfmt);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] filename	Record file save path and name.
	[in] picname	Preview picture save path and name
	[in] format	The format of the preview picture, reference the TDK_IMAGE_FORMAT .
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_StopRecord.	

3.2.16 Stop Recording

Table 3-18 TDKPlayer_StopRecord

Item	Description	
Name	Stop to record.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_StopRecord(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_StartRecord.	

3.2.17 Set Zoom Operation Window

Table 3-19 TDKPlayer_SetZoomWindow

Item	Description	
Name	Set digital zoom operation window.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetZoomWindow(IN TDKHANDLE hPlayer, IN TDKHANDLE hWnd);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hWnd	The handle of the operation window.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.18 Set Zoom Region

Table 3-20 TDKPlayer_SetPlayerZoom

Item	Description	
Name	Set zoom rectangular region.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetPlayerZoom(IN TDKHANDLE hPlayer, IN RECT* rectzoom, IN RECT* controlrect);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] rectzoom	The selected zoom region in operation window.
	[in] controlrect	The whole rectangular region of operation window.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.19 Set Zoom Region Being Selected

Table 3-21 TDKPlayerSetZoomRECT

Item	Description	
Name	Set the selecting zoom region to display.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetPlayerSetZoomRECT(IN TDKHANDLE hPlayer, IN RECT* rectzoom);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] rectzoom	The zoom region being selected.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.20 Get the Count of Smart Information

Table 3-22 TDKSmart_Frame_GetSmartInfo

Item	Description	
Name	Get the count of smart info.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Frame_GetSmartCount(IN TDKHANDLE hFrame, IN TDKU32 type);	
Parameter	[in] hFrame	The handle of the smart frame instance, reference TDKSMART_MSG_FRAME in the TDKCALLBACK .
	[in] type	Smart info type (SMART_FRAME_INFO_TYPE).
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.2.21 Get the Smart Information

Table 3-23 TDKSmart_Frame_GetSmartInfo

Item	Description	
Name	Get the smart information.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI const TDK SMART OBJ * TDKAPI TDKSmart_Frame_GetSmartInfo(IN TDKHANDLE hFrame, IN TDKU32 type, IN TDKU32 index);	
Parameter	[in] hFrame	The handle of the smart frame instance, reference TDKSMART_MSG_FRAME in the TDKCALLBACK .
	[in] type	Smart information type (SMART_FRAME_INFO_TYPE).
	[in] index	Index of the smart information, start from 0.
Return value	Success: The pointer of smart object, non-zero. Failure: NULL.	
Note		

3.3 Record Search

3.3.1 Search All Record File

Table 3-24 TDKMedia_Search

Item	Description	
Name	Search all record files at once.	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI int TDKAPI TDKMedia_Search(IN const char* uri, IN const char* username, IN const char* password, IN TDK_FILE_SEARCH_PARAM * param, IN TDKU32 maxnum, INOUT TDK_FILE_INFO * result, INOUT TDKU32* count);	
Parameter	[in] uri	Record search uri.
	[in] username	Username.
	[in] password	Password.
	[in] param	The parameter of file search.
	[in] maxnum	The max number of search.
	[inout] result	Result of files searched.
	[inout] count	Number of files searched.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.3.2 Create Record Search Instance

Table 3-25 TDKMediaSearchCreate

Item	Description	
Name	Create a record search instance for search by page.	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI HANDLE TDKAPI TDKMediaSearch_Create(IN const char* uri, IN const char* username, IN const char* password, IN TDK_FILE_SEARCH_PARAM * param);	
Parameter	[in] uri	Record search uri.
	[in] username	Username
	[in] password	Password
	[in] param	The parameter of file search.
Return value	Success: The handle of the search file instance.	
Note	[warning] After using, must call TDKMediaDelete to delete the instance.	

3.3.3 Delete the Record Search Instance

Table 3-26 TDKMediaDelete

Item	Description	
Name	Delete the record search instance.	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKMediaSearch_Delete(IN HANDLE h);	
Parameter	[in] handle	The handle of the search file instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKMediaSearchCreate.	

3.3.4 Search the Next Set of Files

Table 3-27 TDKMediaSearchNext

Item	Description	
Name	Search for the next page record file.	
Pre-condition	The Search instance must be created.	
Function	TDKSDKAPI int TDKAPI TDKAPI TDKMediaSearch_Next(IN HANDLE h, IN TDKU32 maxnum, OUT TDK_FILE_INFO * result, OUT TDKU32* count);	
Parameter	[in] h	The handle of the search file instance.
	[in] maxnum	The max number of files search.
	[out] result	The result of files search.
	[out] count	Number of files searched.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4 Record Playback

3.4.1 Create Stream Player

Table 3-28 TDKPlayer_Create

Item	Description	
Name	Create stream player.	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKPlayer_Create(IN TDKHANDLE hWnd, IN TDKS32 type);	
Parameter	[in] hWnd	The handle of a display view that may be the NSOpenGLView or NSOpenGLLayer in MACOS.
	[in] type	Reserved parameter, default value is 0.
Return value	Success: Non-zero. Failure: 0.	
Note	[warning] Must to call TDKPlayer_Delete in the end.	

3.4.2 Delete Stream Player

Table 3-29 TDKPlayer_Delete

Item	Description	
Name	Delete stream player.	
Pre-condition	Got a stream player handle by calling TDKPlayer_Create.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Delete(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Create.	

3.4.3 Attach Callback to the Player

Table 3-30 TDKPlayer_Attach

Item	Description	
Name	Attach callback to the player.	
Pre-condition	Make sure the stream player and callback have been created.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Attach(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	After using, call TDKPlayer_Detach to detach it.	

3.4.4 Detach Callback from the Player

Table 3-31 TDKPlayer_Detach

Item	Description	
Name	Detach callback from the player.	
Pre-condition	Make sure the stream player has attached the callback.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Detach(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hcallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Attach.	

3.4.5 Attach the Decoded Data Callback Function

Table 3-32 TDKPlayer_AttachDecode

Item	Description	
Name	Attach the decode data callback to the player.	
Pre-condition	Make sure the stream player and callback have been created.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_AttachDecode(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning]After using, call TDKPlayer_DetachDecode to detach it.	

3.4.6 Detach the Decoded Data Callback Function

Table 3-33 TDKPlayer_DetachDecode

Item	Description	
Name	Detach the decode data callback from the player.	
Pre-condition	Make sure the stream player has attached the decode data callback.	
Function	<pre>TDKSDKAPI int TDKAPI TDKPlayer_DetachDecode(IN TDKHANDLE hPlayer, IN TDKHANDLE hCallback);</pre>	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_AttachDecode.	

3.4.7 Open Playback Stream Player by File

Table 3-34 TDKPlayer_OpenFile

Item	Description	
Name	Open playback stream player by file.	
Pre-condition	The playback stream player must be created.	
Function	<pre>TDKSDKAPI int TDKAPI TDKPlayer_OpenFile(IN TDKHANDLE hPlayer, IN const char* uri, IN const char* username, IN const char* password, IN TDK_FILE_INFO* list, IN TDKU32 count);</pre>	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] uri	Stream request uri .
	[in] username	Username
	[in] password	Password
	[in] list	The list of file.
	[in] count	The count of file.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, must to call TDKPlayer_Close in the end.	

3.4.8 Close the Stream Player

Table 3-35 TDKPlayer_Close

Item	Description	
Name	Close the playback stream player.	
Pre-condition	The stream player must be opened.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Close(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_OpenFile.	

3.4.9 Start to Request the Stream

Table 3-36 TDKPlayer_Start

Item	Description	
Name	Start to request the stream.	
Pre-condition	The stream player must be created and opened.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Start(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	After using, call TDKPlay_Stop to stop the stream.	

3.4.10 Stop Requesting the Stream

Table 3-37 TDKPlayer_Stop

Item	Description	
Name	Stop requesting the stream.	
Pre-condition	The stream player must be started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Stop(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Start.	

3.4.11 Audio Control of the Player

Table 3-38 TDKPlayer_Listen

Item	Description	
Name	Audio control of the player	
Pre-condition	The stream player must be started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Listen(IN TDKHANDLE hPlayer, IN TDKBOOL enable);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] enable	1-open audio play, 0-stop audio play.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.12 Pause Request the Stream

Table 3-39 TDKPlayer_Pause

Item	Description	
Name	Pause the stream.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Pause(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.13 Resume Request the Stream

Table 3-40 TDKPlayer_Resume

Item	Description	
Name	Resume request the stream.	
Pre-condition	Make sure the player is started and the stream is paused.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Resume(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKPlayer_Pause.	

3.4.14 Seek to the Specified Time

Table 3-41 TDKPlayer_SeekTime

Item	Description	
Name	Seek to the specified time.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SeekTime(IN TDKHANDLE hPlayer, IN TDK_SYSTEM_TIME * time);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] time	The time to be located.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.15 Seek to the Specified File

Table 3-42 TDKPlayer_SeekFile

Item	Description	
Name	Seek to the specified file.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SeekFile(IN TDKHANDLE hPlayer, IN TDK_FILE_INFO * file);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] file	The file to be located.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.16 Play the Next File

Table 3-43 TDKPlayer_SeekNextFile

Item	Description	
Name	Play the next file.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SeekNextFile(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.17 Play the Previous File

Table 3-44 TDKPlayer_SeekPrevFile

Item	Description	
Name	Play the previous file.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SeekPrevFile(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.18 Set the Speed of Playback

Table 3-45 TDKPlayer_SetSpeed

Item	Description	
Name	Set the speed of playback.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetSpeed(IN TDKHANDLE hPlayer, IN TDKS32 speed);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] speed	The speed of the playback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	<pre> #define TDKPLAYER_SPEED_SLOW8 -8 #define TDKPLAYER_SPEED_SLOW4 -4 #define TDKPLAYER_SPEED_SLOW2 -2 #define TDKPLAYER_SPEED_NORMAL 1 #define TDKPLAYER_SPEED_FAST2 2 #define TDKPLAYER_SPEED_FAST4 4 #define TDKPLAYER_SPEED_FAST8 8 </pre>	

3.4.19 Single Frame Playback

Table 3-46 TDKPlayer_Step

Item	Description	
Name	Single frame playback	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Step(IN TDKHANDLE hPlayer);	
Parameter	[in] hPlayer	The handle of the stream player.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.20 Set the Transmission Bitrate of the Stream

Table 3-47 TDKPlayer_SetBitrate

Item	Description	
Name	Set the transmission bitrate of the stream	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetBitrate(IN TDKHANDLE hPlayer, IN TDKU64 bitrate);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] bitrate	Bandwidth value, expressed in bps.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.21 Get the Time of Playback

Table 3-48 TDKPlayer_GetPlaybackTime

Item	Description	
Name	Get current the time of playback.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_GetPlaybackTime(IN TDKHANDLE hPlayer, INOUT TDK_SYSTEM_TIME * time);	
Parameter	[in] hPlayer	The handle of the stream player.
	[inout] time	The current playback time.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.22 Video Snapshot

Table 3-49 TDKPlayer_Snapshot

Item	Description	
Name	Capture the picture data of the video.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_Snapshot(IN TDKHANDLE hPlayer, IN const char* picname, IN TDK_IMAGE_FORMAT format);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] picname	The path to save the picture, it can include the name of the picture, if not, SDK will generate the picture name by the rule.
	[in] format	The format of the picture, reference the TDK_IMAGE_FORMAT
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.23 Set the Playback Mode

Table 3-50 TDKPlayer_SetPlayerMode

Item	Description	
Name	Set the playback mode, forward or backward.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetPlayerMode(IN TDKHANDLE hPlayer, IN int mode);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] mode	1-forward. -1-backward.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.24 Synchronous Playback

Table 3-51 TDKPlayer_TaskSync

Item	Description	
Name	Perform synchronous playback.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_TaskSync(IN TDKHANDLE hPlayer, IN int enable);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] enable	1- Synchronous 0- Asynchronous
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.25 Set Zoom Operation Window

Table 3-52 TDKPlayer_SetZoomWindow

Item	Description	
Name	Set digital zoom operation window.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetZoomWindow(IN TDKHANDLE hPlayer, IN TDKHANDLE hWnd);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] hWnd	The handle of the operation window.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.26 Set Zoom Region

Table 3-53 TDKPlayer_SetPlayerZoom

Item	Description	
Name	Set zoom rectangular region.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetPlayerZoom(IN TDKHANDLE hPlayer, IN RECT* rectzoom, IN RECT* controlrect);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] rectzoom	The selected zoom region in operation window.
	[in] controlrect	The whole rectangular region of operation window.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.4.27 Show Zoom Region Being Selected

Table 3-54 TDKPlayer_SetPlayerSetZoomRECT

Item	Description	
Name	Set the selecting zoom region to display.	
Pre-condition	Make sure the player is started.	
Function	TDKSDKAPI int TDKAPI TDKPlayer_SetPlayerSetZoomRECT(IN TDKHANDLE hPlayer, IN RECT* rectzoom);	
Parameter	[in] hPlayer	The handle of the stream player.
	[in] rectzoom	The zoom region being selected.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.5 Record Download

3.5.1 Create Download Instance

Table 3-55 TDKDownload_Create

Item	Description
Name	Create a record download instance.
Pre-condition	Make sure the SDK is initialized.
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKDownload_Create(void);
Parameter	None
Return value	Success: Non-zero. Failure: 0.
Note	[warning] Must to call TDKDownload_Delete in the end.

3.5.2 Delete the Download Instance

Table 3-56 TDKDownload_Delete

Item	Description
Name	Delete the record download instance.
Pre-condition	Make sure the instance is created.
Function	TDKSDKAPI int TDKAPI TDKDownload_Delete(IN TDKHANDLE hDownload);
Parameter	[in] hDownload The handle of the instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .
Note	Used in pairs with TDKDownload_Create.

3.5.3 Attach Callback Function

Table 3-57 TDKDownload_Attach

Item	Description	
Name	Attach a callback to the record download instance	
Pre-condition	Make sure the instance and the callback are created.	
Function	TDKSDKAPI int TDKAPI TDKDownload_Attach(IN TDKHANDLE hDownload, IN TDKHANDLE hCallback);	
Parameter	[in] hDownload	The handle of the instance.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, must to call TDKDownload_Detach to detach it in the end.	

3.5.4 Detach Callback Function

Table 3-58 TDKDownload_Detach

Item	Description	
Name	Detach a callback from the record download instance.	
Pre-condition	Make sure the instance attached with the callback.	
Function	TDKSDKAPI int TDKAPI TDKDownload_Detach(IN TDKHANDLE hDownload, IN TDKHANDLE hCallback);	
Parameter	[in] hDownload	The handle of the instance.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKDownload_Attach.	

3.5.5 Open the Download Instance by Time

Table 3-59 TDKDownload_OpenByTime

Item	Description	
Name	Open the video download instance by time.	
Pre-condition	Make sure the instance attached with the callback.	
Function	TDKSDKAPI int TDKAPI TDKDownload_OpenByTime(IN TDKHANDLE hDownload, IN const char* url, IN const char* username, IN const char* password, IN const TDK_DOWNLOAD_PARAM_TIME * param);	
Parameter	[in] hDownload	The handle of the instance.
	[in] url	Download request url.
	[in] username	Username
	[in] password	Password
	[in] param	The time parameter.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, call TDKDownload_Close to close the instance.	

3.5.6 Open the Download Instance by File

Table 3-60 TDKDownload_OpenByFile

Item	Description	
Name	Open the video download instance by file	
Pre-condition	Make sure the instance attached with the callback.	
Function	TDKSDKAPI int TDKAPI TDKDownload_OpenByFile(IN TDKHANDLE hDownload, IN const char* url, IN const char* username, IN const char* password, IN const TDK_DOWNLOAD_PARAM_FILE * param);	
Parameter	[in] hDownload	The handle of the instance.
	[in] url	Download request url .
	[in] username	Username.
	[in] password	Password.
	[in] param	The file parameter.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, call TDKDownload_Close to close the instance.	

3.5.7 Close the Download Instance

Table 3-61 TDKDownload_Close

Item	Description	
Name	Close the record download instance	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKDownload_Close(IN TDKHANDLE hDownload);	
Parameter	[in] hDownload	The handle of the instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKDownload_OpenByFile/OpenByTime.	

3.5.8 Start to Download

Table 3-62 TDKDownload_Start

Item	Description	
Name	Start to download the record	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKDownload_Start(IN TDKHANDLE hDownload);	
Parameter	[in] hDownload	The handle of the instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKDownload_Stop.	

3.5.9 Stop Downloading

Table 3-63 TDKDownload_Stop

Item	Description	
Name	Stop downloading the record.	
Pre-condition	Make sure the instance is created and the instance is started.	
Function	TDKSDKAPI int TDKAPI TDKDownload_Stop(IN TDKHANDLE hDownload);	
Parameter	[in] hDownload	The handle of the instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKDownload_Start.	

3.5.10 Get the Download Progress

Table 3-64 TDKDownload_GetProgress

Item	Description	
Name	Get the download progress.	
Pre-condition	Make sure the instance is created and the instance is started.	
Function	TDKSDKAPI int TDKAPI TDKDownload_GetProgress(IN TDKHANDLE hDownload, OUT TDKU32* state, OUT TDKS32* error, OUT TDKU32* progress);	
Parameter	[in] hDownload	The handle of the instance.
	[out] state	The download state.
	[out] error	Error code.
	[out] progress	The download progress, the value is 0-100.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.5.11 Get the Download Bitrate

Table 3-65 TDKDownload_GetBitrate

Item	Description	
Name	Get the download bitrate	
Pre-condition	Make sure the instance is created and the instance is started.	
Function	TDKSDKAPI int TDKAPI TDKDownload_GetBitrate(IN TDKHANDLE hDownload, OUT TDKU64* bitrate);	
Parameter	[in] hDownload	The handle of the instance.
	[out] bitrate	The download bitrate, unit is bps.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.6 Talk Data Transparent Transmission

3.6.1 Create Talk Instance

Table 3-66 TDKTalk_Create

Item	Description
Name	Create a talk instance.
Pre-condition	Make sure the SDK is initialized.
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKTalk_Create(void);
Parameter	None
Return value	Success: Return the handle of the talk instance, non-zero. Failure: Otherwise, refer the TDK_ERROR .
Note	This talk instance is used to transfer audio data between the device and the local platform. [warning] Must to call TDKTalk_Delete in the end.

3.6.2 Delete the Talk Instance

Table 3-67 TDKTalk_Delete

Item	Description
Name	Delete the talk instance.
Pre-condition	Make sure the talk instance is created.
Function	TDKSDKAPI int TDKAPI TDKTalk_Delete(IN TDKHANDLE hTalk);
Parameter	[in] hTalk The handle of talk instance.
Return value	Success: Return the handle of the talk instance. Failure: Otherwise, refer the TDK_ERROR .
Note	Used in pairs with TDKTalk_Create.

3.6.3 Attach Callback Function

Table 3-68 TDKTalk_Attach

Item	Description	
Name	Attach a callback to the talk instance.	
Pre-condition	Make sure the instance and the callback are created.	
Function	TDKSDKAPI int TDKAPI TDKTalk_Attach(IN TDKHANDLE hTalk, IN TDKHANDLE hCallback);	
Parameter	[in] hTalk	The handle of the instance.
	[in] hCallback	The handle of the callback, callback data reference Talk Data .
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, call TDKTalk_Detach to detach it in the end.	

3.6.4 Detach Callback Function

Table 3-69 TDKTalk_Detach

Item	Description	
Name	Detach the callback from the talk instance.	
Pre-condition	Make sure the instance attached with the callback.	
Function	TDKSDKAPI int TDKAPI TDKTalk_Detach(IN TDKHANDLE hTalk, IN TDKHANDLE hCallback);	
Parameter	[in] hTalk	The handle of the talk instance.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKTalk_Attach.	

3.6.5 Open the Talk Instance

Table 3-70 TDKTalk_Open

Item	Description	
Name	Open the talk instance.	
Pre-condition	Make sure the instance is created.	
Function	<pre> TDKSDKAPI int TDKAPI TDKTalk_Open(IN TDKHANDLE hTalk, IN const char* url, IN const char* username, IN const char* password, IN TDKS32 codec, IN TDKU32 samples); </pre>	
Parameter	[in] hTalk	The handle of the instance.
	[in] url	Stream request url.
	[in] username	Username.
	[in] password	Password.
	[in] codec	Local audio encoding type. <pre> #define TDKCODEC_PCMA 4 #define TDKCODEC_PCMU 5 #define TDKCODEC_PCM 9 </pre>
	[in] samples	Local audio sampling rate.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, must to call TDKTalk_Close to close it in the end.	

