ACTi SDK-10000

C Library Edition

v1.2 SP1

API Reference Guide

www.acti.com

Table of Contents

1 OVERVIEW

Introduction

This SDK can help with application go beyond passive viewing to interact with the application developed by system integrator. This SDK provides a real time streaming to deliver live video and other surveillance functions controlling.

Start Up with Streaming Client Library

Streaming Client Library is developed for MPEG-4/MJPEG/H.264 Video Network Streaming Application.

It contains following abilities:

- MPEG-4/MJPEG/H.264 Software Decoding
- Multicast and Unicast Streaming
- Video Render
- Embedded Time Code
- IO Controlling
- Event Notify from Server.
- Recording Trigger by Different Mode
- Discovery the Server that exists on the net

Following is a scenario of an application.

- **Open the Interface**

The application can connect one or more than one Server by using

```
HANDLE  myCamera1 = KOpenInterface ();
HANDLE  myCamera2 = KOpenInterface ();
```

Then the application can use the handle to using the SDK function.

- **Prepare structures**

There are some structures need to be prepared after using the interface.

```
MEDIA_CONNECTION_CONFIG : for Register to the Server
```

```
MEDIA_COMMAND
STREAMING_ENGINE_CONFIG : for Register to the Streaming Engine
MEDIA_VIDEO_CONFIG : for Get/Set Server Setting
MEDIA_PORT_INFO : for Get Server Port Information
MEDIA_RENDER_INFO : for Stream Video Display
MEDIA_MOTION_INFO : for Motion Detect Range Setting
MP4FILE_RECORD_INFO : for retrive record information
```

- **CallBack functions**

There are callback functions to pass information to application.

```
CONTROL_DATA_CALLBACK
RS232_DATA_CALLBACK
TIME_CODE_CALLBACK
TIME_CODE_CALLBACK_EX
VIDEO_LOSS_CALLBACK
VIDEO_RECOVERY_CALLBACK
NETWORK_LOSS_CALLBACK
MOTION_DETECTION_CALLBACK
QUAD_MOTION_DETECTION_CALLBACK
DI_CALLBACK_FOR_4100
DI_CALLBACK
DI_CALLBACK_EX
RAW_DATA_CALLBACK
IMAGE_CALLBACK
AFTER_RENDER_CALLBACK
RESOLUTION_CHANGE_CALLBACK
FILE_PLAY_COMPLETE_CALLBACK
QUAD_VIDEO_LOSS_CALLBACK
FILE_PLAY_COMPLETE_CALLBACK
FIRST_B2_CALLBACK
```

NOTE: The CallBack functions need be set after KOpenInterface.

- **Build a connection and connect to server**

```
MEDIA_CONNECTION_CONFIG mcc;
…
if(KSetMediaConfig(myCamera1, &mcc))
{
    if(KConnect(myCamera1))
    {
        if(KStartStream(myCamera1))
        {
            KPlay(myCamera1);
```

```
        }
    }
}
```

- **Disconnect the server**

```
KStop(myCamera1);
KStopStreaming(myCamera1);
KDisconnect(myCamera1);
```

- **Quit the interface**

```
KCloseInterface(myCamera1);
myCamera1 = NULL;
```

NOTE: The SDK will handle the video preview.

The video will display on the top, left with the width = 360 and height = 240 of the `MyWinInfo.hwnd`. (The video will be stretched to the `MyWinInfo.dwWidth` and `MyWinInfo.Height`)

```
MEDIA_RENDER_INFO mri;
mri.DrawerInterface = DGDI;
mri.rect.top = 0;
mri.rect.left = 0;
mri.rect.right = 360;
mri.rect.bottom = 240;
mri.hWnd = HandleOfTeWin;
mri.hwnd = HandleOfTeWin;
KSetRenderInfo( h, &mri );
```

- If the application just want recording or get the raw data but preview , please call

```
KEnableDecoder( h, false );
```

The SDK will disable the decode and preview capability

- If the application handles the video such as Preview, it needs to set the SDK

KSetImageCallBack function.