3.6.6 Send Audio Data to Device

Table 3-71 TDKTalk_SendData

Item	Description	
Name	Send audio data to device.	
Pre-condition	The talk instance must be opened.	
Function	TDKSDKAPI int TDKAPI TDKTalk_SendData(IN TDKHANDLE hTalk, IN char* data, IN TDKS32 length);	
Parameter	[in] hTalk	The handle of the instance.
	[in] data	Audio data.
	[in] length	The length of data.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.6.7 Close the Talk Instance

Table 3-72 TDKTalk_Close

Item	Description	
Name	Close the talk instance.	
Pre-condition	The talk instance must be opened.	
Function	TDKSDKAPI int TDKAPI TDKTalk_Close(IN TDKHANDLE hTalk);	
Parameter	[in] hTalk	The handle of the instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKTalk_Open.	

3.7 Talk and Broadcast

3.7.1 Create Talk Instance

Table 3-73 TDKTalk_CreateEx

Item	Description
Name	Create a talk instance.
Pre-condition	Make sure the SDK is initialized.
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKTalk_CreateEx(void);
Parameter	None
Return value	Success: Return the handle of the talk instance, non-zero. Failure: Otherwise, refer the TDK_ERROR .
Note	This instance can directly decode and play the audio data. [warning] Must to call TDKTalk_DeleteEx after using.

3.7.2 Delete the Talk Instance

Table 3-74 TDKTalk_DeleteEx

Item	Description
Name	Delete the talk instance.
Pre-condition	Make sure the talk instance is created.
Function	TDKSDKAPI int TDKAPI TDKTalk_DeleteEx(IN TDKHANDLE hTalk);
Parameter	[in] hTalk The handle of talk instance
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .
Note	Used in pairs with TDKTalk_CreateEx.

3.7.3 Start to Talk

Table 3-75 TDKTalk_StartEx

Item	Description	
Name	Start to talk.	
Pre-condition	Make sure the talk instance is created.	
Function	TDKSDKAPI int TDKAPI TDKTalk_StartEx(IN TDKHANDLE hTalk, IN const char* url, IN const char* username, IN const char* password);	
Parameter	[in] hTalk	The handle of talk instance.
	[in] url	The request url.
	[in] username	Username.
	[in] password	Password.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] Must to call TDKTalk_StopEx after using.	

3.7.4 Stop Talking

Table 3-76 TDKTalk_StopEx

Item	Description	
Name	Stop talking.	
Pre-condition	Make sure the talk instance is started.	
Function	TDKSDKAPI int TDKAPI TDKTalk_StopEx(IN TDKHANDLE hTalk, IN const char* url, IN const char* username, IN const char* password);	
Parameter	[in] hTalk	The handle of talk instance.
	[in] url	The request url, same as the TDKTalk_StartEx used.
	[in] username	Username.
	[in] password	Password.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKTalk_StartEx.	

3.8 Alarm Listen

3.8.1 Create Alarm Listen Instance

Table 3-77 TDKAlarm_Create

Item	Description
Name	Create an alarm listen instance.
Pre-condition	Make sure the SDK is initialized.
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKAlarm_Create(void);
Parameter	None
Return value	Success: Return the handle of the alarm listen instance, non-zero. Failure: Otherwise, refer the TDK_ERROR .
Note	[warning] Call TDKAlarm_Delete in the end.

3.8.2 Delete the Alarm Listen Instance

Table 3-78 TDKAlarm_Delete

Item	Description
Name	Delete the alarm listen instance.
Pre-condition	Make sure the instance is created.
Function	TDKSDKAPI int TDKAPI TDKAlarm_Delete(IN TDKHANDLE hAlarm);
Parameter	[in] hAlarm The handle of the alarm listen instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .
Note	Used in pairs with TDKAlarm_Create.

3.8.3 Attach Callback Function

Table 3-79 TDKAlarm_Attach

Item	Description	
Name	Attach a callback to the alarm listen instance.	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI int TDKAPI TDKAlarm_Attach(IN TDKHANDLE hAlarm, IN TDKHANDLE hCallback);	
Parameter	[in] hAlarm	The handle of the alarm listen instance.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, call TDKAlarm_Detach to detach it in the end.	

3.8.4 Detach Callback Function

Table 3-80 TDKAlarm_Detach

Item	Description	
Name	Detach the callback from the alarm listen instance.	
Pre-condition	Make sure the instance attached with the callback.	
Function	TDKSDKAPI int TDKAPI TDKAlarm_Detach(IN TDKHANDLE hAlarm, IN TDKHANDLE hCallback);	
Parameter	[in] hAlarm	The handle of the alarm listen instance.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKAlarm_Attach.	

3.8.5 Open the Alarm Listen Instance

Table 3-81 TDKAlarm_Open

Item	Description	
Name	Open the alarm listen instance.	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKAlarm_Open(IN TDKHANDLE hAlarm, IN const char* url, IN const char* username, IN const char* password);	
Parameter	[in] hAlarm	The handle of the alarm listen instance.
	[in] url	Stream request url .
	[in] username	Username.
	[in] password	Password.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, call TDKAlarm_Close to close the instance.	

3.8.6 Close the Alarm Listen Instance

Table 3-82 TDKAlarm_Delete

Item	Description	
Name	Close the alarm listen instance	
Pre-condition	Make sure the instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKAlarm_Delete(IN TDKHANDLE hAlarm);	
Parameter	[in] hAlarm	The handle of the alarm listen instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKAlarm_Open.	

3.9 Device Search

3.9.1 Search All Device in LAN

Table 3-83 TDKDevice_Search

Item	Description	
Name	Search all device in local area network.	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Search(IN const TDK_DEVICE_SEARCH_PARAM * search_param);	
Parameter	[in] search_param	The parameter of search device.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10 Device Remote Config

3.10.1 Create Device Instance

Table 3-84 TDKDevice_Create

Item	Description	
Name	Create a device instance.	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKDevice_Create(void);	
Parameter	None	
Return value	Success: Return the handle of the device instance, non-zero. Failure: 0.	
Note	[warning] Call TDKDevice_Delete in the end.	

3.10.2 Delete the Device Instance

Table 3-85 TDKDevice_Delete

Item	Description	
Name	Delete the device instance.	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Delete(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of device instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKDevice_Create.	

3.10.3 Attach Callback Function

Table 3-86 TDKDevice_Attach

Item	Description	
Name	Attach a callback function to the device instance	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Attach(IN TDKHANDLE hDevice, IN TDKHANDLE hCallback);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] Must to call TDKDevice_Detach to detach it in the end.	

3.10.4 Detach Callback Function

Table 3-87 TDKDevice_Detach

Item	Description	
Name	Detach the callback function from the device instance.	
Pre-condition	Make sure the instance attached with the callback.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Detach(IN TDKHANDLE hDevice, IN TDKHANDLE hCallback);	
Parameter	[in] hFaceSearch	The handle of the device instance.
	[in] hCallback	The handle of the callback.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKDevice_Attach.	

3.10.5 Open the Device Instance

Table 3-88 TDKDevice_Open

Item	Description	
Name	Open the device instance.	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Open(IN TDKHANDLE hDevice, IN const char* address, IN TDKU16 port, IN const char* username, IN const char* password);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] address	Device ip address.
	[in] port	Media port.
	[in] username	Username.
	[in] password	Password
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] Must to call TDKDevice_Close to close the instance.	

3.10.6 Open the Device Instance (ex)

Table 3-89 TDKDevice_OpenEx

Item	Description	
Name	Open the device instance.	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKDevice_OpenEx(IN TDKHANDLE hDevice, IN const char* url, IN const char* username, IN const char* password);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] url	Request url.
	[in] username	Username.
	[in] password	Password.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] Must to call TDKDevice_Close to close the instance.	

3.10.7 Close the Device Instance

Table 3-90 TDKDevice_Close

Item	Description	
Name	Close the device instance.	
Pre-condition	Make sure the instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Close(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of the device instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKDevice_Open.	

3.10.8 Export Config

Table 3-91 TDKDevice_ExportConfig

Item	Description	
Name	Export device current config.	
Pre-condition	Make sure the instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ExportConfig(IN TDKHANDLE hDevice, IN const char* szFileName);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] szFileName	Config file path and name to save.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.9 Import Config

Table 3-92 TDKDevice_ImportConfig

Item	Description	
Name	Import config to the device.	
Pre-condition	Make sure the instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ImportConfig(IN TDKHANDLE hDevice, IN const char* szFileName);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] szFileName	Config file path and name to send.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.10 Device Upgrade

Table 3-93 TDKDevice_Upgrade

Item	Description	
Name	Upgrade the device firmware.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Upgrade(IN TDKHANDLE hDevice, IN const char* szFileName, IN TDKHANDLE hCallback);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] szFileName	The upgrade file path and name.
	[in] hCallback	The handle of the callback that get upgrade progress.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.11 Get the Device Information

Table 3-94 TDKDevice_GetDeviceInfo

Item	Description	
Name	Get the device system information.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetDeviceInfo(IN TDKHANDLE hDevice, OUT TDK_DEVICE_INFO * pdevInfo);	
Parameter	[in] hDevice	The handle of the device instance.
	[out] pdevInfo	The device information.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.12 Get the Device Time

Table 3-95 TDKDevice_GetTime

Item	Description	
Name	Get the device current time.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetTime(IN TDKHANDLE hDevice, OUT TDK_TIME_CONFIG * time);	
Parameter	[in] hDevice	The handle of the device instance.
	[out] time	The device time.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.13 Set the Device Time

Table 3-96 TDKDevice_SetTime

Item	Description	
Name	Set the device time.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetTime(IN TDKHANDLE hDevice, IN const TDK_TIME_CONFIG * time);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] time	The device time to set.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.14 Get the Device OSD Config

Table 3-97 TDKDevice_GetOSDConfig

Item	Description	
Name	Get the device OSD config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetOSDConfig(IN TDKHANDLE hDevice, INOUT TDK OSD CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[inout] config	The device OSD config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.15 Set the Device OSD Config

Table 3-98 TDKDevice_SetOSDConfig

Item	Description	
Name	Set the device OSD config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetOSDConfig(IN TDKHANDLE hDevice, IN const TDK OSD CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	The device OSD config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.16 Get the Device NTP Config

Table 3-99 TDKDevice_GetNTPConfig

Item	Description	
Name	Get the device NTP config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetNTPConfig(IN TDKHANDLE hDevice, OUT TDK NTP CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[out] config	The device NTP config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.17 Set the Device NTP Config

Table 3-100 TDKDevice_SetNTPConfig

Item	Description	
Name	Set the device NTP config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetNTPConfig(IN TDKHANDLE hDevice, IN const TDK NTP CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	The device NTP config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.18 Get the User Account Config

Table 3-101 TDKDevice_GetUserConfig

Item	Description	
Name	Get user account config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetUserConfig(IN TDKHANDLE hDevice, OUT TDK_USER_MANAGE_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	The user account config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.19 Add User Group

Table 3-102 TDKDevice_AddGroup

Item	Description	
Name	Add user group.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_AddGroup(IN TDKHANDLE hDevice, IN const TDK_USER_GROUP_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	The user group config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.20 Modify User Group

Table 3-103 TDKDevice_ModifyGroup

Item	Description	
Name	Modify user group config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ModifyGroup(IN TDKHANDLE hDevice, IN const TDK_USER_GROUP_CONFIG * oldcfg, IN const TDK_USER_GROUP_CONFIG * newcfg);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] old config	Old user group config.
	[in] new config	New user group config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.21 Delete User Group

Table 3-104 TDKDevice_DeleteGroup

Item	Description	
Name	Delete user group config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_DeleteGroup(IN TDKHANDLE hDevice, IN const TDK_USER_GROUP_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	The user group config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.22 Add User Account

Item	Description	
Name	Add user account.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_AddUser(IN TDKHANDLE hDevice, IN const TDK_USER_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	The user account config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.23 Modify User Account

Item	Description	
Name	Modify user account config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ModifyUser(IN TDKHANDLE hDevice, IN const TDK_USER_CONFIG * oldcfg, IN const TDK_USER_CONFIG * newcfg);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] oldcfg	The old user account config.
	[in] newcfg	The new user account config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.24 Delete User Account

Item	Description	
Name	Delete user account.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_DeleteUser(IN TDKHANDLE hDevice, IN const TDK_USER_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	The user account config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.25 Modify User Password

Table 3-105 TDKDevice_ModifyPassword

Item	Description	
Name	Modify user password.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ModifyPassword(IN TDKHANDLE hDevice, IN const TDK_USER_CONFIG * oldcfg, IN const TDK_USER_CONFIG * newcfg);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] oldcfg	The old user config.
	[in] newcfg	The new user config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.26 Get Channel State

Table 3-106 TDKDevice_GetChannelState

Item	Description	
Name	Get the channel state.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetChannelState(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_CHANNEL_STATE * state);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] state	Channel state information.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.27 Get Face Detection Config

Table 3-107 TDKDevice_GetFaceDetection

Item	Description	
Name	Get the face detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetFaceDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_FACE_DETECTION_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Face detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.28 Set Face Detection Config

Table 3-108 TDKDevice_SetFaceDetection

Item	Description	
Name	Set the face detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetFaceDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_FACE_DETECTION_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Face detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.29 Get Line Crossing Config

Table 3-109 TDKDevice_GetLineCrossing

Item	Description	
Name	Get the line crossing config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetLineCrossing(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_LINE_CROSSING_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Line crossing config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.30 Set Line Crossing Config

Table 3-110 TDKDevice_SetLineCrossing

Item	Description	
Name	Set the face detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetLineCrossing(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_LINE_CROSSING_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Face detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.31 Get Area Intrusion Config

Table 3-111 TDKDevice_GetAreaIntrusion

Item	Description	
Name	Get the area intrusion config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetAreaIntrusion(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_AREA_INTRUSION_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Area intrusion config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.32 Set Area Intrusion Config

Table 3-112 TDKDevice_SetAreaIntrusion

Item	Description	
Name	Set the area intrusion config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetAreaIntrusion(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK AREA_INTRUSION_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Area intrusion config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.33 Get Unattended Object Config

Table 3-113 TDKDevice_GetUnattendedObject

Item	Description	
Name	Get the unattended object config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetUnattendedObject(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_UNATTENDED_OBJECT_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Unattended object config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.34 Set Unattended Object Config

Table 3-114 TDKDevice_SetUnattendedObject

Item	Description	
Name	Set the unattended object config.	
Pre-condition	Make sure the device instance is opened.	
Function	<pre>int TDKAPI TDKDevice_SetUnattendedObject(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_UNATTENDED_OBJECT_CONFIG* config);</pre>	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Unattended object config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.35 Get Object Missing Config

Table 3-115 TDKDevice_GetObjectMissing

Item	Description	
Name	Get the object missing config.	
Pre-condition	Make sure the device instance is opened.	
Function	<pre>TDKSDKAPI int TDKAPI TDKDevice_GetObjectMissing(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_OBJECT_MISSING_CONFIG* config);</pre>	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Object missing config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.36 Set Object Missing Config

Table 3-116 TDKDevice_SetObjectMissing

Item	Description	
Name	Set the object missing config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetObjectMissing(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_OBJECT_MISSING_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Object missing config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.37 Get Region Entrance Config

Table 3-117 TDKDevice_GetRegionEntrance

Item	Description	
Name	Get the region entrance config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetRegionEntrance(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_REGION_ENTRANCE_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Region entrance config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.38 Set Region Entrance Config

Table 3-118 TDKDevice_SetRegionEntrance

Item	Description	
Name	Set the region entrance config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetRegionEntrance(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_REGION_ENTRANCE_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Region entrance config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.39 Get Region Exiting Config

Table 3-119 TDKDevice_GetRegionExiting

Item	Description	
Name	Get the region exiting config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetRegionExiting(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_REGION_EXITING_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Region exiting config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.40 Set Region Exiting Config

Table 3-120 TDKDevice_SetRegionExiting

Item	Description	
Name	Set the region exiting config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetRegionExiting(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_REGION_EXITING_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Region exiting config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.41 Get Fast Moving Config

Table 3-121 TDKDevice_GetFastMoving

Item	Description	
Name	Get the fast moving config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetFastMoving(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_FAST_MOVING_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Fast moving config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.42 Set Fast Moving Config

Table 3-122 TDKDevice_SetFastMoving

Item	Description	
Name	Set the fast moving config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetFastMoving(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_FAST_MOVING_CONFIG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Fast moving config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.43 Get Loitering Detection Config

Table 3-123 TDKDevice_GetLoiteringDetection

Item	Description	
Name	Get the loitering detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetLoiteringDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_LOITERING_DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Loitering detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.44 Set Loitering Detection Config

Table 3-124 TDKDevice_SetLoiteringDetection

Item	Description	
Name	Set the loitering detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetLoiteringDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK LOITERING DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Loitering detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.45 Get People Gathering Detection

Table 3-125 TDKDevice_GetPeopleGatheringDetection

Item	Description	
Name	Get the people gathering detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetPeopleGatheringDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK PEOPLE GATHERING DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	People gathering detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.46 Set People Gathering Detection

Table 3-126 TDKDevice_SetPeopleGatheringDetection

Item	Description	
Name	Set the people gathering detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetPeopleGatheringDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK PEOPLE GATHERING DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	People gathering detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.47 Get Parking Detection Config

Table 3-127 TDKDevice_GetParkingDetection

Item	Description	
Name	Get parking detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetParkingDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK PARKING DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Parking detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.48 Set Parking Detection Config

Table 3-128 TDKDevice_SetParkingDetection

Item	Description	
Name	Set parking detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetParkingDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK PARKING DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Parking detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.49 Get Scene Change Detection Config

Table 3-129 TDKDevice_GetSceneChangeDetection

Item	Description	
Name	Set scene change detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetSceneChangeDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK SCENE CHANGE DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Scene change detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.50 Set Scene Change Detection Config

Table 3-130 TDKDevice_SetSceneChangeDetection

Item	Description	
Name	Set scene change detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetSceneChangeDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_SCENE_CHANGE_DETECTION* config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Scene change detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.51 Get Blurred Detection Config

Table 3-131 TDKDevice_GetBlurredDetection

Item	Description	
Name	Get blurred detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetBlurredDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_BLURRED_DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Blurred detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.52 Set Blurred Detection Config

Table 3-132 TDKDevice_SetBlurredDetection

Item	Description	
Name	Set blurred detection config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetBlurredDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_BLURRED_DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Blurred detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.53 Get Audio Exception Detection

Table 3-133 TDKDevice_GetAudioExceptionDetection

Item	Description	
Name	Get audio exception detection.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetAudioExceptionDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_AUDIO_EXCEPTION_DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Audio exception detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.54 Set Audio Exception Detection

Table 3-134 TDKDevice_SetAudioExceptionDetection

Item	Description	
Name	Set audio exception detection.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetAudioExceptionDetection(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK_AUDIO_EXCEPTION_DETECTION * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Audio exception detection config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.55 Get Crossing Line Statistics Config

Table 3-135 TDKDevice_GetCrossingLineStatistics

Item	Description	
Name	Get crossing line statistics config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetCrossingLineStatistics(IN TDKHANDLE hDevice, IN TDKS32 chn, OUT TDK_CROSSING_LINE_STATISTICS * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[out] config	Crossing line statistics config
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.56 Set Crossing Line Statistic Config

Table 3-136 TDKDevice_SetCrossingLineStatistics

Item	Description	
Name	Set crossing line statistics config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetCrossingLineStatistics(IN TDKHANDLE hDevice, IN TDKS32 chn, IN const TDK CROSSING LINE STATISTICS * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] config	Crossing line statistics config
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.57 Clear Crossing Line Statistics Config

Table 3-137 TDKDevice_ClearCrossingLineStatistics

Item	Description	
Name	Clear crossing line statistics config	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ClearCrossingLineStatistics(IN TDKHANDLE hDevice, IN TDKS32 chn);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.58 Get Crossing Line Statistics Report

Table 3-138 TDKDevice_GetCrossingLineStatisticsReport

Item	Description	
Name	Get crossing line statistics report.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetCrossingLineStatisticsReport(IN TDKHANDLE hDevice, IN TDKS32 chn, IN ENUM TDK CROSSING LINE STATISTICS REPORT type, IN TDK SYSTEM TIME * time, OUT TDK CROSSING LINE STATISTICS REPORT * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] chn	Channel index, start from 0.
	[in] type	Crossing line report type.
	[in] time	Time.
	[out] config	Crossing line statistics report.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.59 Get Smart Ability