Then the SDK will pass the Video Data (BMP) to Application. (See the Sample Program Source Code)

- If the application want to restream the video (It means the application just want the mpeg4/MJPEG/H.264 raw data), the application need to set KSetRawDataCallback function. Then the SDK will pass the Video Data (Mpeg4/MJPEG/H.264) to Application.

- If the application needs the time code, set KSetTimeCodeCallBack function, and the SDK will pass the TimeCode to Application.

- If the application has to receive RS232 response, the application needs to set KSetRS232DataCallback function. Then the SDK will pass the response to Application.

Start Up with Playback Library

Playback Library is developed for MPEG-4/MJPEG/H.264 Video Files Playback Application.

It contains following abilities:

- Use customized MPEG-4/MJPEG/H.264 Software Decoding
- Fast forward/backward and slow forward/backward
- Get time code from media files recorded by video server.
- Support playback multiple media files.
- Support full screen playback mode.

Following is a scenario of an application.

- **Open the Interface**

The application can allocate more then one playback instants

```
HANDLE  hPlayback1 = KOpenInterface ();
HANDLE  hPlayback2 = KOpenInterface ();
```

Then the application can use the handle to using the SDK function.

- **Prepare structures**

There are some structures need to be prepared after using the interface.

```
MEDIA_CONNECTION_CONFIG : for file information
STREAMING_ENGINE_CONFIG : for Register to the Streaming Engine
MEDIA_RENDER_INFO : for Stream Video Display
MEDIA_MOTION_INFO : for Motion Detect Range Setting
MP4FILE_RECORD_INFO : for retrive record information
```

- **CallBack functions**

There are callback functions to pass information to application.

```
TIME_CODE_CALLBACK
TIME_CODE_CALLBACK_EX
DI_CALLBACK
DI_CALLBACK_EX
RAW_DATA_CALLBACK
IMAGE_CALLBACK
AFTER_RENDER_CALLBACK
FILE_PLAY_COMPLETE_CALLBACK
DI_CALLBACK_FOR_4100
QUAD_VIDEO_LOSS_CALLBACK
FILE_PLAY_COMPLETE_CALLBACK
```

```
FIRST_B2_CALLBACK
```

- **Open & Play a media file.**

```
MEDIA_CONNECTION_CONFIG mcc;
…
if(KSetMediaConfig(hPlayback1, &mcc))
{
     if(KConnect(hPlayback1))
     {
          if(KStartStream(hPlayback1))
          {
               KPlay(hPlayback1);
          }
     }
}
```

- **Playback control functions**

```
KPlay(hPlayback1);
KPause(hPlayback1);
KSetRate(hPlayback1, iPlayRate);
KStepNextFrame(hPlayback1);
KStepPrevFrame(hPlayback1);
KSetPlayDirection(hPlayback1, bForward );
KSetCurrentTime(hPlayback1, Timecode);
```

- **Close the media file**

```
KStop(hPlayback1);
KStopStreaming(hPlayback1);
KDisconnect(hPlayback1);
```

- **Quit the interface**

```
KCloseInterface(hPlayback1);
```

NOTE: The SDK will handle the video preview.

The video will display on the top, left with the width = 360 and height = 240 of the `MyWinInfo.hwnd`. (The video will be stretched to the `MyWinInfo.dwWidth` and `MyWinInfo.Height`)

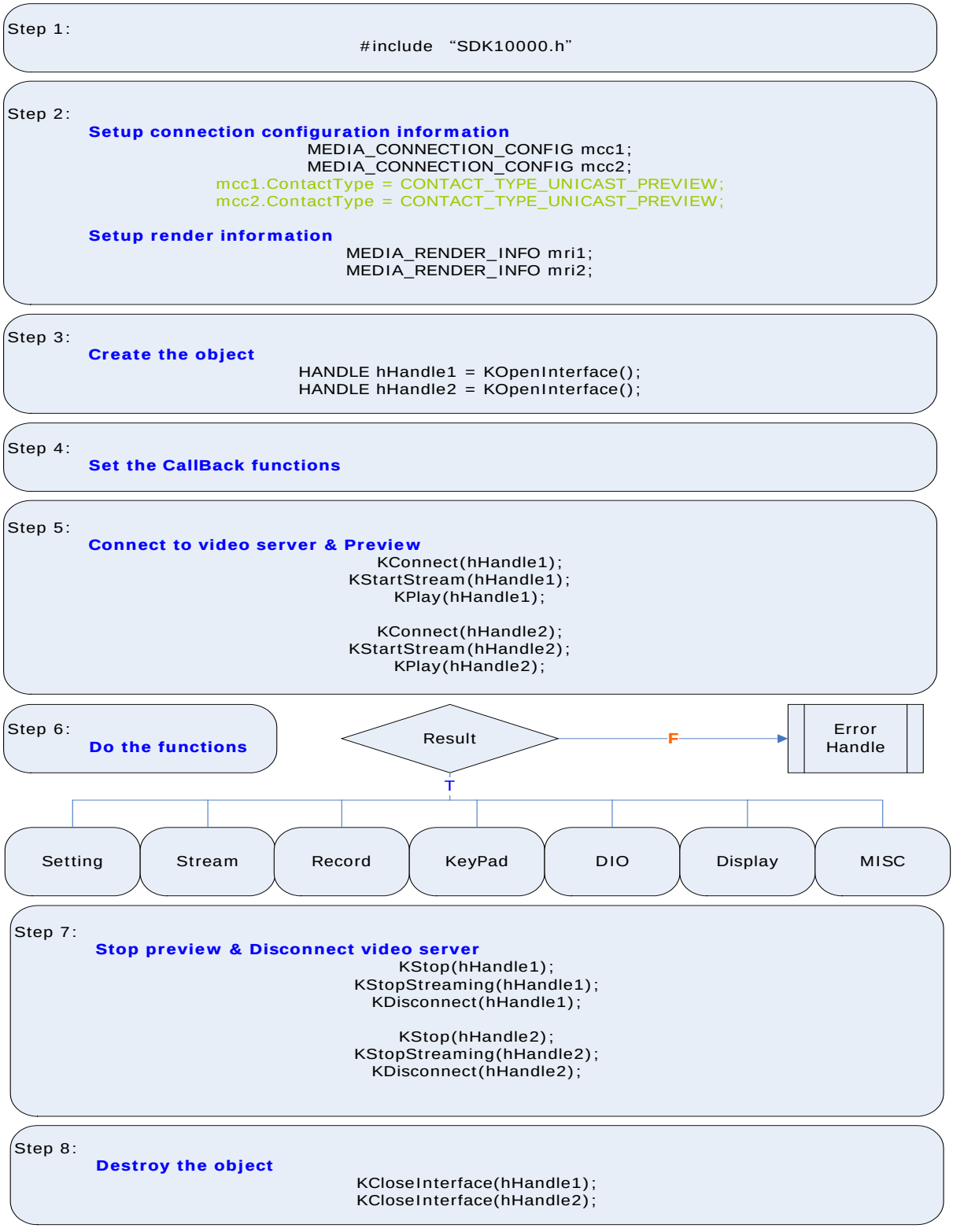
```
MEDIA_RENDER_INFO mri;
```

```
mri.DrawerInterface = DGDI;
mri.rect.top = 0;
mri.rect.left = 0;
mri.rect.right = 360;
mri.rect.bottom = 240;
mri.hWnd = HandleOfTeWin;
KSetRenderInfo( h, &mri );
```