Table 3-139 TDKDevice_GetSmartSupport

Item	Description	
Name	Get smart ability.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI TDKBOOL TDKAPI TDKDevice_GetSmartSupport(IN TDKHANDLE hDevice, IN ENUM TDK SMART type);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] type	Smart type.
Return value	TRUE: support. FALSE: unsupport.	
Note		

3.10.60 Get the Audio In Ability

Table 3-140 TDKDevice_GetAudioSupport

Item	Description	
Name	Get whether support audio in.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI TDKBOOL TDKAPI TDKDevice_GetAudioSupport(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of the device instance.
Return value	TRUE: support. FALSE: unsupport.	
Note		

3.10.61 Get Whether Storage Normal

Table 3-141 TDKDevice_GetStorageNormal

Item	Description	
Name	Get whether storage normal.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI TDKBOOL TDKAPI TDKDevice_GetStorageNormal(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of the device instance.
Return value	TRUE: Normal. FALSE: Abnormal.	
Note		

3.10.62 Set Device TCP/IP Config

Table 3-142 TDKDevice_SetTCPIP

Item	Description	
Name	Set tcp/ip of the device.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetTCPIP(IN TDKHANDLE hDevice, const IN TDK_TCPIP * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] config	TCP/IP config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.63 Modify the Device Network in LAN

Table 3-143 TDKDevice_ModifyNetwork

Item	Description	
Name	Moidify the device network in LAN.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ModifyNetwork(IN const char* username, IN const char* password, IN const TDK_NETWORK * config, IN TDKU32 timeout);	
Parameter	[in] username	Username.
	[in] password	Password.
	[in] config	Network config.
	[in] timeout	Timeout value.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.64 Reboot Device

Table 3-144 TDKDevice_RebootDevice

Item	Description	
Name	Reboot device.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_RebootDevice(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of device.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.65 Restore the Factory Setting

Table 3-145 TDKDevice_FactoryReset

Item	Description	
Name	Reset the factory setting of the device.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_FactoryReset(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of device.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.66 Get Device Alarm In Config

Table 3-146 TDKDevice_GetAlarmIn

Item	Description	
Name	Get device alarm in config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetAlarmIn(IN TDKHANDLE hDevice, IN TDKS32 index, OUT TDK_ALARM_IN * config);	
Parameter	[in] hDevice	The handle of device.
	[in] index	The channel index, start from 0.
	[out] config	Alarm in config
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.67 Set Device Alarm In Config

Table 3-147 TDKDevice_SetAlarmIn

Item	Description	
Name	Set device alarm in config	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetAlarmIn(IN TDKHANDLE hDevice, IN TDKS32 index, IN const TDK_ALARM_IN * config);	
Parameter	[in] hDevice	The handle of device.
	[in] index	The channel index, start from 0.
	[in] config	Alarm in config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.68 Get device Alarm Out Config

Table 3-148 TDKDevice_GetAlarmOut

Item	Description	
Name	Get device alarm out config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetAlarmOut(IN TDKHANDLE hDevice, IN TDKS32 index, OUT TDK_ALARM_OUT * config);	
Parameter	[in] hDevice	The handle of device.
	[in] index	The channel index, start from 0.
	[out] config	Alarm out config
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.69 Set Device Alarm Out Config

Table 3-149 TDKDevice_SetAlarmOut

Item	Description	
Name	Set device alarm out config	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SetAlarmOut(IN TDKHANDLE hDevice, IN TDKS32 index, IN TDK_ALARM_OUT * config);	
Parameter	[in] hDevice	The handle of device.
	[in] index	The channel index, start from 0.
	[in] config	Alarm out config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.70 PTZ Control

Table 3-150 TDKDevice_PTZControl

Item	Description	
Name	PTZ control operation.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_PTZControl(IN TDKHANDLE hDevice, IN TDKS32 chn, IN ENUM TDK_PTZ_CONTROL cmd, IN TDKS32 param, IN TDK_PTZ_PARAM * param1, IN TDKBOOL stop);	
Parameter	[in] hDevice	The handle of device.
	[in] chn	Channel index, start from 0.
	[in] cmd	PTZ control cmd.
	[in] param	if cmd= TDK_PTZ_UP, TDK_PTZ_DOWN, TDK_PTZ_LEFT, TDK_PTZ_RIGHT, the param is horizontal step length. if cmd= TDK_PTZ_UPPER_LEFT, TDK_PTZ_UPPER_RIGHT, TDK_PTZ_LOWER_LEFT, TDK_PTZ_LOWER_RIGHT, the param is horizontal step length and vertical step length. Otherwise the param is 0.
	[in] param1	Reserved parameter, set to NULL.
	[in] stop	Start or stop.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.71 Create Device Online Manage Instance

Table 3-151 TDKDeviceOnline_Create

Item	Description	
Name	Create a device online manage instance.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKDeviceOnline_Create(IN TDKHANDLE hCallback);	
Parameter	[in] hCallback	The handle of callback
Return value	Success: The handle of the device online instance, non-zero. Failure: 0.	
Note		

3.10.72 Query Device Whether Online

Table 3-152 TDKAPI TDKDeviceOnline_Query

Item	Description	
Name	Query device whether online.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDeviceOnline_Query(IN TDKHANDLE hOnline, IN const char* url);	
Parameter	[in] hOnline	The handle of the device online instance.
	[in] url	Device url, reference URL .
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.73 Delete the Device Online Manage Instance

Table 3-153 TDKDeviceOnline_Delete

Item	Description	
Name	Delete the device online manage instance.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDeviceOnline_Delete(IN TDKHANDLE hOnline);	
Parameter	[in] hOnline	The handle of the device online instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.74 Get Device Channel Number

Table 3-154 TDKDevice_GetChannelCount

Item	Description	
Name	Get the device channel number.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetChannelCount(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of the device instance.
Return value	Success: Positive number. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.75 Get Device Record Status

Table 3-155 TDKDevice_GetSYSrecordstatus

Item	Description	
Name	Get device record status.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_GetSYSrecordstatus(IN TDKHANDLE hDevice, OUT TDKS32& count, OUT TDK SYS RECORDSTATUS *& config);	
Parameter	[in] hDevice	The handle of the device instance.
	[out] count	Channel count.
	[out] config	The config array of system record status.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.76 Get Log Config Ability

Table 3-156 TDKDevice_SYSlogINFO_GetCFG

Item	Description	
Name	Get system log information config ability.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SYSlogINFO_GetCFG(IN TDKHANDLE hDevice, OUT TDK SYS LOGINFOCFG * config);	
Parameter	[in] hDevice	The handle of the device instance.
	[out] config	The config ability of log information.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.10.77 Query Log Information

Table 3-157 TDKDevice_SYSlogINFO_Query

Item	Description	
Name	Query system log information.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SYSlogINFO_Query(IN TDKHANDLE hDevice, IN TDK_SYS_LOGININFOQUERY * pQuery, OUT TDKS32& count, OUT TDK_SYS_LOGININFO *& config);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] pQuery	Log query parameter.
	[out] count	Log information count.
	[out] config	Log information array.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.78 Clear Log Information

Table 3-158 TDKDevice_SYSlogINFO_Clear

Item	Description	
Name	Clear all system log information config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_SYSlogINFO_Clear(IN TDKHANDLE hDevice);	
Parameter	[in] hDevice	The handle of the device instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.79 Get Encode Config

Table 3-159 TDKDevice_ENC_GetCFG

Item	Description	
Name	Get device channel encode config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ENC_GetCFG(IN TDKHANDLE hDevice, INOUT TDK_ENC_CHNCFG * pCfg);	
Parameter	[in] hDevice	The handle of the device instance.
	[inout] pCfg	Channel encode config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] The pCode field in TDK_ENC_CHNCFG must be released by TDKDevice_ENC_Release function.	

3.10.80 Get FPS Config

Table 3-160 TDKDevice_ENC_GetCFG_Fps

Item	Description	
Name	Get device channel fps config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ENC_GetCFG_Fps(IN TDKHANDLE hDevice, INOUT TDK_ENC_QUERYFPS* pCfg);	
Parameter	[in] hDevice	The handle of the device instance.
	[inout] pCfg	Channel fps config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.81 Get Bitrate Config

Table 3-161 TDKDevice_ENC_GetCFG_Bit

Item	Description	
Name	Get device channel bitrate config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ENC_GetCFG_Bit(IN TDKHANDLE hDevice, INOUT TDK_ENC_QUERYBITRATE * pQuery);	
Parameter	[in] hDevice	The handle of the device instance.
	[inout] pQuery	The channel bitrate config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.82 Set Encode Config

Table 3-162 TDKDevice_ENC_SetCFG

Item	Description	
Name	Set channel encode config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ENC_SetCFG(IN TDKHANDLE hDevice, IN TDKENCONDCEFGDATA* cfg, OUT TDKS32* status);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] cfg	Encode config data.
	[out] status	Device whether need reboot. 1- need reboot. 0- not need reboot.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.83 Get Sub Stream Resolutuin

Table 3-163 TDKDevice_ENC_GetCFG_SubRes

Item	Description	
Name	Get sub stream resolution when main stream resolution changed.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ENC_GetCFG_SubRes(IN TDKHANDLE hDevice, INOUT TDK_ENC_QUERYRES * pQuery);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] pQuery	The data of query.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.84 Release Encode Config Data

Table 3-164 TDKDevice_ENC_Release

Item	Description	
Name	Release device encode config data.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_ENC_Release(IN TDKHANDLE pcodecfg);	
Parameter	[in] pcodecfg	Device encode config handle, the pCode field in TDK_ENC_CHNCFG that gotten by TDKDevice_ENC_GetCFG function.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.85 Get Area Cover Config

Table 3-165 TDKDevice_Cover_GetCFG

Item	Description	
Name	Get device area cover config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Cover_GetCFG(IN TDKHANDLE hDevice, INOUT TDK_ENC_COVER * pCover);	
Parameter	[in] hDevice	The handle of the device instance.
	[inout] pCover	Device area cover config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.86 Set Area Cover Config

Table 3-166 TDKDevice_Cover_SetCFG

Item	Description	
Name	Set device area cover config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Cover_SetCFG(IN TDKHANDLE hDevice, IN TDK_ENC_COVER * pCover);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] pCover	Device cover config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.87 Get OSD Config

Table 3-167 TDKDevice_OSDSetting_CFG_GET

Item	Description	
Name	Get device OSD config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_OSDSetting_CFG_GET(IN TDKHANDLE hDevice, INOUT TDK_ENC_OSD* pOSD);	
Parameter	[in] hDevice	The handle of the device instance.
	[inout] pOSD	Device osd config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.88 Set OSD Config

Table 3-168 TDKDevice_OSDSetting_CFG_SET

Item	Description	
Name	Set device OSD config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_OSDSetting_CFG_SET(IN TDKHANDLE hDevice, IN TDK_ENC_OSD* pOSD);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] pOSD	Device osd config.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.89 Get Device TCP/IP Config

Table 3-169 TDKDevice_Network_GETTCPIP

Item	Description	
Name	Get device tcp/ip config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Network_GETTCPIP(IN TDKHANDLE hDevice, OUT TDK_DEVICE_TCPIP * cfg);	
Parameter	[in] hDevice	The handle of the device instance.
	[out] cfg	The tcp/ip config of device.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] The pcfg field in TDK_DEVICE_TCPIP must be released by TDKDevice_Network_REALESETCPCFG function.	

3.10.90 Check IP Conflict

Table 3-170 TDKDevice_Network_CheckIPConflict

Item	Description	
Name	Check whether device ip conflict.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Network_CheckIPConflict(IN TDKHANDLE hDevice, IN TDKHANDLE IP);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] IP	IP address string.
Return value	Success: 0-not conflict, 1-conflict. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.91 Set Device TCP/IP Config

Table 3-171 TDKDevice_Network_SETTCPIP

Item	Description	
Name	Set device TCP/IP config.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Network_SETTCPIP(IN TDKHANDLE hDevice, IN TDK_DEVICE_TCPIP * cfg, OUT TDKHANDLE pState);	
Parameter	[in] hDevice	The handle of the device instance.
	[in] cfg	Device TCP/IP config.
	[out] pState	Device state. Bit 0 – whether need relogin. Bit 1 – whether need reboot.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.10.92 Release TCP/IP Config Data

Table 3-172 TDKDevice_Network_REALESETCPCFG

Item	Description	
Name	Release device TCP/IP config data.	
Pre-condition	Make sure the device instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKDevice_Network_REALESETCPCFG(IN TDKHANDLE pTCPCFG);	
Parameter	[in] pTCPCFG	The device TCP/IP config handle, is the pcfg field in the TDK_DEVICE_TCPIP gotten by TDKDevice_Network_GETTCPIP function.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.11 Smart Stream

3.11.1 Create Smart Stream Instance

Table 3-173 TDKSmart_Create

Item	Description
Name	Create a smart stream instance.
Pre-condition	Make sure the SDK is initialized.
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKSmart_Create(void);
Parameter	None.
Return value	Success: Return the handle of the smart stream instance, non-zero. Failure: Otherwise, refer the TDK_ERROR .
Note	[warning] Must to call TDKSmart_Delete in the end.

3.11.2 Delete the Smart Stream Instance

Table 3-174 TDKSmart_Delete

Item	Description
Name	Delete the smart stream instance.
Pre-condition	Make sure the instance is created.
Function	TDKSDKAPI int TDKAPI TDKSmart_Delete(IN TDKHANDLE hStream);
Parameter	[in] hStream The handle of the smart stream instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .
Note	Used in pairs with TDKSmart_Create.

3.11.3 Attach Callback Function

Table 3-175 TDKSmart_Attach

Item	Description	
Name	Attach a callback function to the smart stream instance	
Pre-condition	Make sure the SDK is initialized.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Attach(IN TDKHANDLE hStream, IN TDKHANDLE hCallback);	
Parameter	[in] hStream	The handle of the smart stream instance.
	[in] hCallback	The handle of the callback function.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, must to call TDKSmart_Detach to detach it in the end.	

3.11.4 Detach Callback Function

Table 3-176 TDKSmart_Detach

Item	Description	
Name	Detach the callback fuction from the smart stream instance.	
Pre-condition	Make sure the instance attached with the callback.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Detach(IN TDKHANDLE hStream, IN TDKHANDLE hCallback);	
Parameter	[in] hStream	The handle of the smart stream instance.
	[in] hCallback	The handle of the callback function.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKFaceSearch_Detach.	

3.11.5 Open the Smart Stream Instance

Table 3-177 TDKSmart_open

Item	Description	
Name	Open the smart stream instance.	
Pre-condition	Make sure the instance is created.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Open(IN TDKHANDLE hStream, IN const char* address, IN TDKU16 port, IN const char* username, IN const char* password);	
Parameter	[in] hStream	The handle of the smart stream instance.
	[in] address	IP address.
	[in] port	Port.
	[in] username	Username.
	[in] password	Password.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	[warning] After using, must to call TDKSmart_Close to close the instance.	

3.11.6 Close the Smart Stream Instance

Table 3-178 TDKSmart_Close

Item	Description	
Name	Close the smart stream instance	
Pre-condition	Make sure the instance is opened.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Close(IN TDKHANDLE hStream);	
Parameter	[in] hStream	The handle of the smart stream instance.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note	Used in pairs with TDKSmart_Open.	

3.11.7 Get Picture in the Smart Frame

Table 3-179 TDKSmart_Frame_GetPicture

Item	Description	
Name	Get picture in the smart frame.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Frame_GetPicture(IN TDKHANDLE hFrame, INOUT TDK SMART PIC * pic);	
Parameter	[in] hFrame	The handle of the smart stream instance.
	[inout] pic	Snapshot picture.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.11.8 Get Face Count

Table 3-180 TDKSmart_Frame_GetFaceNum

Item	Description	
Name	Get face count in the smart frame	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Frame_GetFaceNum(IN TDKHANDLE hFrame);	
Parameter	[in] hFrame	The handle of the smart stream instance.
Return value	Success: The face count. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.11.9 Get the Face Picture Instance

Table 3-181 TDKSmart_Frame_GetFace

Item	Description	
Name	Get the face picture instance.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKSmart_Frame_GetFace(IN TDKHANDLE hFrame, IN TDKS32 idx);	
Parameter	[in] hStream	The handle of the smart stream instance.
	[in] idx	The index of face picture.
Return value	Success: The handle of face picture instance, non-zero. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.11.10 Get Face Information

Table 3-182 TDKSmart_Face_GetInfo

Item	Description	
Name	Get face information in the face picture.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI const TDK_FACEOBJ * TDKAPI TDKSmart_Face_GetInfo(IN TDKHANDLE hFace);	
Parameter	[in] hFace	The handle of the face picture instance.
Return value	Success: The pointer of face data object, non-zero. Failure: NULL.	
Note		

3.11.11 Get Face Attribute

Table 3-183 TDKSmart_Face_GetAttr

Item	Description	
Name	Get face attribute in the face picture.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI const TDK_FACE_ATTR * TDKAPI TDKSmart_Face_GetAttr(IN TDKHANDLE hFace);	
Parameter	[in] hFace	The handle of the face picture instance.
Return value	Success: The pointer of face attribute data, non-zero. Failure: NULL.	
Note		

3.11.12 Get Face Picture

Table 3-184 TDKSmart_Face_GetPicture

Item	Description	
Name	Get face picture in the face picture instance.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Face_GetPicture(IN TDKHANDLE hFace, INOUT TDK_SMART_PIC * pic);	
Parameter	[in] hFace	The handle of the face picture instance.
	[inout] pic	Picture data.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

3.11.13 Get Face Picture with Shoulder

Table 3-185 TDKSmart_Face_GetPictureEx

Item	Description	
Name	Get face picture with shoulder in the face picture instance.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Face_GetPictureEx(IN TDKHANDLE hFace, INOUT TDK SMART PIC * pic);	
Parameter	[in] hFace	The handle of the face picture instance.
	[inout] pic	Picture data.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.11.14 Get License Plate Count

Table 3-186 TDKSmart_Frame_GetLicensePlateNum

Item	Description	
Name	Get the number of license plate number in the picture.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI int TDKAPI TDKSmart_Frame_GetLicensePlateNum(IN TDKHANDLE hFrame);	
Parameter	[in] hFrame	The handle of the frame.
Return value	Success: License plate count. Failure: Otherwise, refer the TDK ERROR .	
Note		

3.11.15 Get the License Plate Picture Instance

Table 3-187 TDKSmart_Frame_GetLicensePlate

Item	Description	
Name	Get the license plate picture instance.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI TDKHANDLE TDKAPI TDKSmart_Frame_GetLicensePlate(IN TDKHANDLE hFrame, IN TDKS32 idx);	
Parameter	[in] hFrame	The handle of the frame.
	[in] idx	The index of the plate, start from 0.
Return value	Success: The handle of license plate. Failure: NULL.	
Note		

3.11.16 Get License Plate Information

Table 3-188 TDKSmart_LicensePlate_GetInfo

Item	Description	
Name	Get license plate information.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI const TDK_LICENCEPALTE_OBJ * TDKAPI TDKSmart_LicensePlate_GetInfo(IN TDKHANDLE hLicensePlate);	
Parameter	[in] hLicensePlate	The handle of the license plate instance.
Return value	Success: The pointer of licence plate data object. Failure: NULL.	
Note		

3.11.17 Get License Plate Picture

Table 3-189 TDKSmart_LicensePlate_GetPicture

Item	Description	
Name	Get license plate picture.	
Pre-condition	The smart stream must be opened.	
Function	TDKSDKAPI int TDKAPI TDKSmart_LicensePlate_GetPicture(IN TDKHANDLE hLicensePlate, INOUT TDK SMART PIC * pic);	
Parameter	[in] hLicensePlate	The handle of the license plate instance.
	[inout] pic	License plate picture data.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK ERROR .	
Note		

4 Callback

4.1 Create SDK Callback Function

Table 4-1 TDKCBCreate

Item	Description	
Name	Create a SDK callback function.	
Pre-condition	You should implement a callback function in the format of TDKCALLBACK before use the API.	
Function	TDKHANDLE TDKAPI TDKCBCreate(IN void* context, IN TDKCALLBACK func);	
Parameter	[in] context	User data. SDK will return the data to user for subsequent operation.
	[in] func	The callback fuction that implemented in the format of TDKCALLBACK.
Return value	Success: Non-zero. Failure: 0.	
Note	[warning]Must to call TDKCBDelete in the end.	

4.2 Delete SDK Callback Function

Table 4-2 TDKCBDelete

Item	Description	
Name	Delete SDK callback function.	
Pre-condition	You have got a handle of SDK callback by calling TDKCBCreate.	
Function	void TDKAPI TDKCBDelete(IN TDKHANDLE hCallback);	
Parameter	[in] hCallback	The handle of the callback you want to delete.
Return value	None.	
Note	Used in pairs with TDKCBCreate.	

4.3 The Format of the TDKCALLBACK

Item	Description	
Name	SDK callback function format. (The parameters will have different mean in different module or different CMD)	
Pre-condition	None	
Function	typedef int (TDKAPI *TDKCALLBACK)(void* context, TDKU32 cmd, void* param0, void* param1, void* param2);	
Parameter	context	User data. SDK will return the data to user for subsequent operation.
	cmd	Message type.
	param0	The pointer of instance that attach callback.
	param1	State, frame data.
	param2	Error code.
Return value	Success: TDK_OK. Failure: Otherwise, refer the TDK_ERROR .	
Note		

4.3.1 Preview and Playback

CMD	Value	Discription	p0	p1	p2
TDKPLAYER_MSG_STATE	2000	The state of the player is changed.		TDKU32 state	TDK_ERROR
TDKPLAYER_MSG_FRAME	2001	The decoded data frame.		TDK_PICTURE*	NULL
TDKPLAYER_MSG_PACKET	2002	Received a new data frame from device.		TDK_FRAMEHEAD*	NULL
TDKSMART_MSG_FRAME	1001	Received a smart frame.		TDKHANDLE: The handle of the smart frame instance.	NULL

4.3.2 Record Download

CMD	Value	Discription	p0	p1	p2
TDKDOWNLOAD_MSG_PACKET	2200	Received a new data frame from device.		TDK_FRAMEHEAD*	NULL

4.3.3 Talk Data

CMD	Value	Discription	p0	p1	p2
TDKTALK_MSG_STATE	3001	The state of the talk instance is changed.		TDKU32 state	TDK_ERROR
TDKTALK_MSG_PACKET	3000	Received a new data frame.		TDK_FRAMEHEAD*	NULL

4.3.4 Alarm Listen

CMD	Value	Discription	p0	p1	p2
TDKALARM_MSG_ALARM	3100	Recieve a new alarm event form device.		TDK_ALARM_INFO *	NULL

The type of the Alarm:

#define TDKALARM_UNKNOWN	0	//Unknown type
#define TDKALARM_PORT_ALARMIN	1	//Alarm input
#define TDKALARM_MOTION	2	//Motion detect alarm
#define TDKALARM_VIDEOLOST	3	//Video lost alarm
#define TDKALARM_SHELTER	4	//Shelter alarm
#define TDKALARM_DISKFULL	5	//Disk full alarm
#define TDKALARM_DISKERROR	6	//Disk error alarm
#define TDKALARM_NO_DISK	7	//No disk alarm
#define TDKALARM_CROSSLINE	8	//Cross-line alarm
#define TDKALARM_AREA_INTRUSION	9	//Area intrusion alarm
#define TDKALARM_IP_CONFLICT	10	//IP conflict alarm
#define TDKALARM_HDD_SMART	11	//HDD S.M.A.R.T alarm
#define TDKALARM_REGION_ENTRANCE	12	//Region entrance alarm
#define TDKALARM_REGION_EXIT	13	//Region exiting alarm
#define TDKALARM_FAST_MOVE	14	//Fast moving alarm
#define TDKALARM_OBJECT_UNATTEND	15	//Unattended object alarm
#define TDKALARM_OBJECT_MISS	16	//Object missing alarm
#define TDKALARM_FACE_DETECT	17	//Face detection alarm
#define TDKALARM_LOITER_DETECT	18	//Loitering detection alarm
#define TDKALARM_PARK_DETECT	19	//Parking detection alarm
#define TDKALARM_PEOPLE_GATHER	20	//People gathering alarm
#define TDKALARM_DEFOCUS_DETECT	21	//Defocus detection alarm
#define TDKALARM_SCENE_CHANGE	22	//Scene change alarm
#define TDKALARM_AUDIO_EXCEPTION	23	//Audio exception alarm
#define TDKALARM_SOUND_MUTATION	24	//Sound mutation alarm
#define TDKALARM_SOUND_ZOOM	25	//Sound Zoom alarm
#define TDKALARM_SOUND_PLUNGE	26	//Sound Plunge alarm

4.3.5 Device Config

CMD	Value	Discription	p0	p1	p2
TDKDEVICE_MSG_UPGRADE_PROGRESS	3200	The progress of the update is changed.		(TDKU32)progress	NULL
TDKDEVICE_MSG_DEV_SEARCH	3201	The device information searched.		TDK_DEVICE_NET_INFO*	NULL
TDKDEVICE_MSG_STATE	3202	The state of the device is changed.		TDKU32 state	TDK_ERROR
TDKDEVICE_MSG_ONLINE	3203	The device whether online		(char*)Device ip or cloud id.	1- online. 0- offline.

4.3.6 Smart Stream

CMD	Value	Discription	p0	p1	p2
TDKSMART_MSG_STATE	1000	The state of the smart stream instance is changed.		TDKU32 state	TDK_ERROR
TDKSMART_MSG_FRAME	1001	Received a new smart frame.		TDKHANDLE: The handle of the smart frame instance.	NULL

4.4 Stream State Code

Item	Value	Discription
TDKSTATE_STREAM_CLOSE	0	The stream is closed.
TDKSTATE_STREAM_OPEN	1	The stream is opened.
TDKSTATE_STREAM_CONNECTING	2	The stream is connecting.
TDKSTATE_STREAM_DISCONNECT	3	The stream is disconnected.
TDKSTATE_STREAM_START	4	The stream is started.
TDKSTATE_STREAM_STOP	5	The stream is stoped.
TDKSTATE_STREAM_RECONNECT	6	The stream is reconnected.

5 Structure Definition

5.1 TDK_FILE_INFO

Table 5-1 TDK_FILE_INFO

Option	Instruction	
Struct description	Record file information.	
Structure	<pre>typedef struct tag_TDK_FILE_INFO { TDKU8 filetype; TDKU8 rectype; TDKU8 channel; TDKU8 streamtype; TDKU8 lockstatus; TDKU8 codec; TDKU8 res[2]; TDK_SYSTEM_TIME starttime; TDK_SYSTEM_TIME endtime; TDKU64 size; TDKU32 picindex; TDKU8 filename[80]; TDKU8 res1[32]; }TDK_FILE_INFO;</pre>	
Members	filetype	File type (0-unknown, 1-media, 2-picture).
	rectype	Record type (0-timer, 1-manual, 2-event, 255-all).
	channel	Channel index, start from 0.
	streamtype	Stream type (0-main, 1-sub, 2-third).
	lockstatus	0-unlocked, 1-locked.
	codec	Codec type, TDKCODEC_XXX (0-None, 1-H264, 2-H265, 3-JPEG, 4-PCMA, 5-PCMU, 6-ADPCM, 7-MP3, 8-AAC, 9-PCM, 10-SMART, 11-CURRC).
	res[2]	Reserve byte.
	starttime	Start time, format refer to TDK_SYSTEM_TIME .
	endtime	End time, format refer to TDK_SYSTEM_TIME .
	size	File size, the unit is byte.
	picindex	This parameter is valid when the file type is picture.
	filename[80]	File name string.
	res1[32]	Reserve byte.

5.2 TDK_SYSTEM_TIME

Table 5-2 TDK_SYSTEM_TIME

Option	Instruction	
Struct description	Time format.	
Structure	<pre>typedef struct tag_TDK_SYSTEM_TIME { TDKS32 year; TDKS32 month; TDKS32 day; TDKS32 wday; TDKS32 hour; TDKS32 minute; TDKS32 second; TDKS32 isdst; }TDK_SYSTEM_TIME;</pre>	
Members	year	Year [2000, 2037].
	month	Month, January = 1, February = 2, and so on.
	day	Month day.
	wday	Week day, Sunday = 0, Monday = 1, and so on.
	hour	Hour.
	minute	Minute.
	second	Second.
	isdst	Flag daylight saving time.

5.3 RECT

Table 5-3 RECT

Option	Instruction	
Struct description	Rectangle coordinate value.	
Structure	<pre>typedef struct tagRECT { LONG left; LONG top; LONG right; LONG bottom; } RECT, *PRECT, NEAR *NPRECT, FAR *LPRECT;</pre>	
Members	left	Left coordinate value.
	top	Top coordinate value.
	right	Right coordinate value.
	bottom	Bottom coordinate value.

5.4 TDK_DOWNLOAD_PARAM_TIME

Table 5-4 TDK_DOWNLOAD_PARAM_TIME

Option	Instruction	
Struct description	Relevant parameters of record download by time	
Structure	<pre>typedef struct tag_TDK_DOWNLOAD_PARAM_TIME { TDKU32 channel; TDKU32 filetype; TDKU32 recordtype; TDKU32 streamtype; TDK_SYSTEM_TIME starttime; TDK_SYSTEM_TIME endtime; char destpath[256]; TDK_RECORD_FORMAT format; }TDK_DOWNLOAD_PARAM_TIME;</pre>	
Members	channel	Channel index, start from 0.
	filetype	File type (0-unknown, 1-media, 2-picture).
	recordtype	Record type (0-timer, 1-manual, 2-event, 255-all).
	streamtype	Stream type (0-main, 1-sub, 2-third).
	starttime	Start time.
	endtime	End time.
	destpath	File saving path.
	format	File saving format, (This parameter is valid when the file type is TDK_FILE_MEDIA).

5.5 TDK_DOWNLOAD_PARAM_FILE

Table 5-5 TDK_DOWNLOAD_PARAM_FILE

Option	Instruction	
Struct description	Relevant parameters of record download by file	
Structure	<pre>typedef struct tag_TDK_DOWNLOAD_PARAM_FILE { TDKU32 count; TDK_FILE_INFO* fileList; char destpath[256]; TDK_RECORD_FORMAT format; }TDK_DOWNLOAD_PARAM_FILE;</pre>	
Members	count	File number to download.
	filelist	List of files.
	destpath[256]	File saving path.
	format	File saving format.

5.6 TDK_FILE_SEARCH_PARAM

Table 5-6 TDK_FILE_SEARCH_PARAM

Option	Instruction	
Struct description	Relevant parameters of file search	
Structure	<pre>typedef struct tag_MZM_FILE_SEARCH_PARAM { TDKU8 filetype; TDKU8 rectype; TDKU8 res[6]; TDKU64 chnmask0; TDKU64 chnmask1; TDKU8 streamtype; TDKU8 res1[3]; TDK_SYSTEM_TIME starttime; TDK_SYSTEM_TIME endtime; TDKU8 res2[28]; }TDK_FILE_SEARCH_PARAM;</pre>	
Members	filetype	File type(0-unknown, 1-media, 2-picture).
	rectype	Record type(0-timer, 1-manual,2-event, 255-all).
	res[6]	Reserve byte.
	chnmask0	Channel mask, first 64 channels.
	chnmask1	Channel mask, back 64 channels.
	streamtype	Stream type (0-main, 1-sub, 2-third).
	res1[3]	Reserve byte.
	starttime	Start time.
	endtime	End time.
	res2[28]	Reserve byte.

5.7 TDK_FRAMEHEAD

Table 5-7 TDK_FRAMEHEAD

Option	Instruction	
Struct description	Frame head.	
Structure	<pre>typedef struct tag_TDK_FRMHDR { TDKU8 prefix[4]; TDKU32 length; TDK_TIME tms; TDKU16 ms; TDKU8 codec; union { TDKU8 flag; TDKU8 num; TDKU8 fps; TDKU8 channum; }; union { TDKU16 width; TDKU16 freq; }; TDKU16 height; }TDK_FRAMEHEAD, *TDK_PFRAMEHEAD;</pre>	
Members	prefix[4]	Prefix[0] = 0, prefix[1] = 0, prefix[2] = 1, prefix[3] = TDKFRAME_BASE + TDKFRAME_XXX.
	length	The length of frame data.
	tms	Absolute time to the second.
	ms	Millisecond (0-999).
	codec	Codec type, TDKCODEC_XXX (0-NONE, 1-H264, 2-H265, 3-JPEG, 4-PCMA, 5-PCMU, 6-ADPCM, 7-MP3, 8-AAC, 9-PCM, 10-SMART, 11-CURRC).
	flag	Whether the smart frame object is marked enum PackSmartFlag value. The highest bit is used to mark whether there is additional data.
	num	Number of smart objects in a smart frame.
	fps	Video frames per second, 6.2 format.
	channum	Audio channel numbers.
	width	Video width.
	freq	Audio sampling rate.
	height	Video height.

5.8 TDK_PICTURE

Table 5-8 TDK_PICTURE

Option	Instruction	
Struct description	Relevant parameters of picture.	
Structure	<pre>typedef struct tag_TDK_PICTURE { TDKU16 format; TDKU16 res; TDKU32 width; TDKU32 height; TDKU32 linesnum[3]; TDKU32 linesize[3]; TDKU32 pitch[3]; TDKU8* data[3]; }TDK_PICTURE;</pre>	
Members	format	Picture format.
	res	Reserve byte.
	width	Picture width.
	height	Picture height.
	linesnum[3]	The number of line.
	linesize[3]	Number of valid data bytes per line.
	pitch[3]	The number of bytes between two lines.
	data[3]	Data buffer address.

5.9 TDK_ALARM_INFO

Table 5-9 TDK_ALARM_INFO

Option	Instruction	
Struct description	Relevant parameters of alarm information.	
Structure	<pre>typedef struct tag_TDK_ALARM_INFO { TDKU16 type; TDKU16 count; TDK_ALARM_INFO_CHN info[TDK_MAX_CHN_NUM]; TDKU8 res[32]; }TDK_ALARM_INFO;</pre>	
Members	type	Alarm type.
	count	Alarm count.
	info[TDK_MAX_CHN_NUM]	Alarm information.
	res[32]	Reserve byte.

5.10 TDK_ALARM_INFO_CHN

Table 5-10 TDK_ALARM_INFO_CHN

Option	Instruction	
Struct description	Relevant parameters of channel alarm information.	
Structure	<pre>typedef struct tag_TDK_ALARM_INFO_CHN { TDKU16 chn; TDKU16 type; TDKU16 status; TDKU16 number; TDKU8 res[32]; }TDK_ALARM_INFO_CHN;</pre>	
Members	chn	Channel index, start from 0.
	type	Alarm type.
	status	0-no alarm, 1-alarm.
	number	Number of alarm region or line, for smart.
	res[32]	Reserve byte.

5.11 TDK_SNAPSHOT_INFO

Table 5-11 TDK_SNAPSHOT_INFO

Option	Instruction	
Struct description	Relevant parameters of snapshot information.	
Structure	<pre>typedef struct tag_TDK_SNAPSHOT_INFO { TDKU32 alarmCount; TDK_ALARM_INFO_CHN alarmInfo[TDK_MAX_CHN_NUM]; TDK_SNAPSHOT_PIC pic; }TDK_SNAPSHOT_INFO;</pre>	
Members	alarmCount	The count of alarm.
	alarmInfo[TDK_MAX_CHN_NUM]	Alarm information.
	pic	Snapshot picture.

5.12 TDK_SNAPSHOT_PIC

Table 5-12 TDK_SNAPSHOT_PIC

Option	Instruction	
Struct description	Snapshot information	
Structure	<pre>typedef struct tag_TDK_SNAPSHOT_PIC { TDKU16 codec; TDKU16 w; TDKU16 h; TDKU32 len; unsigned char* data; }TDK_SNAPSHOT_PIC, *PTDK_SNAPSHOT_PIC;</pre>	
Members	codec	Codec type, TDKCODEC_XXX (0-None, 1-H264, 2-H265, 3-JPEG, 4-PCMA, 5-PCMU, 6-ADPCM, 7-MP3, 8-AAC, 9-PCM, 10-SMART, 11-CURRC).
	w	Width.
	h	Height.
	len	Data length.
	data	Data buffer address.

5.13 TDK_PICTURE_DATA

Table 5-13 TDK_PICTURE_DATA

Option	Instruction	
Struct description	Picture data.	
Structure	<pre>typedef struct tag_TDK_PICTURE_DATA { char* picdata; TDKU32 piclen; }TDK_PICTURE_DATA, PTDK_PICTURE_DATA;</pre>	
Members	picdata	Picture data buffer address.
	piclen	Picture data length.

5.14 TDK_FACE_SEARCH_INFO

Table 5-14 TDK_FACE_SEARCH_INFO

Option	Instruction	
Struct description	Face search information.	
Structure	<pre>typedef struct tag_TDK_FACE_SEARCH_INFO { TDKU8 id; TDKU8 chnid; char cloudid[32]; char* pic; TDKU32 piclen; TDK_SYSTEM_TIME time; float sim; }TDK_FACE_SEARCH_INFO, *PTDK_FACE_SEARCH_INFO;</pre>	
Members	id	Face picture information id.
	chnid	Channel id.
	cloudid[32]	Cloud id.
	pic	Reference picture data.
	piclen	Reference picture data length.
	time	Search time.
	sim	Similarity.

5.15 TDK_DEVICE_SEARCH_PARAM

Table 5-15 TDK_DEVICE_SEARCH_PARAM

Option	Instruction	
Struct description	Device Search parameter.	
Structure	typedef struct { TDKHANDLE callback; TDKU32 timeout; }TDK_DEVICE_SEARCH_PARAM;	
Members	callback	Create by TDKCBCreate.
	timeout	Search time, millisecond.

5.16 TDK_DEVICE_INFO

Table 5-16 TDK_DEVICE_INFO

Option	Instruction	
Struct description	Device information.	
Structure	<pre>typedef struct { TDKU8 devtype; TDKU16 analogchnnum; TDKU16 digitalchnnum; TDKU8 alarmin; TDKU8 alarmout; TDKU8 audioin; TDKU8 audioout; TDKU8 devmodel[128]; TDKU8 devversion[128]; TDKU64 digital[3]; TDKU64 analog[3]; }TDK_DEVICE_INFO;</pre>	
Members	devtype	Device type, refer TDK_DEVICE_TYPE .
	analogchnnum	Number of analog channels.
	digitalchnnum	Number of digital channels, total number of channels: analogchnnum + digitalchnnum.
	alarmin	Number of alarm input port.
	alarmout	Number of alarm output port.
	audioin	Number of audio in.
	audioout	Number of audio out.
	devmodel[128]	Device model
	devversion[128]	Device version.
	digital[3]	Digital channel index in channel order.
	analog[3]	Analog channel index in channel order.

5.17 TDK_TIME_CONFIG

Table 5-17 TDK_TIME_CONFIG

Option	Instruction	
Struct description	Time config.	
Structure	<pre>typedef struct tag_TDK_TIME_CONFIG { TDKS8 timezone; TDK_SYSTEM_TIME time; }TDK_TIME_CONFIG;</pre>	
Members	timezone	Time zone.
	time	Time.

5.18 TDK_OSD_CONFIG

Table 5-18 TDK_OSD_CONFIG

Option	Instruction	
Struct description	OSD config.	
Structure	<pre>typedef struct tag_TDK_OSD_CONFIG { TDKS32 channel; TDKS8 name[TDK_CHANNEL_NAME_SIZE]; TDK_OSD_WIDGET name_osd; TDK_OSD_WIDGET time_osd; }TDK_OSD_CONFIG;</pre>	
Members	channel	Channel, start from 0.
	name[TDK_CHANNEL_NAME_SIZE]	OSD name
	name_osd	Name OSD position.
	time_osd	Time OSD position.

5.19 TDK_OSD_WIDGET

Table 5-19 TDK_OSD_WIDGET

Option	Instruction	
Struct description	OSD widget config.	
Structure	<pre>typedef struct tag_TDK_OSD_WIDGET { TDK_RECT32 pos; TDKBOOL encodeblend; }TDK_OSD_WIDGET;</pre>	
Members	pos	Object's edge distance / total length * 8192.
	encodeblend	Whether to encode.

5.20 TDK_RECT32

Table 5-20 TDK_RECT32

Option	Instruction	
Struct description	Related parameters of RECT.	
Structure	<pre>typedef struct tag_TDK_RECT32 { TDKS32 x; TDKS32 y; TDKS32 w; TDKS32 h; }TDK_RECT32, *PTDK_RECT32;</pre>	
Members	x	X value.
	y	Y value.
	w	Width.
	h	Height.

5.21 TDK_NTP_CONFIG

Table 5-21 TDK_NTP_CONFIG

Option	Instruction	
Struct description	NTP config.	
Structure	<pre>typedef struct tag_TDK_NTP_CONFIG { TDKS32 count; TDK_NTP_INFO ntpinfo[32]; TDKS32 id; TDKBOOL enable; TDKS32 port; TDKU32 update_interval; }TDK_NTP_CONFIG;</pre>	
Members	count	NTP count.
	ntpinfo[32]	NTP information.
	id	Current NTP id.
	enable	Enable or disable.
	port	NTP port.
	update_interval	Update interval, uint is minute(s).