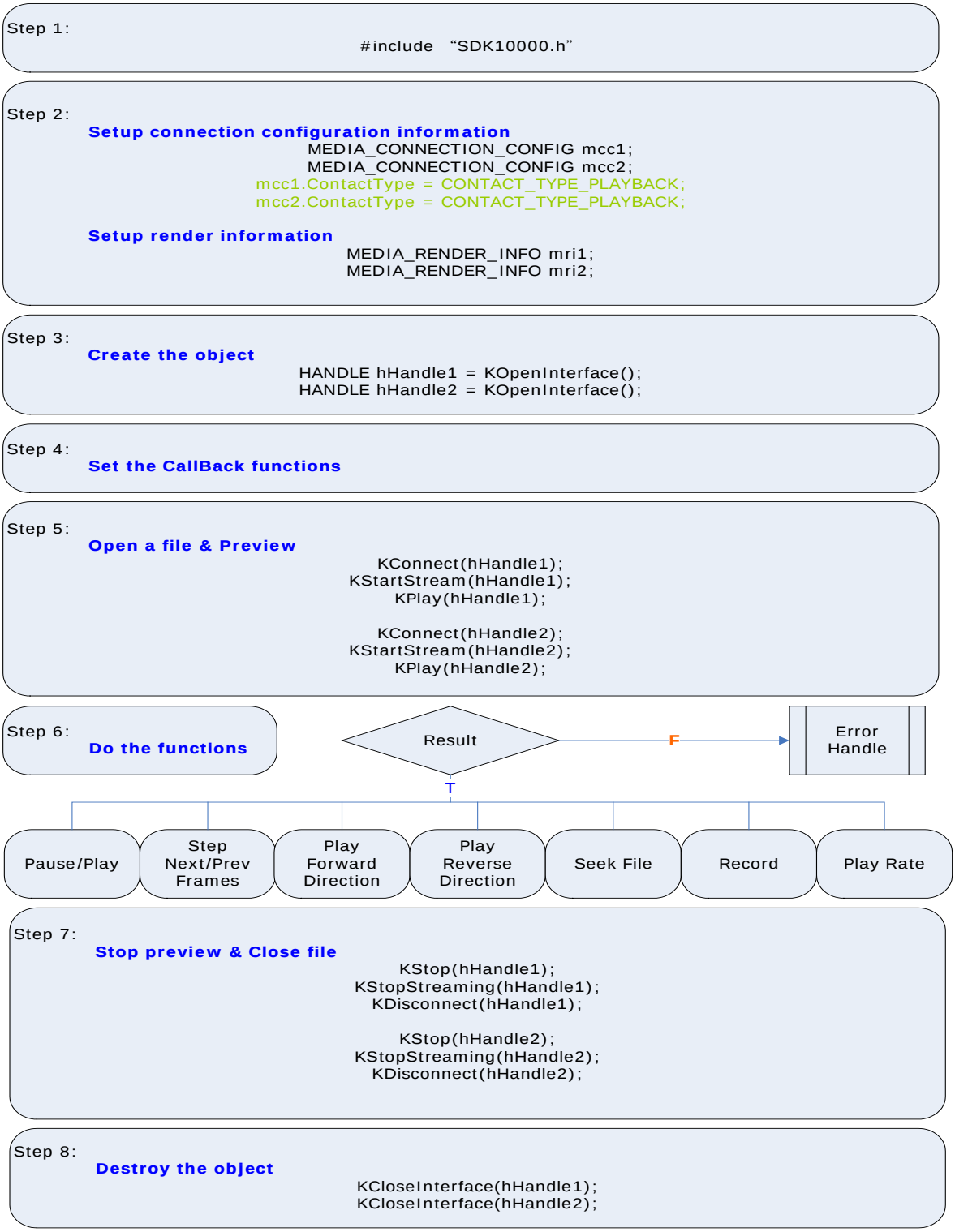
The SDK will determine the video window size according the video size

NOTE: Required utilities can be accessed in the bundled CD.

Streaming API Architectures



API Architectures



What’s New in this release

1. Add support to Megapixel MPEG-4/MJPEG/H.264 decoding
2. Add support to Intel IPP decoder

2 Data Structure

MEDIA_CONNECTION_CONFIG

The **MEDIA_CONNECTION_CONFIG** structure enables the media source information.

```
typedef struct structural_MEDIA_CONNECTION_CONFIG
{
    int              ContactType;

    // int ChannelNumber resize to unsigned short,
    // add unsigned short StreamID
    unsigned short    ChannelNumber;
    unsigned short    StreamID;

    char         UniCastIP[16];
    char         MultiCastIP[16];
    char         PlayFileName[256];
    char         UserID[64];
    char         Password[64];
    unsigned long RegisterPort;
    unsigned long StreamingPort;
    unsigned long ControlPort;
    unsigned long MultiCastPort;
    unsigned long SearchPortC2S;
    unsigned long SearchPortS2C;
    unsigned long HTTPPort;
    unsigned long RTSPPort;
    unsigned long VideoRTPOverMCastPort;
    unsigned long AudioRTPOverMCastPort;

    // int ConnectTimeOut resize to unsigned short,
    // add unsigned short EncryptionType
    unsigned short    ConnectTimeOut;
    unsigned short    EncryptionType;
}MEDIA_CONNECTION_CONFIG;
```

Members

ContactType

Contact Type	Description
CONTACT_TYPE_UNICAST_WOC_PREVIEW	Preview - Uni-cast without control port, using ATCP10 and ATCP20
CONTACT_TYPE_MULTICAST_WOC_PREVIEW	Preview - Multicast without control port, using AMCST10 and AMCST20
CONTACT_TYPE_RTSP_PREVIEW	Preview - RTSP , using ARTSP(Not Support)
CONTACT_TYPE_CONTROL_ONLY	Control only - using ATCP10 and ATCP20
CONTACT_TYPE_UNICAST_PREVIEW	Uni-cast , using ATCP10 and ATCP20
CONTACT_TYPE_MULTICAST_PREVIEW	Preview - Multicast, using AMCST10 and AMCST20
CONTACT_TYPE_PLAYBACK	Playback - Playback, using ARAW
CONTACT_TYPE_CARD_PREVIEW	Preview - 4100 preview, using A4100

ChannelNumber

Camera channel number for Multi-Channel video to use.

StreamID

Streaming ID number for “Dual Stream” devices (0 or 1).

UniCastIP

Camera IP address.

MultiCastIP

Camera Multicast IP address.

PlayFileName

File name for Playback.

UserID

User login ID.

Password

User login password.

RegisterPort

Register port number.

StreamingPort

Streaming port number.

ControlPort

Control port number.

MultiCastPort

Multicast port number.

SearchPortC2S

Search port number (Client to Server)

SearchPortS2C

Search port number (Server to Client).

HTTPPort

HTTP port number.

RTSPPort

RTSP port number

VideoRTPOverMCastPort

Video RTP over multicast port number.

AudioRTPOverMCastPort

Audio RTP over multicast port number.

ConnectTimeOut

Time out value for connect.

MEDIA_MOTION_INFO

The **MEDIA_MOTION_INFO** structure is used to set/retrieve motion information on video server.

```
typedef struct structural_MEDIA_MOTION_INFO
{
     DWORD    dwEnable;
     DWORD    dwRangeCount;
     DWORD    dwRange[3][4];
     DWORD    dwSensitive[3];
} MEDIA_MOTION_INFO;
```

Members

dwEnable

Flag to enable motion

dwRangCount

Number of Ranger count.

dwRange

Range area (3 can be set).

dwSensitive

Sensitive of range (3 can be set).

MEDIA_PORT_INFO

The **MEDIA_PORT_INFO** structure is used to retrieve video server port information.

```
typedef struct structural_MEDIA_PORT_INFO /** Device port info. */
{
    unsigned long    PORT_HTTP;
    unsigned long    PORT_SearchPortC2S;
    unsigned long    PORT_SearchPortS2C;
    unsigned long    PORT_Register;
    unsigned long    PORT_Control;
    unsigned long    PORT_Streaming;
    unsigned long    PORT_Multicast;
    unsigned long    PORT_RTSP;
} MEDIA_PORT_INFO;
```

Members

PORT_HTTP

HTTP Port

PORT_SearchPortC2S

Search Port Client to Server

PORT_SearchPortS2C

Search Port Server to Client

PORT_Register

Register port number

PORT_Control

Control Port number

PORT_Streaming

Streaming Port number

PORT_Multicast

Multicast Port number

PORT_RTSP

RTSP Port number

MEDIA_PTZ_PROTOCOL

The **MEDIA_PTZ_PROTOCOL** structure is used to specify the protocol resource.

```
typedef struct structural_MEDIA_PTZ_PROTOCOL
{
    int nSourceType;
    char szVender[32];
    char szProtocol[32];
    char szProtocolFileName[512];
    DWORD dwAddressID;
} MEDIA_PTZ_PROTOCOL;
```

Members

nSourceType

Specify the source type is inside resource or a PTZ protocol file

szVender[32]

The vender name.

szProtocol[32]

The protocol name.

szProtocolFileName[512]

The PTZ protocol file name.

dwAddressID

Address ID.