5.22 TDK_NTP_INFO

Table 5-22 TDK_NTP_INFO

Option	Instruction	
Struct description	NTP information.	
Structure	<pre>typedef struct tag_TDK_NTP_INFO { TDKS32 id; TDKS8 name[64]; }TDK_NTP_INFO;</pre>	
Members	id	NTP id.
	name[64]	NTP name.

5.23 TDK_USER_MANAGE_CONFIG

Table 5-23 TDK_USER_MANAGE_CONFIG

Option	Instruction	
Struct description	NTP information.	
Structure	<pre>typedef struct tag_TDK_USER_MANAGE_CONFIG { TDKU32 rightnum; TDK_OPR_RIGHT rightlist[TDK_MAX_RIGHT_NUM]; TDKU32 groupnum; TDK_USER_GROUP_CONFIG grouplist[TDK_MAX_GROUP_NUM]; TDKU32 usernum; TDK_USER_CONFIG userlist[TDK_MAX_USER_NUM]; }TDK_USER_MANAGE_CONFIG;</pre>	
Members	rightnum	User right number.
	rightlist[TDK_MAX_RIGHT_NUM]	User right list.
	groupnum	User group number.
	grouplist[TDK_MAX_GROUP_NUM]	User group list.
	usernum	User number.
	userlist	User list.

5.24 TDK_OPR_RIGHT

Table 5-24 TDK_OPR_RIGHT

Option	Instruction	
Struct description	User operate right.	
Structure	<pre>typedef struct tag_TDK_OPR_RIGHT { TDKU32 id; TDKS8 name[TDK_RIGHT_NAME_LENGTH]; TDKS8 memo[TDK_MEMO_LENGTH]; }TDK_OPR_RIGHT;</pre>	
Members	id	Id.
	name[TDK_RIGHT_NAME_LENGTH]	Name.
	memo[TDK_MEMO_LENGTH]	Memo.

5.25 TDK_USER_GROUP_CONFIG

Table 5-25 TDK_USER_GROUP_INFO

Option	Instruction	
Struct description	User group config.	
Structure	<pre>typedef struct tag_TDK_USER_GROUP_INFO { TDKU32 id; TDKS8 name[TDK_GROUP_NAME_LENGTH]; TDKU32 rightnum; TDKU32 rightid[TDK_MAX_RIGHT_NUM]; char memo[TDK_MEMO_LENGTH]; }TDK_USER_GROUP_CONFIG;</pre>	
Members	id	User group id.
	name[TDK_GROUP_NAME_LENGTH]	User group name.
	rightnum	Right number.
	rightid[TDK_MAX_RIGHT_NUM]	Right id.
	memo[TDK_MEMO_LENGTH]	Memo.

5.26 TDK_USER_CONFIG

Table 5-26 TDK_USER_CONFIG

Option	Instruction	
Struct description	User config.	
Structure	typedef struct tag_TDK_USER_CONFIG { TDKU32 id; TDKU32 groupid; TDKS8 name[96]; TDKS8 passWord[96]; TDKU32 rightnum; TDKU32 rightid[TDK_MAX_RIGHT_NUM]; TDKS8 memo[TDK_MEMO_LENGTH]; TDKBOOL reuse; }TDK_USER_CONFIG;	
Members	id	User id.
	groupid	User group id.
	name[96]	Username.
	password[96]	Password.
	rightnum	User right number.
	rightid[TDK_MAX_RIGHT_NUM]	Right id.
	memo[TDK_MEMO_LENGTH]	Memo
	reuse	Whether support account reuse.

5.27 TDK_CHANNEL_STATE

Table 5-27 TDK_CHANNEL_STATE

Option	Instruction	
Struct description	Channel state.	
Structure	typedef struct tag_TDK_CHANNEL_STATE { TDK_ENUM_CHANNEL_STATE state[TDK_MAX_CHANNEL]; }TDK_CHANNEL_STATE;	
Member	state[TDK_MAX_CHANNEL]	Channel state array.

5.28 TDK_FACE_DETECTION_CONFIG

Table 5-28 TDK_FACE_DETECTION_CONFIG

Option	Instruction	
Struct description	Related parameters of face detection.	
Structure	<pre>typedef struct tag_TDK_FACE_DETECTION_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_FACE_DETECT rule[TDK_MAX_RULE]; TDK_RULE_FACE_DETECT_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_FACE_DETECTION_CONFIG;</pre>	
Members	enable	Enable or disable
	ruleNum	The number of valid TDK_RULE_FACE_DETECT and TDK_AREA
	rule[TDK_MAX_RULE]	Sensitivity.
	ruleRange	The range of sensitivity.
	area[TDK_MAX_RULE]	The area of interest.
	alarm	Alarm information.

5.29 TDK_AREA

Table 5-29 TDK_AREA

Option	Instruction	
Struct description	Related parameters of area.	
Structure	<pre>typedef struct tag_TDK_AREA { TDK_POINT32 point[10]; TDKS32 pointNum; TDKBOOL exist; }TDK_AREA;</pre>	
Members	point[10]	Points.
	pointNum	Effective number of points.
	exist	True means valid.

5.30 TDK_RULE_FACE_DETECT

Table 5-30 TDK_RULE_FACE_DETECT

Option	Instruction	
Struct description	Face detect sensitivity.	
Structure	<pre>typedef struct tag_TDK_RULE_FACE_DETECT { TDKS32 sensitiveValue; }TDK_RULE_FACE_DETECT;</pre>	
Members	sensitiveValue	Sensitivity.

5.31 TDK_RULE_FACE_DETECT_RANGE

Table 5-31 TDK_RULE_FACE_DETECT_RANGE

Option	Instruction	
Struct description	Face detect sensitivity range.	
Structure	<pre>typedef struct tag_TDK_RULE_FACE_DETECT_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_FACE_DETECT_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.32 TDK_SMART_ALARM

Table 5-32 TDK_SMART_ALARM

Option	Instruction	
Struct description	Smart alarm config.	
Structure	<pre>typedef struct tag_TDK_SMART_ALARM { TDKS32 scheduleNum; TDK_SCHEDULE schedule[TDK_DYAS_WEEK][TDK_MAX_SCHEDULE]; TDKS32 intervalValue; TDKS32 intervalMin; TDKS32 intervalMax; TDKBOOL alarmOutSupport; TDKBOOL alarmOutEnable; TDKS32 alarmOutPortNum;</pre>	

	TDKS32 alarmOutPortMask; TDKS32 alarmOutDelayValue; TDKS32 alarmOutDelayMin; TDKS32 alarmOutDelayMax; TDKBOOL showMessageSupport; TDKBOOL showMessageEnable; TDKBOOL sendEmailSupport; TDKBOOL sendEmailEnable; TDKBOOL recordChannelSupport; TDKBOOL recordChannelEnable; TDKU64 recordChannelMask; TDKS32 recordDelayValue; TDKS32 recordDelayMin; TDKS32 recordDelayMax; TDKBOOL ptzLinkSupport; TDKBOOL ptzLinkEnable; TDK_PTZ_LINK ptzLink[TDK_MAX_CHANNEL]; TDK_PTZ_LINK_RANGE ptzLinkRange[TDK_MAX_PTZ_LINK_EVENT_TYPE]; TDKBOOL smartLightSupport; TDKBOOL smartLightEnable; TDKS32 smartLightDelayValue; TDKS32 smartLightDelayMin; TDKS32 smartLightDelayMax; TDKBOOL tourSupport; TDKBOOL tourEnable; TDKU64 tourChannelMask; TDKBOOL buzzerSupport; TDKBOOL buzzerEnable; TDKS32 buzzerAlarmTimeValue; TDKS32 buzzerAlarmTimeMin; TDKS32 buzzerAlarmTimeMax; TDKBOOL snapshotSupport; TDKBOOL snapshotEnable; TDKU64 snapshotChannelMask; }TDK_SMART_ALARM;	
Members	scheduleNum	Number of time periods supported in a day, scheduleNum <= TDK_MAX_SCHEDULE.
	schedule[TDK_DYAS_WEEK][TDK_MAX_SCHEDULE]	Time periods supported in a week.
	intervalVal	Interval value, range: intervalMin ~ intervalMax.
	intervalMin	Minimum interval.

	intervalMax	Maximum Interval.
	alarmOutSupport	Whether to support alarm output.
	alarmOutEnable	Alarm output enable or not.
	alarmOutPortNum	The number of alarm output ports.
	alarmOutPortMask	Alarm output port bit mask.
	alarmOutDelayValue	Alarm output delay value, range: alarmOutDelayMin~ alarmOutDelayMax.
	alarmOutDelayMin	Minimum alarm out delay.
	alarmOutDelayMax	Maximum alarm out delay.
	showMessageSupport	Whether to support show Message.
	showMessageEnable	Show message enable or not.
	sendEmailSupport	Whether to support send email.
	sendEmailEnable	Send Email enable or not.
	recordChannelSupport	Whether to support record channel and record delay.
	recordChannelEnable	Record channel enable or not.
	recordChannelMask	Record channel bit mask.
	recordDelayValue	Record delay value, range: recordDelayMin~ recordDelayMax
	recordDelayMin	Minimum record delay.
	recordDelayMax	Maximum record delay.
	ptzLinkSupport	Whether to support PTZ link action.
	ptzLinkEnable	PTZ link action enable or not.
	ptzLink[TDK_MAX_CHANNEL]	PTZ link action data, the maximum number set at the same time is 16.
	ptzLinkRange[TDK_MAX_PTZ_LINK_EVENT_TYPE]	PTZ link action parameter range.
	smartLightSupport	Whether to support smartLight.
	smartLightEnable	SmartLight enable or not.
	smartLightDelayValue	SmartLight delay value, range: smartLightDelayMin~ smartLightDelayMax.
	smartLightDelayMin	Minimum smartlight delay.
	smartLightDelayMax	Maximum smartlight delay.
	tourSupport	Whether to support tour.
	tourEnable	Tour enable or not.
	tourChannelMask	Tour channel bit mask.
	buzzerSupport	Whether to support buzzer.

	buzzerEnable	Buzzer enable or not.
	buzzerAlarmTimeValue	Buzzer alarm time value, range: buzzerAlarmTimeMin~ buzzerAlarmTimeMax
	buzzerAlarmTimeMin	Minimum buzzer alarm time.
	buzzerAlarmTimeMax	Maximum buzzer alarm time. If buzzerAlarmTimeMax = 0, buzzer alarm time is not supported
	snapshotSupport	Whether to support snapshot.
	snapshotEnable	Snapshot enable or not.
	snapshotChannelMask	Snapshot channel bit mask.

5.33 TDK_SCHEDULE

Table 5-33 TDK_SCHEDULE

Option	Instruction	
Struct description	Related parameters of Schedule.	
Structure	<pre>typedef struct tag_TDK_SCHEDULE { TDK_SYSTEM_TIME beginTime; TDK_SYSTEM_TIME endTime; TDKBOOL enable; }TDK_SCHEDULE;</pre>	
Members	beginTime	Begin time.
	endTime	End time.
	enable	Enable or disable.

5.34 TDK_PTZ_LINK

Table 5-34 TDK_PTZ_LINK

Option	Instruction	
Struct description	PTZ link config.	
Structure	<pre>typedef struct tag_TDK_PTZ_LINK { TDKS32 eventTypeID; TDKS32 eventTypeValue; }TDK_PTZ_LINK;</pre>	
Members	eventTypeID	0: never 1: preset 2:Tour 3:pattern
	eventTypeValue	Event type id.

5.35 TDK_PTZ_LINK_RANGE

Table 5-35 TDK_PTZ_LINK_RANGE

Option	Instruction	
Struct description	Related parameters of PTZ link range.	
Structure	<pre>typedef struct tag_TDK_PTZ_LINK_RANGE { TDKS32 eventTypeID; TDKS32 valueMin; TDKS32 valueMax; }TDK_PTZ_LINK_RANGE;</pre>	
Members	eventTypeID	0: never 1: preset 2:Tour 3:pattern
	valueMin	Minimum value.
	valueMax	Maximum value.

5.36 TDK_LINE_CROSSING_CONFIG

Table 5-36 TDK_LINE_CROSSING_CONFIG

Option	Instruction	
Struct description	Related parameters of line crossing.	
Structure	<pre>typedef struct tag_TDK_LINE_CROSSING_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_LINE_CROSSING rule[TDK_MAX_RULE]; TDK_RULE_LINE_CROSSING_RANGE ruleRange; TDK_LINE line[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_LINE_CROSSING_CONFIG;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_LINE_CROSSING and TDK_LINE.
	rule[TDK_MAX_RULE]	The direction of line and the sensitivity.
	ruleRange	The range of rule parameter.
	line[TDK_MAX_RULE]	The line of interest.
	alarm	Alarm information.

5.37 TDK_LINE

Table 5-37 TDK_LINE

Option	Instruction	
Struct description	Related parameters of Line.	
Structure	<pre>typedef struct tag_TDK_LINE { TDK_POINT32 pointStart; TDK_POINT32 pointEnd; TDKBOOL exist; }TDK_LINE;</pre>	
Members	pointStart	Start position.
	pointEnd	End position.
	exist	True means valid.

5.38 TDK_RULE_LINE_CROSSING

Table 5-38 TDK_RULE_LINE_CROSSING

Option	Instruction	
Struct description	Related parameters of line crossing rule.	
Structure	<pre>typedef struct tag_TDK_RULE_LINE_CROSSING { TDKS32 direction; TDKS32 sensitiveValue; }TDK_RULE_LINE_CROSSING;</pre>	
Members	direction	The direction of line, 0: A<->B 1: A->B 2: B<-A.
	sensitiveValue	Sensitivity.

5.39 TDK_RULE_LINE_CROSSING_RANGE

Table 5-39 TDK_RULE_LINE_CROSSING_RANGE

Option	Instruction	
Struct description	Related parameters of line crossing rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_LINE_CROSSING_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_LINE_CROSSING_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.40 TDK_AREA_INTRUSION_CONFIG

Table 5-40 TDK_AREA_INTRUSION_CONFIG

Option	Instruction	
Struct description	Related parameters of area intrusion config.	
Structure	<pre>typedef struct tag_TDK_AREA_INTRUSION_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_AREA_INTRUSION rule[TDK_MAX_RULE]; TDK_RULE_AREA_INTRUSION_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_AREA_INTRUSION_CONFIG;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_AREA_INTRUSION and TDK_AREA.
	rule[TDK_MAX_RULE]	Include time threshold, sensitivity, proportion.
	ruleRange	The range of TDK_RULE_AREA_INTRUSION.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.41 TDK_RULE_AREA_INTRUSION

Table 5-41 TDK_RULE_AREA_INTRUSION

Option	Instruction	
Struct description	Related parameters of area intrusion rule.	
Structure	<pre>typedef struct tag_TDK_RULE_AREA_INTRUSION { TDKS32 timeThresholdValue; TDKS32 sensitiveValue; TDKS32 percentValue; }TDK_RULE_AREA_INTRUSION;</pre>	
Members	timeThresholdValue	The value of time threshold.
	sensitiveValue	Sensitivity.
	percentValue	Proportion.

5.42 TDK_RULE_AREA_INTRUSION_RANGE

Table 5-42 TDK_RULE_AREA_INTRUSION_RANGE

Option	Instruction	
Struct description	Related parameters of area intrusion rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_AREA_INTRUSION_RANGE { TDKS32 timeThresholdMin; TDKS32 timeThresholdMax; TDKS32 sensitiveMin; TDKS32 sensitiveMax; TDKS32 percentMin; TDKS32 percentMax; }TDK_RULE_AREA_INTRUSION_RANGE;</pre>	
Members	timeThresholdMin	Minimum value of time threshold.
	timeThresholdMax	Maximum value of time threshold.
	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.
	percentMin	Minimum proportion.
	percentMax	Maximum proportion.

5.43 TDK_UNATTENDED_OBJECT_CONFIG

Table 5-43 TDK_UNATTENDED_OBJECT_CONFIG

Option	Instruction	
Struct description	Unattended object config.	
Structure	<pre>typedef struct tag_TDK_UNATTENDED_OBJECT_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_UNATTENDED_OBJECT rule[TDK_MAX_RULE]; TDK_RULE_UNATTENDED_OBJECT_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_UNATTENDED_OBJECT_CONFIG;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_UNATTENDED_OBJECT and TDK_AREA.
	rule[TDK_MAX_RULE]	Time threshold and sensitivity.
	ruleRange	The range of TDK_RULE_UNATTENDED_OBJECT.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information

5.44 TDK_RULE_UNATTENDED_OBJECT

Table 5-44 TDK_RULE_UNATTENDED_OBJECT

Option	Instruction	
Struct description	Related parameters of unattended object rule.	
Structure	<pre>typedef struct tag_TDK_RULE_UNATTENDED_OBJECT { TDKS32 timeThresholdValue; TDKS32 sensitiveValue; }TDK_RULE_UNATTENDED_OBJECT;</pre>	
Members	timeThresholdValue	The value of time threshold.
	sensitiveValue	Sensitivity.

5.45 TDK_RULE_UNATTENDED_OBJECT_RANGE

Table 5-45 TDK_RULE_UNATTENDED_OBJECT_RANGE

Option	Instruction	
Struct description	Related parameters of unattended object rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_UNATTENDED_OBJECT_RANGE { TDKS32 timeThresholdMin; TDKS32 timeThresholdMax; TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_UNATTENDED_OBJECT_RANGE;</pre>	
Members	timeThresholdMin	Minimum value of time threshold.
	timeThresholdMax	Maximum value of time threshold.
	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.46 TDK_OBJECT_MISSING_CONFIG

Table 5-46 TDK_OBJECT_MISSING_CONFIG

Option	Instruction	
Struct description	Object missing config.	
Structure	<pre>typedef struct tag_TDK_OBJECT_MISSING_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_OBJECT_MISSING rule[TDK_MAX_RULE]; TDK_RULE_OBJECT_MISSING_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_OBJECT_MISSING_CONFIG;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_OBJECT_MISSING and TDK_AREA.
	rule[TDK_MAX_RULE]	Time threshold and sensitivity.
	ruleRange	The range of TDK_RULE_OBJECT_MISSING.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.47 TDK_RULE_OBJECT_MISSING

Table 5-47 TDK_RULE_OBJECT_MISSING

Option	Instruction	
Struct description	Related parameters of object missing rule.	
Structure	<pre>typedef struct tag_TDK_RULE_OBJECT_MISSING { TDKS32 timeThresholdValue; TDKS32 sensitiveValue; }TDK_RULE_OBJECT_MISSING;</pre>	
Members	timeThresholdValue	The value of time threshold.
	sensitiveValue	Sensitivity.

5.48 TDK_RULE_OBJECT_MISSING_RANGE

Table 5-48 TDK_RULE_OBJECT_MISSING_RANGE

Option	Instruction	
Struct description	Related parameters of object missing rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_OBJECT_MISSING_RANGE { TDKS32 timeThresholdMin; TDKS32 timeThresholdMax; TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_OBJECT_MISSING_RANGE;</pre>	
Members	timeThresholdMin	Minimum value of time threshold.
	timeThresholdMax	Maximum value of time threshold.
	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.49 TDK_REGION_ENTRANCE_CONFIG

Table 5-49 TDK_REGION_ENTRANCE_CONFIG

Option	Instruction	
Struct description	Region entrance config.	
Structure	<pre>typedef struct tag_TDK_REGION_ENTRANCE_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_REGION_ENTRANCE rule[TDK_MAX_RULE]; TDK_RULE_REGION_ENTRANCE_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_REGION_ENTRANCE_CONFIG;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_REGION_ENTRANCE and TDK_AREA.
	rule[TDK_MAX_RULE]	Time threshold and sensitivity.
	ruleRange	The range of TDK_RULE_REGION_ENTRANCE.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.50 TDK_RULE_REGION_ENTRANCE

Table 5-50 TDK_RULE_REGION_ENTRANCE

Option	Instruction	
Struct description	Related parameters of region entrance rule.	
Structure	<pre>typedef struct tag_TDK_RULE_REGION_ENTRANCE { TDKS32 sensitiveValue; }TDK_RULE_REGION_ENTRANCE;</pre>	
Members	sensitiveValue	Sensitivity.