MEDIA_RENDER_INFO

The **MEDIA_RENDER_INFO** structure is used to set render information.

```
typedef struct structural_MEDIA_RENDER_INFO
{
    int      DrawerInterface;
    HWND     hWnd;
    RECT     rect;
} MEDIA_RENDER_INFO;
```

Members

DrawerInterface

DrawInterface	Description
DGDI (0)	use Windows GDI for draw
DXDRAW (1)	use Direct Draw for draw

hWnd

Handle of window.

rect

Area to draw.

MEDIA_VIDEO_CONFIG

The **MEDIA_VIDEO_CONFIG** structure is used to set/retrieve video configuration.

```
typedef struct structural_MEDIA_VIDEO_CONFIG
{
    DWORD    dwTvStander;
    DWORD    dwVideoResolution;
    DWORD    dwBitsRate;
    DWORD    dwVideoBrightness;
    DWORD    dwVideoContrast;
    DWORD    dwVideoSaturation;
    DWORD    dwVideoHue;
    DWORD    dwFps;
} MEDIA_VIDEO_CONFIG;
```

Members

dwTvStander

TV Stander	Description
NTSC (0)	NTSC
PAL (1)	PAL

dwVideoResolution

Resolution	Description
NTSC_720x480 (0)	NTSC - 720 x 480
NTSC_352x240 (1)	NTSC - 352 x 240.
NTSC_160x112 (2)	NTSC - 160 x 112.
PAL_720x576 (3)	PAL - 720 x 576
PAL_352x288 (4)	PAL - 352 x 288
PAL_176x144 (5)	PAL - 176 x 144.
PAL_176x120 (6)	PAL - 176 x 120
NTSC_640x480 (64)	NTSC - 640 x 480.
PAL_640x480 (192)	PAL - 640 x 480.
NTSC_1280x720 (65)	NTSC – 1280 x 720
NTSC_1280x900 (66)	NTSC – 1280 x 900
NTSC_1280x1024 (67)	NTSC – 1280 x 1024

NTSC_1920x1080 (68)	NTSC – 1920 x 1080

dwBitRate

BitRate	Description
BITRATE_28K (0)	28K Bits per second
BITRATE_56K (1)	56K Bits per second
BITRATE_128K (2)	128K Bits per second
BITRATE_256K (3)	256K Bits per second
BITRATE_384K (4)	384K Bits per second
BITRATE_500K (5)	500K Bits per second
BITRATE_750K (6)	750K Bits per second
BITRATE_1000K (7)	1M Bits per second
BITRATE_1200K (8)	1.2M Bits per second
BITRATE_1500K (9)	1.5M Bits per second
BITRATE_2000K (10)	2M Bits per second
BITRATE_2500K (11)	2.5M Bits per second
BITRATE_3000K (12)	3M Bits per second

dwVideoBrightness

0 ~ 100 : Low ~ High

dwVideoContrast

0 ~ 100 : Low ~ High

dwVideoSaturation

0 ~ 100 : Low ~ High

dwVideoHue

0 ~ 100 : Low ~ High

dwFps

0 ~ 30 frame pre second

MP4FILE_RECORD_INFO

The **MP4FILE_RECORD_INFO** structure is used to retrieve file record information after recording.

```
typedef struct structural_MP4FILE_RECORD_INFO
{
    time_t            tBeginTime;
    time_t            tEndTime;
    BYTE              btTimeZone;
    DWORD             dwGOP;
    DWORD             dwFrameCount;
    ULONGLONG         FileSize;
} MP4FILE_RECORD_INFO;
```

Members

tBeginTime

Begin time of record file.

tEndTime

End time of record file.

btTimeZone

Time zone of record file.

dwGOP

Number of GOP in the record file.

dwFrameCount

Number of frame in the record file.

FileSize

Size of the record file.

NOTIFY_AFTER_RENDER

The **NOTIFY_AFTER_RENDER** structure is used to notify render has finished.

```
typedef struct structural_NOTIFY_AFTER_RENDER
{
    HANDLE   AfterRenderEvent;
}NOTIFY_AFTER_RENDER;
```

Members

AfterRenderEvent

Event handle for After Render.

NOTIFY_DI

The **NOTIFY_DI** structure is used to notify DI event.

```
typedef struct structural_NOTIFY_DI
{
    HANDLE       DIEvent;
    BYTE         DI;
}NOTIFY_DI;
```

Members

DIEvent

Event handle to notify DI.

DI

DI information.

NOTIFY_IMAGE_REFRESH

The **NOTIFY_IMAGE_REFRESH** structure is used to notify change on image.

```
typedef struct structural_NOTIFY_IMAGE_REFRESH
{
    HANDLE   ImageRefreshEvent;
    void*    pImage;
    int      nFillLength;
}NOTIFY_IMAGE_REFRESH;
```

Members

ImageRefreshEvent

Event handle for Image Refresh.

pImage

Image.

nFillLength

Image Length.

NOTIFY_MOTION_DETECTION

The **NOTIFY_MOTION_DETECTION** structure is used to notify motion.

```
typedef struct structural_NOTIFY_MOTIONDETECTION
{
    HANDLE   MotionDetectionEvent;
    BYTE     MotionDetection;
}NOTIFY_MOTION_DETECTION;
```

Members

MotionDetectionEvent

Event handle for Motion Detect Event.

MotionDetection

Byte for which motion has been detected.

NOTIFY_NETWORK_LOSS

The **NOTIFY_NETWORK_LOSS** structure is used to notify network loss.

```
typedef struct structural_NOTIFY_NETWORKLOSS
{
    HANDLE   NetworkLossEvent;
}NOTIFY_NETWORK_LOSS;
```

Members

NetworkLossEvent

Event handle for Network Loss.

NOTIFY_RAWDATA_REFRESH

The **NOTIFY_RAWDATA_REFRESH** structure is used to notify raw data has changed.

```
typedef struct structural_NOTIFY_RAWDATAREFRESH
{
    HANDLE   RawDataRefreshEvent;
    void*    pBuffer;
    int      nFillLength;
}NOTIFY_RAWDATA_REFRESH;
```

Members

RawDataRefreshEvent

Event handle to notify raw data changed.

pBuffer

Buffer contain raw data.

nFillLength

Raw data length.

NOTIFY_RESOLUTION_CHANGE

The **NOTIFY_RESOLUTION_CHANGE** structure is used to notify resolution has changed.

```
typedef struct structural_NOTIFY_RESOLUTION_CHANGE
{
    HANDLE   ResolutionChangeEvent;
    int      nResolution;
}NOTIFY_RESOLUTION_CHANGE;
```

Members

ResolutionChangeEvent

Event handle for Resolution Change.

nResolution

New Resolution.

NOTIFY_RS232DATA_REFRESH

The **NOTIFY_RS232DATA_REFRESH** structure is used to notify RS232 data return.

```
typedef struct structural_NOTIFY_RS232DATA_REFRESH
{
    HANDLE   RS232DataRefreshEvent;
    void*    pBuffer;
    int      nFillLength;
}NOTIFY_RS232DATA_REFRESH;
```

Members

RS232DataRefreshEvent

Event handle for RS232 Data Refresh.

pBuffer

RS232 return data.

nFillLength

Length of return data.

NOTIFY_TIMECODE

The **NOTIFY_TIMECODE** structure is used to notify time code.

```
typedef struct structural_NOTIFY_TIMECODE
{
    HANDLE   TimeCodeEvent;
    DWORD    dwTimeCode;
}NOTIFY_TIMECODE;
```

Members

TimeCodeEvent

Event handle to notify time code.

dwTimeCode

Time code information.

NOTIFY_VIDEO_STATUS

The **NOTIFY_VIDEO_STATUS** structure is used to notify video status.

```
typedef struct structural_NOTIFY_VIDEOSTATUS
{
    HANDLE   VideoLossEvent;
    HANDLE   VideoRecoveryEvent;
}NOTIFY_VIDEO_STATUS;
```

Members

VideoLossEvent

Event handle for Video Loss.

VideoRecoveryEvent

Event handle for Video Recovery.