5.51 TDK_RULE_REGION_ENTRANCE_RANGE

Table 5-51 TDK_RULE_REGION_ENTRANCE_RANGE

Option	Instruction	
Struct description	Related parameters of region entrance rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_REGION_ENTRANCE_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_REGION_ENTRANCE_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.52 TDK_REGION_EXITING_CONFIG

Table 5-52 TDK_REGION_EXITING_CONFIG

Option	Instruction	
Struct description	Region exiting config.	
Structure	<pre>typedef struct tag_TDK_REGION_EXITING_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_REGION_EXITING rule[TDK_MAX_RULE]; TDK_RULE_REGION_EXITING_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_REGION_EXITING_CONFIG;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_REGION_EXITING and TDK_AREA.
	rule[TDK_MAX_RULE]	Time threshold and sensitivity.
	ruleRange	The range value of TDK_RULE_REGION_EXITING.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.53 TDK_RULE_REGION_EXITING

Table 5-53 TDK_RULE_REGION_EXITING

Option	Instruction	
Struct description	Related parameters of region exiting rule.	
Structure	<pre>typedef struct tag_TDK_RULE_REGION_EXITING { TDKS32 sensitiveValue; }TDK_RULE_REGION_EXITING;</pre>	
Members	sensitiveValue	Sensitivity.

5.54 TDK_RULE_REGION_EXITING_RANGE

Table 5-54 TDK_RULE_EXITING_ENTRANCE_RANGE

Option	Instruction	
Struct description	Related parameters of region exiting rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_EXITING_ENTRANCE_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_REGION_EXITING_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.55 TDK_FAST_MOVING_CONFIG

Table 5-55 TDK_FAST_MOVING_CONFIG

Option	Instruction	
Struct description	Fast moving config.	
Structure	<pre>typedef struct tag_TDK_FAST_MOVING_CONFIG { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_FAST_MOVING rule[TDK_MAX_RULE]; TDK_RULE_FAST_MOVING_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_FAST_MOVING_CONFIG;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_FAST_MOVING and TDK_AREA.
	rule[TDK_MAX_RULE]	Include sensitivity.
	ruleRange	The range value of TDK_RULE_FAST_MOVING.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.56 TDK_RULE_FAST_MOVING

Table 5-56 TDK_RULE_FAST_MOVING

Option	Instruction	
Struct description	Related parameters of fast moving rule.	
Structure	<pre>typedef struct tag_TDK_RULE_FAST_MOVING { TDKS32 sensitiveValue; }TDK_RULE_FAST_MOVING;</pre>	
Members	sensitiveValue	Sensitivity.

5.57 TDK_RULE_FAST_MOVING_RANGE

Table 5-57 TDK_RULE_FAST_MOVING_RANGE

Option	Instruction	
Struct description	Related parameters of fast moving rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_FAST_MOVING_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_FAST_MOVING_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.58 TDK_LOITERING_DETECTION

Table 5-58 TDK_LOITERING_DETECTION

Option	Instruction	
Struct description	Loitering detection config.	
Structure	<pre>typedef struct tag_TDK_LOITERING_DETECTION { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_LOITERING_DETECTION rule[TDK_MAX_RULE]; TDK_RULE_LOITERING_DETECTION_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_LOITERING_DETECTION;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_LOITERING_DETECTION and TDK_AREA.
	rule[TDK_MAX_RULE]	Include time threshold, sensitivity and config mask.
	ruleRange	The range of TDK_RULE_LOITERING_DETECTION.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.59 TDK_RULE_LOITERING_DETECTION

Table 5-59 TDK_RULE_LOITERING_DETECTION

Option	Instruction	
Struct description	Related parameters of loitering detection rule.	
Structure	<pre>typedef struct tag_TDK_RULE_LOITERING_DETECTION { TDKS32 timeThresholdValue; TDKS32 sensitiveValue; TDKU8 mask; }TDK_RULE_LOITERING_DETECTION;</pre>	
Members	timeThresholdValue	The value of time threshold.
	sensitiveValue	Sensitivity.
	mask	Detection rule enable or not, bit-mask, 0-diasble, 1-enable. bit: 0 : Journey 1: Turn round 2 : Offset.

5.60 TDK_RULE_LOITERING_DETTECTION_RANGE

Table 5-60 TDK_RULE_LOITERING_DETECTION_RANGE

Option	Instruction	
Struct description	Related parameters of loitering dettection rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_LOITERING_DETECTION_RANGE { TDKS32 timeThresholdMin; TDKS32 timeThresholdMax; TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_LOITERING_DETECTION_RANGE;</pre>	
Members	timeThresholdMin	Minimum value of time threshold.
	timeThresholdMax	Maximum value of time threshold.
	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.61 TDK_PEOPLE_GATHERING_DETECTION

Table 5-61 TDK_PEOPLE_GATHERING_DETECTION

Option	Instruction	
Struct description	People gathering defection config.	
Structure	<pre>typedef struct tag_TDK_PEOPLE_GATHERING_DETECTION { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_PEOPLE_GATHERING_DETECTION rule[TDK_MAX_RULE]; TDK_RULE_PEOPLE_GATHERING_DETECTION_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_PEOPLE_GATHERING_DETECTION;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_PEOPLE_GATHERING_DETECTION and TDK_AREA.
	rule[TDK_MAX_RULE]	Percent and sensitivity.
	ruleRange	The range value of TDK_RULE_PEOPLE_GATHERING_DETECTION.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.62 TDK_RULE_PEOPLE_GATHERING_DETECTION

Table 5-62 TDK_RULE_PEOPLE_GATHERING_DETECTION

Option	Instruction	
Struct description	Related parameters of people gathering detection rule.	
Structure	<pre>typedef struct tag_TDK_RULE_PEOPLE_GATHERING_DETECTION { TDKS32 percentValue; TDKS32 sensitiveValue; }TDK_RULE_PEOPLE_GATHERING_DETECTION;</pre>	
Members	percentValue	Proportion.
	sensitiveValue	Sensitivity.

5.63 TDK_RULE_PEOPLE_GATHERING_DETECTION_RANGE

Table 5-63 TDK_RULE_PEOPLE_GATHERING_DETECTION_RANGE

Option	Instruction	
Struct description	Related parameters of people gathering detection rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_PEOPLE_GATHERING_DETECTION_RANGE { TDKS32 percentMin; TDKS32 percentMax; TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_PEOPLE_GATHERING_DETECTION_RANGE;</pre>	
Members	percentMin	Minimum proportion.
	percentMax	Maximum proportion.
	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.64 TDK_PARKING_DETECTION

Table 5-64 TDK_PARKING_DETECTION

Option	Instruction	
Struct description	Parking detection config.	
Structure	<pre>typedef struct tag_TDK_PARKING_DETECTION { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_PARKING_DETECTION rule[TDK_MAX_RULE]; TDK_RULE_PARKING_DETECTION_RANGE ruleRange; TDK_AREA area[TDK_MAX_RULE]; TDK_SMART_ALARM alarm; }TDK_PARKING_DETECTION;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_PARKING_DETECTION and TDK_AREA.
	rule[TDK_MAX_RULE]	Include time threshold and sensitivity.
	ruleRange	The range of TDK_RULE_PARKING_DETECTION.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.65 TDK_RULE_PARKING_DETECTION

Table 5-65 TDK_RULE_PARKING_DETECTION

Option	Instruction	
Struct description	Related parameters of parking detection rule.	
Structure	<pre>typedef struct tag_TDK_RULE_PARKING_DETECTION { TDKS32 timeThresholdValue; TDKS32 sensitiveValue; }TDK_RULE_PARKING_DETECTION;</pre>	
Members	timeThresholdValue	The time threshold value.
	sensitiveValue	Sensitivity.

5.66 TDK_RULE_PARKING_DETECTION_RANGE

Table 5-66 TDK_RULE_LOITERING_DETECTION_RANGE

Option	Instruction	
Struct description	Related parameters of parking detection rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_LOITERING_DETECTION_RANGE { TDKS32 timeThresholdMin; TDKS32 timeThresholdMax; TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_LOITERING_DETECTION_RANGE;</pre>	
Members	timeThresholdMin	Minimum value of time threshold.
	timeThresholdMax	Maximum value of time threshold.
	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.67 TDK_SCENE_CHANGE_DETECTION

Table 5-67 TDK_SCENE_CHANGE_DETECTION

Option	Instruction	
Struct description	Related parameters of scene change detection config.	
Structure	<pre>typedef struct tag_TDK_SCENE_CHANGE_DETECTION { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_SCENE_CHANGE_DETECTION rule[TDK_MAX_RULE]; TDK_RULE_SCENE_CHANGE_DETECTION_RANGE ruleRange; TDK_SMART_ALARM alarm; }TDK_SCENE_CHANGE_DETECTION;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_SCENE_CHANGE_DETECTION .
	rule[TDK_MAX_RULE]	Include sensitivity.
	ruleRange	The range of TDK_RULE_SCENE_CHANNGE_DETECTION.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.68 TDK_RULE_SCENE_CHANGE_DETECTION

Table 5-68 TDK_RULE_SCENE_CHANGE_DETECTION

Option	Instruction	
Struct description	Related parameters of scene change detection rule.	
Structure	<pre>typedef struct tag_TDK_RULE_SCENE_CHANGE_DETECTION { TDKS32 sensitiveValue; }TDK_RULE_SCENE_CHANGE_DETECTION;</pre>	
Members	sensitiveValue	Sensitivity.

5.69 TDK_RULE_SCENE_CHANGE_DETECTION_RANGE

Table 5-69 TDK_RULE_SCENE_CHANGE_DETECTION_RANGE

Option	Instruction	
Struct description	Related parameters of scene change detection rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_SCENE_CHANGE_DETECTION_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_SCENE_CHANGE_DETECTION_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.70 TDK_BLURRED_DETECTION

Table 5-70 TDK_BLURRED_DETECTION

Option	Instruction	
Struct description	Blurred detection config.	
Structure	<pre>typedef struct tag_TDK_BLURRED_DETECTION { TDKBOOL enable; TDKS32 ruleNum; TDK_RULE_BLURRED_DETECTION rule[TDK_MAX_RULE]; TDK_RULE_BLURRED_DETECTION_RANGE ruleRange; TDK_SMART_ALARM alarm; }TDK_BLURRED_DETECTION;</pre>	
Members	enable	Enable or not.
	ruleNum	The number of valid TDK_RULE_BLURRED_DETECTION.
	rule[TDK_MAX_RULE]	Sensitivity.
	ruleRange	The range of TDK_RULE_BLURRED_DETECTION.
	area[TDK_MAX_RULE]	The area of intrest.
	alarm	Alarm information.

5.71 TDK_RULE_BLURRED_DETECTION

Table 5-71 TDK_RULE_BLURRED_DETECTION

Option	Instruction	
Struct description	Related parameters of blurred detection rule.	
Structure	<pre>typedef struct tag_TDK_RULE_BLURRED_DETECTION { TDKS32 sensitiveValue; }TDK_RULE_BLURRED_DETECTION;</pre>	
Members	sensitiveValue	Sensitivity.

5.72 TDK_RULE_BLURRED_DETECTION_RANGE

Table 5-72 TDK_RULE_BLURRED_DETECTION_RANGE

Option	Instruction	
Struct description	Related parameters of blurred detection rule range.	
Structure	<pre>typedef struct tag_TDK_RULE_BLURRED_DETECTION_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_RULE_BLURRED_DETECTION_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.73 TDK_AUDIO_EXCEPTION_DETECTION

Table 5-73 TDK_AUDIO_EXCEPTION_DETECTION

Option	Instruction	
Struct description	Audio exception detection config.	
Structure	<pre>typedef struct tag_TDK_AUDIO_EXCEPTION_DETECTION { TDKBOOL abnormalAudioInputEnable; TDKBOOL strongSoundIntensityEnable; TDKBOOL soundIntensityDroppedSharplyEnable; TDK_STRONG_SOUND_INTENSITY strongSoundIntensity; TDK_STRONG_SOUND_INTENSITY_RANGE strongSoundIntensityRange; TDK_SOUND_INTENSITY_DROPPED_SHARPLY soundIntensityDropped; TDK_SOUND_INTENSITY_DROPPED_SHARPLY_RANGE soundIntensityDroppedRange; TDK_SMART_ALARM alarm; }TDK_AUDIO_EXCEPTION_DETECTION;</pre>	
Members	abnormalAudioInputEnable	Audio input exception enable or not.
	strongSoundIntensityEnable	Sound rise exception enable or not.
	soundIntensityDroppedSharplyEnable	Sound drop exception enable or not.
	strongSoundIntensity	Sound rise exception config.
	strongSoundIntensityRange	The range of TDK_STRONG_SOUND_INTENSITY.
	soundIntensityDropped	Sound drop exception config.
	soundIntensityDroppedRange	The range of TDK_SOUND_INTENSITY_DROPPED.
	alarm	Alarm information.

5.74 TDK_STRONG_SOUND_INTENSITY

Table 5-74 TDK_STRONG_SOUND_INTENSITY

Option	Instruction	
Struct description	Related parameters of strong sound intensity.	
Structure	<pre>typedef struct tag_TDK_STRONG_SOUND_INTENSITY { TDKS32 sensitiveValue; TDKS32 thresholdValue; }TDK_STRONG_SOUND_INTENSITY;</pre>	
Members	sensitiveValue	Sensitivity.
	thresholdValue	Time threshold.

5.75 TDK_STRONG_SOUND_INTENSITY_RANGE

Table 5-75 TDK_STRONG_SOUND_INTENSITY_RANGE

Option	Instruction	
Struct description	The range of strong sound intensity.	
Structure	<pre>typedef struct tag_TDK_STRONG_SOUND_INTENSITY_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; TDKS32 thresholdMin; TDKS32 thresholdMax; }TDK_STRONG_SOUND_INTENSITY_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.
	thresholdMin	Minimum value of time threshold.
	thresholdMax	Maximum value of time threshold.

5.76 TDK_SOUND_INTENSITY_DROPPED_SHARPLY

Table 5-76 TDK_SOUND_INTENSITY_DROPPED_SHARPLY

Option	Instruction	
Struct description	The intensity of the sound dropped sharply.	
Structure	<pre>typedef struct tag_TDK_SOUND_INTENSITY_DROPPED_SHARPLY { TDKS32 sensitiveValue; }TDK_SOUND_INTENSITY_DROPPED_SHARPLY;</pre>	
Members	sensitiveValue	Sensitivity.

5.77 TDK_SOUND_INTENSITY_DROPPED_SHARPLY_RANGE

Table 5-77 TDK_SOUND_INTENSITY_DROPPED_SHARPLY_RANGE

Option	Instruction	
Struct description	The range of sound drop intensity.	
Structure	<pre>typedef struct tag_TDK_SOUND_INTENSITY_DROPPED_SHARPLY_RANGE { TDKS32 sensitiveMin; TDKS32 sensitiveMax; }TDK_SOUND_INTENSITY_DROPPED_SHARPLY_RANGE;</pre>	
Members	sensitiveMin	Minimum sensitivity.
	sensitiveMax	Maximum sensitivity.

5.78 TDK_CROSSING_LINE_STATISTICS

Table 5-78 TDK_CROSSING_LINE_STATISTICS

Option	Instruction	
Struct description	Crossing line statistics config.	
Structure	<pre>typedef struct tag_TDK_CROSSING_LINE_STATISTICS { TDKBOOL enable; TDKBOOL osdOverlayEnable; TDK_POINT32 osdPositon; TDKS32 ruleNum; TDK_RULE_CROSSING_LINE_STATISTICS line[TDK_MAX_RULE]; TDKBOOL zeroFlag; TDKS32 scheduleNum; TDK_SCHEDULE schedule[TDK_DYAS_WEEK][TDK_MAX_SCHEDULE]; }TDK_CROSSING_LINE_STATISTICS;</pre>	
Members	enable	Enable or not.
	osdOverlayEnable	OSD enable or not.
	osdPosition	The upper-left coordinate of osd.
	ruleNum	The number of valid TDK_RULE_CROSSING_LINE_STATISTICS.
	zeroFlag	If zeroFlag == 1, the crossing line statistics revert to 0.
	scheduleNum	Number of time periods supported in a day, scheduleNum <= TDK_MAX_SCHEDULE
	schedule[TDK_DYAS_WEEK][TDK_MAX_SCHEDULE]	Week schedule.

5.79 TDK_RULE_CROSSING_LINE_STATISTICS

Table 5-79 TDK_RULE_CROSSING_LINE_STATISTICS

Option	Instruction	
Struct description	Related parameters of crossing line statistic rule.	
Structure	<pre>typedef struct tag_TDK_RULE_CROSSING_LINE_STATISTICS { TDKS32 index; TDK_POINT32 pointStart; TDK_POINT32 pointEnd; TDKS32 directionMask; TDKS32 direction; TDKBOOL exist; }TDK_RULE_CROSSING_LINE_STATISTICS;</pre>	
Members	index	Line index.
	pointStart	Start position.
	pointEnd	End position.
	directionMask	Direction bit mask supported.
	direction	1:A->B 2:A<-B 0:A<=>B
	exist	Whether exist.

5.80 TDK_CROSSING_LINE_STATISTICS_REPORT

Table 5-80 TDK_CROSSING_LINE_STATISTICS_REPORT

Option	Instruction	
Struct description	Related crossing line statistics report.	
Structure	<pre>typedef struct tag_TDK_CROSSING_LINE_STATISTICS_REPORT { TDKU32 count; ENUM_TDK_CROSSING_LINE_STATISTICS_REPORT type; TDK_CROSSING_LINE_STATISTICS_DATA data[TDK_MAX_CROSSING_LINE_STATISTICS_REPORT]; }TDK_CROSSING_LINE_STATISTICS_REPORT;</pre>	
Members	count	The count of the report.
	type	Report type.
	data[TDK_MAX_CROSSING_LINE_STATISTICS_REPORT]	If type is daily report, data[0] : 0:00 ~ 1:00 , data[1]: 1:00 ~ 2:00 If type is weekly report, data[0]: Sun. , data[i]: Mon. If type is monthly report, data[0]:the first day, data[1]:the second day If type is annual report, data[0]:January, data[1]:February

5.81 TDK_CROSSING_LINE_STATISTICS_DATA

Table 5-81 TDK_CROSSING_LINE_STATISTICS_DATA

Option	Instruction	
Struct description	Related parameters of crossing line statistics data.	
Structure	<pre>typedef struct tag_TDK_CROSSING_LINE_STATISTICS_DATA { TDKU32 in; TDKU32 out; }TDK_CROSSING_LINE_STATISTICS_DATA;</pre>	
Members	in	Number of people entering.
	out	Number of people leaving.

5.82 TDK_TCPIP

Table 5-82 TDK_TCPIP

Option	Instruction	
Struct description	Related parameters of device TCP/IP config.	
Structure	<pre>typedef struct tag_TDK_TCPIP { TDKS8 ip[TDK_MAX_IPADDR_LEN]; TDKU16 media_port; TDKU16 http_port; TDKS8 submask[TDK_MAX_IPADDR_LEN]; TDKS8 gateway[TDK_MAX_IPADDR_LEN]; TDKS8 major_dns[TDK_MAX_IPADDR_LEN]; TDKS8 minor_dns[TDK_MAX_IPADDR_LEN]; TDKBOOL dhcp; }TDK_TCPIP;</pre>	
Members	ip[TDK_MAX_IPADDR_LEN]	IP address.
	media_port	Media port.
	http_port	Http port.
	submask[TDK_MAX_IPADDR_LEN]	Sub net mask.
	gateway[TDK_MAX_IPADDR_LEN]	Gateway.
	major_dns[TDK_MAX_IPADDR_LEN]	Major DNS.
	minor_dns[TDK_MAX_IPADDR_LEN]	Minor DNS, cannot be the same as major DNS.
	dhcp	1- DHCP valid. 2- DHCP invalid.

5.83 TDK_NETWORK

Table 5-83 TDK_NETWORK

Option	Instruction	
Struct description	Related parameters of network config.	
Structure	<pre>typedef struct tag_TDK_NETWORK { TDKS8 ip[TDK_MAX_IPADDR_LEN]; TDKS8 networkcardip[TDK_MAX_IPADDR_LEN]; TDKS8 mac_addr[TDK_MACADDR_LEN]; TDKS8 submask[TDK_MAX_IPADDR_LEN]; TDKS8 gateway[TDK_MAX_IPADDR_LEN]; TDKS8 major_dns[TDK_MAX_IPADDR_LEN]; TDKS8 minor_dns[TDK_MAX_IPADDR_LEN]; TDKBOOL dhcp; }TDK_NETWORK;</pre>	
Members	ip[TDK_MAX_IPADDR_LEN]	IP address.
	networkcardip[TDK_MAX_IPADDR_LEN]	Broadcast card ip address.
	mac_addr[TDK_MACADDR_LEN]	Mac address.
	submask[TDK_MAX_IPADDR_LEN]	Sub net mask.
	gateway[TDK_MAX_IPADDR_LEN]	Gateway.
	major_dns[TDK_MAX_IPADDR_LEN]	Major DNS.
	minor_dns[TDK_MAX_IPADDR_LEN]	Minor DNS, cannot be the same as major DNS
	dhcp	1- DHCP valid. 2- DHCP invalid.

5.84 TDK_ALARM_IN

Table 5-84 TDK_ALARM_IN

Option	Instruction	
Struct description	Related parameters of Alarm In.	
Structure	<pre>typedef struct tag_TDK_ALARM_IN { TDKBOOL enable; TDKBOOL portNameSupport; TDKS8 portName[64]; ENUM TDK_ALARM_IN_TYPE type; TDK_SMART_ALARM alarm; }TDK_ALARM_IN;</pre>	
Members	enable	Enable or not.
	portNameSupport	Whether support to set port name.
	portName[64]	Port name (utf-8 encoding).
	type	Alarm In type (normally opened-1, normally closed-0).
	alarm	Alarm information.

5.85 TDK_ALARM_OUT

Table 5-85 TDK_ALARM_OUT

Option	Instruction	
Struct description	Related parameters of Alarm Out.	
Structure	<pre>typedef struct tag_TDK_ALARM_OUT { TDKBOOL portNameSupport; TDKS8 portName[64]; ENUM TDK_ALARM_OUT_TYPE type; ENUM TDK_ALARM_OUT_STATUS status; }TDK_ALARM_OUT;</pre>	
Members	portNameSupport	Whether support to set port name.
	portName[64]	Port name (utf-8 encoding).
	type	Alarm out type (0-stop, 1-manual, 2-schedule).
	status	Alarm out status(0-off, 1-on).

5.86 TDK_PTZ_PARAM

Table 5-86 TDK_PTZ_PARAM

Option	Instruction	
Struct description	Related parameters of PTZ control.	
Structure	<pre>typedef struct tag_TDK_PTZ_PARAM { void* param1; //reserve void* param2; //reserve void* param3; //reserve }TDK_PTZ_PARAM;</pre>	
Members	param1	Reserve param.
	param2	Reserve param.
	param3	Reserve param.

5.87 TDK_SYS_RECORDSTATUS

Table 5-87 TDK_SYS_RECORDSTATUS

Option	Instruction	
Struct description	Related parameters of system record status.	
Structure	<pre>typedef struct tag_TDK_SYS_RECORDSTATUS { TDKS32 chn; TDKS32 status; TDKS32 stream; TDKU32 fps; TDKU32 bitrate; TDKS32 redundant; TDKS8 resolution[128]; }TDK_SYS_RECORDSTATUS;</pre>	
Members	chn	Channel number, start from 0.
	status	Record status. 0- Unknown. 1- Open. 2- Close.
	stream	Data type. 0- Unknown. 1- Video. 2- Video and audio.
	fps	Frames per second.
	bitrate	Bitrate value.
	redundant	Redundancy or not.
	resolution	Resolution of stream.

5.88 TDK_SYS_LOGINFOCFG

Table 5-88 TDK_SYS_LOGINFOCFG

Option	Instruction	
Struct description	System login information config.	
Structure	<pre>typedef struct tag_TDK_SYS_LOGINFOCFG { int support_clearall; }TDK_SYS_LOGINFOCFG;</pre>	
Members	support_clearall	Whether support to clear all login information.

5.89 TDK_SYS_LOGININFOQUERY

Table 5-89 TDK_SYS_LOGININFOQUERY

Option	Instruction	
Struct description	Related parameters of PTZ control.	
Structure	<pre>typedef struct tag_TDK_SYS_LOGININFOQUERY { TDKU32 searchtype; TDK_SYSTEM_TIME stime; TDK_SYSTEM_TIME etime; }TDK_SYS_LOGININFOQUERY;</pre>	
Members	searchtype	Search log type. (ALL-0, System-1, Config-2, Storage-3, Alaram-4, Record-5, Account-6, Playback-7, Modifylog-8, TypeCount-9)
	stime	Start time.
	etime	End time.

5.90 TDK_ENC_CHNCFG

Table 5-90 TDK_ENC_CHNCFG

Option	Instruction	
Struct description	Channel encode config.	
Structure	<pre>typedef struct tag_TDK_ENC_CHNCFG { int chn; int sup; TDKHANDLE pCode; TDK_ENC_CFG streamCFG[4]; }TDK_ENC_CHNCFG;</pre>	
Members	chn	Channel id, start from 0.
	sup	Support stream type, bit mask.
	pCode	Handle of encode config object.
	streamCFG[4]	Encode config data and capabilities.

5.91 TDK_ENC_CFG

Table 5-91 TDK_ENC_CFG

Option	Instruction
Struct description	Stream config data and capability.
Structure	<pre> typedef struct tag_TDK_ENC_CFG { TDKU32 type; struct { TDKU32 val; TDKU32 sup; } cmp; struct { TDKU32 val; TDKU32 h265_sup; TDKU32 h264_sup; TDKU32 setting_sup; TDKU32 complex_sup; } complex; struct { TDKS32 video; TDKS32 audio; TDKU32 audio_sup; } AVido; struct { TDKS32 h264plus; TDKS32 h264plus_sup; } h264plus; TDKU32 imageQ; struct { TDKS32 alarm_MAXfps; TDKS32 MAXfps; TDKS32 cur; } fps; struct { </pre>

	<pre> TDKS32 min; TDKS32 max; TDKS32 cur; }lframeGop; struct { TDKS32 type; TDKS32 cur; TDKS32 setting_sup; TDKS32 cus_sup; TDKS32 cus_min[6]; TDKS32 cus_max[6]; TDKS32 list[6]; }br; TDK_ENC_RES resolution; }TDK_ENC_CFG;</pre>	
Members	type	Stream type. 1- Main stream. 2- Second stream. 3- Third stream. 4- Alarm stream.
	com.val	Compression type. 5- MJPG. 7- H264. 8- H265.
	com.sup	Support encode type, bit mask. bit 7-H264, bit 8-H265.
	cmplex.val	Encode complexity. 0- Base profile. 1- Main profile. 2- High profile.
	cmplex.h256_sup	Support h256 encode complexity.
	cmplex.h264_sup	Support h264 encode complexity.
	cmplex.setting_sup	Whether complexity can be changed.
	cmplex.cmplex_sup	Whether support complexity. .
	AVido.video	Video encode enable or not.
	AVido.audio	Audio encode enable or not.
	AVido.audio_sup	Whether support audio encode.
	h264plus.h264plus	H264 plus enable or not.
	h264plus.h264plus_sup	Whether support h264 plus.
	imageQ	Compression quality. 0- Lowest.

		1- General. 2- Normal. 3- Good. 4- Better. 5- Best.
	fps.alarm_MAXfps	Max Fps of alarm stream in second stream and third stream.
	fps.MAXfps	Max FPS.
	fps.cur	Current FPS.
	IframeGop.min	Minimum GOP.
	IframeGop.max	Maximum GOP.
	IframeGop.cur	Current GOP.
	br.type	Current bitrate type. 0-CBR, 1-VBR.
	br.cur	Current bitrate value.
	br.setting_sup	Whether support set bitrate type.
	br.cus_sup	Whether support custom bitrate.
	br.cus_min[6]	Minimum custom bitrate.
	br.cus_max[6]	Maximum custom bitrate.
	br.list[6]	Bitrate value list.
	resolution	Stream resolution options.

5.92 TDK_ENC_RES

Table 5-92 TDK_ENC_RES

Option	Instruction	
Struct description	Resolution information.	
Structure	<pre>typedef struct tag_TDK_ENC_RES { TDK_RULE_FAST_MOVING_RANGE* pSize; TDKS32 count; TDKS32 idx; }TDK_ENC_RES;</pre>	
Members	pSize	The resolution array.
	count	The count of resolution.
	idx	The index of resolution.

5.93 TDK_ENC_QUERYBITRATE

Table 5-93 TDK_ENC_QUERYBITRATE

Option	Instruction	
Struct description	Query bitrate config.	
Structure	<pre>typedef struct tag_TDK_ENC_QUERYBITRATE { TDKHANDLE pCode; TDKS32 chn; TDKS32 stream; TDK_RULE_FAST_MOVING_RANGE main_res; TDKS32 main_fps; TDKS32 bt_cus; //result TDKS32 cus_min[6]; TDKS32 cus_max[6]; TDKS32 list[6]; }TDK_ENC_QUERYBITRATE;</pre>	
Members	pCode	Encode config handle.
	chn	Channel id, start from 0.
	stream	Stream type.
	main_res	Main resolution.
	main_fps	Main FPS.
	bt_cus	Whether it is custom type. 0- Not custom, result in list[6]. 1- Custom, result in cus_min[6] and cus_max[6].
	cus_min[6]	Minimum custom bitrate value
	cus_max[6]	Maximum custom bitrate value.
	list[6]	Bitrate value list.

5.94 TDKENCONDCFGDATA

Table 5-94 TDKENCONDCFGDATA

Option	Instruction	
Struct description	Encode config data.	
Structure	<pre>typedef struct TDKENCONDCFGDATA { INT32 chn; TDKHANDLE pCode; TDKSTREAMDATA main_stream; TDKSTREAMDATA sub_stream; }TDKENCONDCFGDATA;</pre>	
Members	chn	Channel id, start from 0.
	pCode	Encode config handle.
	main_stream	Main stream config data.
	sub_stream	Sub stream config data.

5.95 TDKSTREAMDATA

Table 5-95 TDKSTREAMDATA

Option	Instruction	
Struct description	Related parameters of stream data.	
Structure	<pre>typedef struct tag_TDKSTREAMDATA { int type; int cmp; int enc_cap; int video_checked; int avdio_checked; TDK_RULE_FAST_MOVING_RANGE res; int fps; int bit_type; int imageQ; int br; int iframe; int h264p; }TDKSTREAMDATA;</pre>	
Members	type	Stream type.
	cmp	Current compression type.
	enc_cap	Encode profile.
	video_checked	Video enable or not.
	avdio_checked	Avdio enable or not.
	res	Current resolution.
	fps	Current fps.
	bit_type	Capture bit type. 0- CAPTURE_BRC_CBR 1- CAPTURE_BRC_VBR 2- CAPTURE_BRC_MBR 3- CAPTURE_BRC_NR
	imageQ	Compression quality.
	br	Stream bitrate.
	iframe	Stream GOP.
	h264p	Whether support h246 plus.

5.96 TDK_ENC_QUERYRES

Table 5-96 TDK_ENC_QUERYRES

Option	Instruction	
Struct description	Query resolution.	
Structure	<pre>typedef struct tag_TDK_ENC_QUERYRES { TDKHANDLE pCode; int chn; int dest_stream; int main_stream; TDK_RULE_FAST_MOVING_RANGE main_res; TDK_RULE_FAST_MOVING_RANGE* res; TDKS32 count; TDKS32 cur_index; }TDK_ENC_QUERYRES;</pre>	
Members	pCode	Code config handle.
	chn	Channel.
	dest_stream	Requested stream.
	main_stream	Main stream or alarm stream. 0- Alarm stream. 1- Main stream.
	main_res	Main stream resolution or alarm stream resolution.
	res	Resolution result array.
	count	Result count.
	cur_index	Current result index.

5.97 TDK_ENC_COVER

Table 5-97 TDK_ENC_COVER

Option	Instruction	
Struct description	Cover config.	
Structure	<pre>typedef struct { int chn; int count; int enable; TDK_RECT32 cover[ENCCOVERNUM_MAX]; }TDK_ENC_COVER;</pre>	
Members	chn	Channel.
	count	Cover count.
	enable	Enable or not.
	cover[ENCCOVERNUM_MAX]	Cover rect.

5.98 TDK_ENC_OSD

Table 5-98 TDK_ENC_OSD

Option	Instruction	
Struct description	OSD config.	
Structure	<pre>typedef struct { int chn; struct { int sup; char name[64]; struct { int sup; int modify_sup; int checked; TDK_RECT32 rect; }nameWidget; }chnName; struct { struct</pre>	

	<pre> { int sup; int modify_sup; int checked; TDK_RECT32 rect; }timeWidget; }chnTime; }TDK_ENC_OSD; </pre>	
Members	chn	Channel id.
	chnName.sup	Support channel name.
	chnName.name[64]	Whether support channel name.
	chnName.nameWidget t.sup	Whether support name widget.
	chnName.nameWidget t.modify_sup	Whether support modify name position.
	chnName.nameWidget t.check	Encode or not.
	chnName.nameWidget t.rect	channel name position.
	chnTime.timeWidget. sup	Whether support channel time widget.
	chnTime.timeWidget. modify_sup	Whether support modify time position.
	chnTime.timeWidget. checked	Encode or not
	chnTime.timeWidget. rect	Channel time position.

5.99 TDK_DEVICE_TCPIP

Table 5-99 TDK_DEVICE_TCPIP

Option	Instruction
Struct description	Device tcp/ip config.
Structure	<pre> typedef struct { TDK_NETWORK_TCP tcp; TDK_NETWORK_RTSP rtsp; TDK_NETWORK_DNS dns; TDK_NETWORK_ADAPTER adapter[MAX_NETWORK_ADAPTER_NUM]; TDKS8 network_adapter_count; TDKS8 network_adapter_default; </pre>

	TDKHANDLE } TDK_DEVICE_TCPIP; pcfg;	
Members	tcp	TCP config.
	rtsp	RTSP config.
	dns	DNS config.
	adapter[MAX_NETWORK_ADAPTER_NUM]	Network Adapter configs.
	network_adapter_count	Network adapter count.
	network_adapter_default	Default network adapter.
	pcfg	Handle of config internal.

5.100 TDK_NETWORK_TCP

Table 5-100 TDK_NETWORK_TCP

Option	Instruction
Struct description	Device tcp config.
Structure	<pre>typedef struct { TDKS32 HS_download_sup; TDKS32 HS_download; TDKS32 transfer_mode_sup; TDKS32 transfer_CheckBox_enable; TDKS32 transfer_CheckBox_checked; TDKS32 transfer_comboBox_enable; TDKS32 transfer_comboBox_value; TDKS32 http_port; TDKS32 http_port_sup; TDKS32 max_con_max; TDKS32 max_con_val; TDKS32 max_con_min; TDKS32 max_con_sup; TDKS32 moblie_port; TDKS32 moblie_port_sup; TDKS32 tcp_port;</pre>

	TDKS32 tcp_port_sup; TDKS32 https_port_sup; TDKS32 https_port; } TDK_NETWORK_TCP;	
Members	HS_download_sup	Whether support high speed download.
	HS_download	High speed download enable or not.
	transfer_mode_sup	Whether support transfer mode.
	transfer_CheckBox_enable	Whether support enable or disable transfer mode.
	transfer_CheckBox_checked	Transfer mode enable or not.
	transfer_comboBox_enable	Enable transfer mode.
	transfer_comboBox_value	Current transfer mode.
	http_port	Http port.
	http_port_sup	Whether support http port.
	max_con_max	Maximum connect number.
	max_con_val	Current connect limit value.
	max_con_min	Minimum connect number.
	max_con_sup	Whether support modify connect number.
	moblie_port	Mobile app port.
	moblie_port_sup	Whether support mobile port.
	tcp_port	Tcp port.
	tcp_port_sup	Whether support tcp port.
	https_port_sup	Whether support https port.
	https_port	Https port.

5.101 TDK_NETWORK_RTSP

Table 5-101 TDK_NETWORK_RTSP

Option	Instruction	
Struct description	RTSP config.	
Structure	<pre>typedef struct { TDKS32 rtsp_sup; TDKS32 rtsp_port; TDKS8 rtsp_url[MAX_RTSPURL_LEN]; } TDK_NETWORK_RTSP;</pre>	
Members	rtsp_sup	Whether support RTSP.
	rtsp_port	RTSP port.
	rtsp_url[MAX_RTSPURL_LEN]	RTSP access url.

5.102 TDK_NETWORK_DNS

Table 5-102 TDK_NETWORK_DNS

Option	Instruction	
Struct description	Network DNS config.	
Structure	<pre>typedef struct { TDKS8 major_dns_sup; TDKS8 major_dns[TDK_MAX_IPADDR_LEN]; TDKS8 minor_dns_sup; TDKS8 minor_dns[TDK_MAX_IPADDR_LEN]; } TDK_NETWORK_DNS;</pre>	
Members	major_dns_sup	Whether support major DNS.
	major_dns[TDK_MAX_IPADDR_LEN]	Major DNS.
	minor_dns_sup	Whether support minor DNS.
	minor_dns[TDK_MAX_IPADDR_LEN]	Minor DNS.

5.103 TDK_NETWORK_ADAPTER

Table 5-103 TDK_NETWORK_ADAPTER

Option	Instruction	
Struct description	Device tcp/ip config.	
Structure	<pre>typedef struct { TDKS8 ip[TDK_MAX_IPADDR_LEN]; TDKS8 mac_addr[TDK_MACADDR_LEN]; TDKS8 submask[TDK_MAX_IPADDR_LEN]; TDKS8 gateway[TDK_MAX_IPADDR_LEN]; TDKS8 inner_ip[TDK_MAX_IPADDR_LEN]; TDKS8 inner_ip_sup; TDKS8 dhcp_sup; TDKS8 dhcp_checked; TDKS8 ethType; TDKS32 ethType_sup; } TDK_NETWORK_ADAPTER;</pre>	
Members	ip[TDK_MAX_IPADDR_LEN]	IPv4 address.
	mac_addr[TDK_MACADDR_LEN]	MAC address.
	submask[TDK_MAX_IPADDR_LEN]	Ipv4 sub net mask.
	gateway[TDK_MAX_IPADDR_LEN]	Ipv4 gateway.
	inner_ip[TDK_MAX_IPADDR_LEN]	IPv4 internal ip.
	inner_ip_sup	Whether support inner ip.
	dhcp_sup	Whether support dhcp.
	dhcp_checked	DHCP enable or not.
	ethType	Ethernet type.
	ethType_sup	Whether support modify ethernet type.

5.104 TDK_SMART_PIC

Table 5-104 TDK_SMART_PIC

Option	Instruction	
Struct description	Related parameters of smart picture	
Structure	<pre>typedef struct tag_TDK_SMART_PIC { TDKU16 w; TDKU16 h; TDKU16 codec; TDKU16 ms; time_t tmf; TDKU32 len; char* data; char name[32]; char serialno[32]; char info[64]; }TDK_SMART_PIC, *PTDK_SMART_PIC;</pre>	
Members	w	Picture width.
	h	Picture height.
	codec	Codec type, TDKCODEC_XXX.
	ms	Picture time: millisecond.
	tmf	Picture time.
	len	Picture data length.
	data	Picture data.
	name[32]	Picture name, utf-8.
	serialno[32]	Serial number, utf-8.
	info[64]	Monitoring point information.

5.105 TDK_FACEOBJ

Table 5-105 TDK_FACEOBJ

Option	Instruction	
Struct description	Related parameters of face object.	
Structure	<pre>typedef struct tag_TDK_FACEOBJ { TDKU32 rsvd; TDKU16 id; TDKU16 quality; TDK_POINT16 lefttop; TDK_POINT16 rightend; TDKU32 attrEffectFlag; TDKU32 attr; }TDK_FACEOBJ, *PTDK_FACEOBJ;</pre>	
Members	rsvd	Reserved.
	id	ID.
	quality	Quality.
	lefttop	Left top coordinate in picture.
	rightend	Right end coordinate in picture.
	attrEffectFlag	Valid attribute flag.(ENUM_TDK_FACE_ATTR)
	attr	Object attribute value.(ENUM_TDK_FACE_ATTR)

5.106 TDK_FACE_ATTR

Table 5-106 TDK_FACE_ATTR

Option	Instruction	
Struct description	Related parameters of face attribute.	
Structure	<pre>typedef struct tag_TDK_FACE_ATTR { TDKU32 effectFlag; char name[64]; TDKU8 sim; TDKU8 age; TDKU8 attractive; TDKU8 glasses; TDKU8 race; TDKU8 emotion; TDKU16 featurelen; float feature[512]; TDKU32 res2; }TDK_FACE_ATTR, *PTDK_FACE_ATTR;</pre>	
Members	effectFlag	Specifies a valid flag. The value is obtained by bit. 1 indicates attribute valid. Reference ENUM_TDK_FACE_ATTR_EFFECTFLAG.
	name[64]	Name
	sim	Similarity
	age	Age
	attractive	Attractive
	glasses	Glasses
	race	Race
	emotion	Emotion
	featurelen	Feature description length.
	feature[512]	Feature description.
	res2	Reserved bytes.

5.107 TDK_LICENCEPLATE_OBJ

Table 5-107 TDK_LICENCEPLATE_OBJ

Option	Instruction	
Struct description	Related parameters of licence plate object.	
Structure	<pre>typedef struct tag_TDK_LICENCEPALTE_OBJ { TDKU32 rsvd; TDKU16 id; TDKU16 quality; TDK_POINT16 lefttop; TDK_POINT16 rightend; char licenceplate[256]; }TDK_LICENCEPALTE_OBJ, *PTDK_LICENCEPALTE_OBJ;</pre>	
Members	rsvd	Reserved bytes.
	id	ID
	quality	Quality.
	lefttop	The location of the recognized license plate in the source image, upper left, range : 0 ~ 8192.
	rightend	The location of the recognized license plate in the source image, lower right, range : 0 ~ 8192
	licenceplate	License plate, utf-8.

5.108 TDK_SMART_OBJ

Table 5-108 TDK_SMART_OBJ

Option	Instruction	
Struct description	Related parameters of smart object.	
Structure	<pre>typedef struct tag_TDK_SMART_OBJ { TDKU16 type; TDKU16 id; TDKU16 state; TDKU16 pointnum; TDK_POINT32 point[TDK_MAX_NUM_LINE_POINT]; }TDK_SMART_OBJ;</pre>	
Members	type	Smart event type, reference SMART_FRAME_INFO_TYPE
	id	Object Id.
	state	0-normal, 1-alarm.
	pointnum	Object point number.
	point[TDK_MAX_NUM_LINE_POINT]	Object points.

5.109 TDK_DEVICE_NET_INFO

Table 5-109 TDK_DEVICE_NET_INFO

Option	Instruction	
Struct description	Related parameters of device network in search result.	
Structure	<pre>typedef struct { TDKS8 ip[TDK_MAX_IPADDR_LEN]; TDKS8 ipv6[TDK_MAX_IPV6ADDR_LEN]; TDKU16 media_port; TDKU16 media_port_ipv6; TDKU16 http_port; TDKS8 submask[TDK_MAX_IPADDR_LEN]; TDKU32 ipv6_perfix; TDKS8 gateway[TDK_MAX_IPADDR_LEN]; TDKS8 gateway_ipv6[TDK_MAX_IPV6ADDR_LEN]; TDKS8 mac_addr[TDK_MACADDR_LEN]; TDKS8 major_dns[TDK_MAX_IPADDR_LEN]; TDKS8 major_dns_ipv6[TDK_MAX_IPV6ADDR_LEN]; TDKS8 minor_dns[TDK_MAX_IPADDR_LEN]; TDKS8 minor_dns_ipv6[TDK_MAX_IPV6ADDR_LEN]; }</pre>	

	TDKS8 device_type[TDK_DEV_TYPE_LEN]; TDKS8 device_oem[TDK_DEV_OEM_LEN]; TDKS8 device_model[TDK_DEV_MODEL_LEN]; TDKS8 networkcardip[TDK_MAX_IPADDR_LEN]; }TDK_DEVICE_NET_INFO;	
Members	ip[TDK_MAX_IPADDR_LEN]	IPv4 ip address.
	ipv6[TDK_MAX_IPV6ADDR_LEN]	IPv6 ip address.
	media_port	Media port.
	media_port_ipv6	IPv6 media port.
	http_port	Http port.
	submask[TDK_MAX_IPADDR_LEN]	IPv4 sub net mask.
	ipv6_perfix	IPv6 prefix.
	gateway[TDK_MAX_IPADDR_LEN]	IPv4 gateway.
	gateway_ipv6[TDK_MAX_IPV6ADDR_LEN]	IPv6 gateway.
	mac_addr[TDK_MACADDR_LEN]	MAC address
	major_dns[TDK_MAX_IPADDR_LEN]	IPv4 major DNS.
	major_dns_ipv6[TDK_MAX_IPV6ADDR_LEN]	IPv6 major DNS.
	minor_dns[TDK_MAX_IPADDR_LEN]	IPv4 minor DNS.
	minor_dns_ipv6[TDK_MAX_IPV6ADDR_LEN]	IPv6 minor DNS.
	device_type[TDK_DEV_TYPE_LEN]	Device type.
	device_oem[TDK_DEV_OEM_LEN]	Device OEM.
	device_model[TDK_DEV_MODEL_LEN]	Device model
	networkcardip[TDK_MAX_IPADDR_LEN]	The network card IP.

5.110 TDK_GLOBAL_PARAM

Table 5-110 TDK_GLOBAL_PARAM

Option	Instruction	
Struct description	Related parameters of global init.	
Structure	<pre>typedef struct tag_TDK_GLOBAL_PARAM { TDKU32 streamThreadCount; TDKBOOL enableLogging; TDKBOOL enableAutoReconnect; }TDK_GLOBAL_PARAM, * PTDK_GLOBAL_PARAM;</pre>	
Members	streamThreadCount	The number of threads in the thread pool that receives video and audio streams. If streamThreadCount is less than 1, the actual number of threads is 1, and if streamThreadCount is greater than 256, the actual number of threads is 256.
	enableLogging	Enable SDK logging or not.
	enableAutoReconnect	Enable auto reconnect or not when logging and pulling streams fail.

6 Enumeration

6.1 TDK_IMAGE_FORMAT

Table 6-1 TDK_IMAGE_FORMAT

Item	Description
Enumeration description	Image type enumeration.
Structure	<pre>typedef enum tag_TDK_IMAGE_FORMAT { TDK_IMAGE_BMP, //.bmp format TDK_IMAGE_JPEG, //.jpg format }TDK_IMAGE_FORMAT;</pre>

6.2 TDK_RECORD_FORMAT

Table 6-2 TDK_RECORD_FORMAT

Item	Description
Enumeration description	Record type enumeration.
Structure	<pre>typedef enum tag_TDK_RECORD_FORMAT { TDK_RECORD_DAV, //.dav format TDK_RECORD_MP4, //.mp4 format }TDK_RECORD_FORMAT;</pre>

6.3 TDK_DEVICE_TYPE

Table 6-3 TDK_DEVICE_TYPE

Item	Description
Enumeration description	Device type enumeration.
Structure	<pre>typedef enum { TDKDEVICE_TYPE_DVR, //DVR TDKDEVICE_TYPE_IPC, //IPC }TDK_DEVICE_TYPE;</pre>

6.4 TDK_ENUM_CHANNEL_STATE

Table 6-4 TDK_ENUM_CHANNEL_STATE

Item	Description
Enumeration description	Channel state enumeration.
Structure	<pre>typedef enum tag_TDK_ENUM_CHANNEL_STATE { TDK_CHN_STATE_UNKOWN = 0, //Unkown TDK_CHN_STATE_NOCONFIG = 1, //No config TDK_CHN_STATE_DISABLE = 2, //Disable TDK_CHN_STATE_CONNECTING = 3, //Connecting TDK_CHN_STATE_AUTHORITY = 4, //User authentication error TDK_CHN_STATE_LINKED = 5, //Linked }TDK_ENUM_CHANNEL_STATE;</pre>

6.5 ENUM_TDK_CROSSING_LINE_STATISTICS_REPORT

Table 6-5 ENUM_TDK_CROSSING_LINE_STATISTICS_REPORT

Item	Description
Enumeration description	Crossing line statistics report enumeration.
Structure	<pre>typedef enum tag_ENUM_TDK_CROSSING_LINE_STATISTICS_REPORT { TDK_DAILY_REPORT = 0, //Daily report TDK_WEEKLY_REPORT, //Weekly report TDK_MONTHLY_REPORT, //Monthly report TDK_ANNUAL_REPORT, //Annual report }ENUM_TDK_CROSSING_LINE_STATISTICS_REPORT;</pre>

6.6 ENUM_TDK_SMART

Table 6-6 ENUM_TDK_SMART

Item	Description
Enumeration description	Smart events enumeration.
Structure	<pre> typedef enum tag_ENUM_TDK_SMART { TDK_SMART_LINE_CROSSING = 0, //line crossing TDK_SMART_AREA_INTRUSION, //area intrusion TDK_SMART_MOTION_DETCTION, //motion detection TDK_SMART_REGION_ENTRANCE, //region entrance TDK_SMART_REGION_EXITING, //region exiting TDK_SMART_FAST_MOVING, //fast moving TDK_SMART_UNATTENDED_OBJECT, //unattended object TDK_SMART_OBJECT_MISSING, //object missing TDK_SMART_FACE_DETECTION, //face detection TDK_SMART_LOITERING_DETECTION, //loitering detection TDK_SMART_PARKING_DETECTION, //parking detection TDK_SMART_PEOPLE_GATHERING_DETECTION, //people gathering detection TDK_SMART_CROSSING_LINE_STATISTICS, //crossing line statistics TDK_SMART_BLURRED_DETECTION, //blurred detection TDK_SMART_SCENE_CHANGE_DETECTION, //scene change detection TDK_SMART_AUDIO_EXCEPTION_DETECTION, //audio exception detection }ENUM_TDK_SMART; </pre>

6.7 ENUM_TDK_ALARM_IN_TYPE

Table 6-7 ENUM_TDK_ALARM_IN_TYPE

Item	Description
Enumeration description	Alarm input type enumeration.
Structure	<pre> typedef enum tag_ENUM_TDK_ALARM_IN_TYPE { TDK_NORMAL_CLOSE = 0, //normal closed TDK_NORMAL_OPEN, //normal opened }ENUM_TDK_ALARM_IN_TYPE; </pre>

6.8 ENUM_TDK_ALARM_OUT_TYPE

Table 6-8 ENUM_TDK_ALARM_OUT_TYPE

Item	Description
Enumeration description	Alarm output type enumeration.
Structure	<pre>typedef enum tag_ENUM_TDK_ALARM_OUT_TYPE { TDKTYPE_STOP = 0, //stop TDKTYPE_MANUAL, //manual TDKTYPE_SCHEDULE, //schedule }ENUM_TDK_ALARM_OUT_TYPE;</pre>

6.9 ENUM_TDK_ALARM_OUT_STATUS

Table 6-9 ENUM_TDK_ALARM_OUT_STATUS

Item	Description
Enumeration description	Alarm output status enumeration.
Structure	<pre>typedef enum tag_ENUM_TDK_ALARM_OUT_STATUS { TDK_OFF = 0, TDK_ON, }ENUM_TDK_ALARM_OUT_STATUS;</pre>

6.10 ENUM_TDK_PTZ_CONTROL

Table 6-10 ENUM_TDK_PTZ_CONTROL

Item	Description
Enumeration description	PTZ control enumeration.
Structure	<pre>typedef enum tag_ENUM_TDK_PTZ_CONTROL { TDK_PTZ_UP = 0, //up TDK_PTZ_DOWN, //down TDK_PTZ_LEFT, //left TDK_PTZ_RIGHT, //right TDK_PTZ_UPPER_LEFT, //upper left TDK_PTZ_UPPER_RIGHT, //upper right TDK_PTZ_LOWER_LEFT, //lower left TDK_PTZ_LOWER_RIGHT, //lower right TDK_PTZ_ZOOM_EXPAND, //zoom expand TDK_PTZ_ZOOM_REDUCE, //zoom reduce TDK_PTZ_FOCUS_EXPAND, //focus expand TDK_PTZ_FOCUS_REDUCE, //focus reduce TDK_PTZ_IRIS_EXPAND, //iris expand TDK_PTZ_IRIS_REDUCE, //iris reduce }ENUM_TDK_PTZ_CONTROL;</pre>

6.11 ENUM_TDK_FACE_ATTR

Table 6-11 EMUM_TDK_FACE_ATTR

Item	Description
Enumeration description	Face attribute enumeration.
Structure	<pre>typedef enum tag_ENUM_TDK_FACE_ATTR { ENUM_TDK_FACE_ATTR_SMILE = 0, //smile 0: no smile, 1: smile ENUM_TDK_FACE_ATTR_GENDER, //gender 0: man, 1: women ENUM_TDK_FACE_ATTR_MASK, //mask 0: no mask, 1: take mask ENUM_TDK_FACE_ATTR_EYE, //eye 0: close, 1: open ENUM_TDK_FACE_ATTR_MOUTH, //mouth 0: close, 1: open ENUM_TDK_FACE_ATTR_BEARD, //bread 0: no, 1: have ENUM_TDK_FACE_ATTR_GLASSES, //glasses 0: no, 1: have }ENUM_TDK_FACE_ATTR;</pre>

6.12 SMART_FRAME_INFO_TYPE

Item	Description
Enumeration description	Smart information type
Structure	<pre>typedef enum tag_SMART_FRAME_INFO_TYPE { OBJECT_INFO = 1, LINE_INFO = 2, AREA_INFO = 3, AREAIN_INFO = 4, AREAOUT_INFO = 5, AREAMOVE_INFO = 6, AREALEFT_INFO = 7, AREALOSE_INFO = 8, AREAFACEDTECT_INFO = 9, AREAWANDER_INFO = 10, AREAPARK_INFO = 11, AREACROWD_INFO = 12, AREAHUMANSHAPE_INFO = 26, AREALPR_INFO = 27, AREAVEHICLESHAPE_INFO = 30, }SMART_FRAME_INFO_TYPE;</pre>

7 Error Code

Name	Value	Description	Note
TDK_OK	0	Success	
TDK_FAIL	-1	General error	
TDKERR_TIMEOUT	-2	Timeout	
TDKERR_PENDING	-3	The operation is pending and completed later.	
TDKERR_PARAMETER	-4	Invalid input parameters.	
TDKERR_HANDLE	-5	Invalid handle.	
TDKERR_MEMORY	-6	Memory error.	
TDKERR_STATE	-7	Invalid state.	General because interface call order wrong.
TDKERR_FULL	-8	Buffer full.	General because data count beyond the maximum limit.
TDKERR_EMPTY	-9	Data is empty.	
TDKERR_SUPPORT	-10	The operation is not support.	
TDKERR_FOUND	-11	The resource or object cannot be found.	
TDKERR_RESOURCE	-12	Resource depletion.	
TDKERR_SEND	-13	Send data fail.	
TDKERR_RECV	-14	Receive data fail.	
TDKERR_FORMAT	-15	Data format error.	
TDKERR_AUTHEN	-16	Authenticate fail.	
TDKERR_EXISTED	-17	Object is already existed.	
TDKERR_END	-18	Operation has ended.	
TDKERR_CANCELLED	-19	Operation is cancelled.	
TDKERR_FILE_OPEN	-20	Open file failed.	

TDKERR_FILE_WRITE	-21	Write data to file failed.	
TDKERR_DISK_FULL	-22	Disk has no space.	
TDKERR_FOLDER_CREATE	-23	Create folder failed.	
TDKERR_FILE_READ	-24	Read data from file failed.	
TDKERR_ADDRESS	-25	Address url error.	
TDKERR_REMOVED	-26	Object or source has been removed.	
TDKERR_BUSY	-27	System is busy.	
TDKERR_NETWORK	-28	Network error.	
TDKERR_PASSWORD	-29	Password error.	
TDKERR_ACCOUNT	-30	Account error.	
TDKERR_RELOGIN	-31	Repeated login.	
TDKERR_LOCKED	-32	Account is locked.	
TDKERR_BLACKLIST	-33	Account is in blacklist.	
TDKERR_NOT_FIND_HOST	-34	Not find the device host.	
TDKERR_INACTIVE	-35	Device is unactivated.	
TDKERR_CUSTOMER_NUM	-36	The number of customer error.	
TDKERR_OPENSTREAM	-700	Failed to open the stream	
TDKERR_OPENCHANNEL	-900	Failed to open the channel.	

8 URL

8.1 Real-time Monitoring

protocol://ip domain:port/[authen=devusr&devpwd/cloudid=id/]mode=real&idc=1&ids=1[&hp=80]		
Parameter	Description	Value
protocol	Data transmission protocol type.	tdk - Private protocol. umsp - Cloud 1.0 tdks - Cloud 2.0
ip domain	IP address or domain name of the device. It represents device ip or domain when protocol is tdk. It represents cloud server ip or domain when protocol is umsp or tdks.	tdk - IP or domain. umsp - api.qvcloud.net tdks - global.qvcloud.net
port	Device media port. It represents device media port when protocol is tdk. It represents cloud server port when protocol is umsp or tdks.	Default value: tdk - 34567 umsp - 8300 tdks - 443
authen	[optional] Device username, password. They are needed when protocol is umsp or tdks.	
cloudid	[optional] Device cloud id. It is needed when protocol is umsp or tdks.	
idc	Preview channel id.	Start from 1.
ids	Stream id.	1-Main stream 2-Sub stream 3-Third stream
hp	[optional] Device http port. It is needed when protocol is tdk.	Default value is 80

8.2 Remote Config

protocol://ip domain:port/[authen=devusr&devpwd/cloudid=id/][hp=80]		
Parameter	Description	Value
protocol	Data transmission protocol type.	tdk - Private protocol. umsp - Cloud 1.0 tdks - Cloud 2.0
ip domain	IP address or domain name of the device. It represents device ip or domain when protocol is tdk. It represents cloud server ip or domain when protocol is umsp or tdks.	tdk - IP or domain. umsp - api.qvcloud.net tdks - global.qvcloud.net
port	Device media port. It represents device media port when protocol is tdk. It represents cloud server port when protocol is umsp or tdks.	tdk - 34567 umsp - 8300 tdks - 443
authen	[optional] Device username, password. They are needed when protocol is umsp or tdks.	
cloudid	[optional] Device cloud id. It is needed when protocol is umsp or tdks.	
hp	[optional] device http port. It is needed when protocol is tdk.	Default-80

8.3 Record Search and Playback

protocol://ip domain:port/[authen=devusr&devpwd/cloudid=id/][hp=80]		
Parameter	Description	Value
protocol	Data transmission protocol type.	tdk - Private protocol. umsp - Cloud 1.0 tdks - Cloud 2.0
ip domain	IP address or domain name of the device. It represents device ip or domain when protocol is tdk. It represents cloud server ip or domain when protocol is umsp or tdks.	tdk - IP or domain. umsp - api.qvcloud.net tdks - global.qvcloud.net
port	Device media port. It represents device media port when protocol is tdk. It represents cloud server port when protocol is umsp or tdks.	tdk - 34567 umsp - 8300 tdks - 443
authen	[optional] Device username, password. They are needed when protocol is umsp or tdks.	
cloudid	[optional] Device cloud id. It is needed when protocol is umsp or tdks.	
hp	[optional] device http port. It is needed when protocol is tdk.	Default-80

8.4 Talk and Broadcast

protocol://ip domain:port/[authen=devusr&devpwd/cloudid=id/]talk[&hp=80]		
Parameter	Description	Value
protocol	Data transmission protocol type.	tdk - Private protocol. umsp - Cloud 1.0 tdks - Cloud 2.0
ip domain	IP address or domain name of the device. It represents device ip or domain when protocol is tdk. It represents cloud server ip or domain when protocol is umsp or tdks.	tdk - IP or domain. umsp - api.qvcloud.net tdks - global.qvcloud.net
port	Device media port. It represents device media port when protocol is tdk. It represents cloud server port when protocol is umsp or tdks.	tdk - 34567 umsp - 8300 tdks - 443
authen	[optional] Device username, password. They are needed when protocol is umsp or tdks.	
cloudid	[optional] Device cloud id. It is needed when protocol is umsp or tdks.	
hp	[optional] device http port. It is needed when protocol is tdk.	Default-80

8.5 Alarm Listen

protocol://ip domain:port/[authen=devusr&devpwd/cloudid=id/]alarm[&hp=80]		
Parameter	Description	Value
protocol	Data transmission protocol type.	tdk - Private protocol. umsp - Cloud 1.0 tdks - Cloud 2.0
ip domain	IP address or domain name of the device. It represents device ip or domain when protocol is tdk. It represents cloud server ip or domain when protocol is umsp or tdks.	tdk - IP or domain. umsp - api.qvcloud.net tdks - global.qvcloud.net
port	Device media port. It represents device media port when protocol is tdk. It represents cloud server port when protocol is umsp or tdks.	tdk - 34567 umsp - 8300 tdks - 443
authen	[optional] Device username, password. They are needed when protocol is umsp or tdks.	
cloudid	[optional] Device cloud id. It is needed when protocol is umsp or tdks.	
hp	[optional] device http port. It is needed when protocol is tdk.	Default-80