STREAMING_ENGINE_CONFIG

The **STREAMING_ENGINE_CONFIG** structure enable the streaming engine connection information.

```
typedef struct structural_STREAMING_ENGINE_CONFIG
{
    char     szUserID[16];
    char     szUserPwd[16];
    char     szServerIP[16];
    DWORD    dwStreamingPort;
    DWORD    dwControlPort;
}STREAMING_ENGINE_CONFIG;
```

Members

szUser

User ID for login Streaming Engine.

szUserPwd

User password for login Streaming Engine.

szServerIP

Streaming Engine IP address.

dwStreamingPort

Streaming port number for Streaming Engine.

dwControlPort

Control port number for Streaming Engine.

3 API Reference Guide

Initialization

Name	*Description*
`KCloseInterface`	Close SDK Interface
`KOpenInterface`	Open SDK Interface

KCloseInterface

KOpenInterace

Description

`KOpenInterface` and `KCloseInterface` are used for open and close SDK's Interface.

User call `HANDLE h = KOpenInterface();` to get the ip camera's object handle.

Then user can use the handle to deal with the IP Camera.

When the user wants to end the process, just call `KCloseInterface(h);` to delete the object.

Syntax

```
HANDLE   KOpenInterface (void);
void     KCloseInterface(HNADLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface.

Returns

Valid handle returned if success otherwise NULL.

Remarks

Check available memory for instance to allocate.

Requirements

Header file: **SDK-10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

```
HANDLE h = KOpenInterface();
.. .. ..
KCloseInterface(h);
```

See Also

(Back To Initialization List)

Connection

Name	*Description*
KConnect	Create a connection connects to IP Camera Server.
KDisconnect	Disconnect connection from IPCamera Server
KResetMediaConfig	Reset media configuration setting.
KSendControlCommand	Send command to video server through control port..
KSendURLCommand	Send URL command to video server
KSendURLCommandToDevice	Send URL command to device and get return result.
KSetEvent_NetworkLoss	Set event structural for network loss.
KSetMediaConfig	Set media configuration setting.
KSetNetworkLossCallback	Set callback function for newwork loss.

KConnect

Description

Create a connection and connects to IPCamera Server.

Syntax

```
bool KConnect(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by `KOpenInterface`.

Returns

If the function succeeds, then connect to video server.

If the function fails, fail to connect.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
```

```
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               . . . . .
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KDisconnect, (Back To Connection List)

KDisconnect

Description

Disconnect connection from IPCamera Server

Syntax

```
void KDisconnect(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
```

```
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               . . . . .
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KConnect, (Back To Connection List)

KResetMediaConfig

Description

Reset media configuration setting

Syntax

```
void KResetMediaConfig(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by `KOpenInterface`.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
if(NULL != h)
{
    KSetResetMediaConfig(h);
    . . . . .
}
```

See Also

KSetMediaConfig, (Back To Connection List)

KSendControlCommand

Description

Send control command to video server through control port.

Syntax

```
void KSendControlCommand(HANDLE h, DWORD dwCmdType, BYTE* ControlCommand
DWORD dwLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by `KOpenInterface()`.
dwCmdType	**DWORD**	[in] Command type
ControlCommand	**BYTE***	[in] Control command
dwLen	**DWORD**	[in] Control command length.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

See Also

KSendURLCommand, **(Back To Connection List)**

KSendURLCommand

Description

Send URL command to video server.

Syntax

```
void KSendURLCommand(HANDLE h, char* URLCommand, DWORD dwLen, char* ResultBuffer,
DWORD& ResultBufferLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by `KOpenInterface`.
URLCommand	**char***	[in] The url command string
dwLen	**DWORD**	[in] Length of URL Command
ResultBuffer	**char***	[in/out] The buffer prepare for get return data
ResultBufferLen	**DWORD&**	[in/out] The length of buffer and will return actual length of retuen bytes

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
```

```
        if(KConnect(h))
        {
          char szRequest[1024] = {0};
          char szAnswer[1024] = {0};
          DWORD nRet = 1024;
          sprintf(szRequest,"http://172.16.1.82:80/cgi-bin/system?
          USER=Admin&PWD=123456&V2_MULTICAST_IP");

          KSendURLCommand(h, szRequest, (DWORD)(strlen(szRequest)+1),
          szAnswer, nRet);
        }
    }
}
```

See Also

KSendControlCommand, KSendURLCommandToDevice,

(Back To Connection List)

KSendURLCommandToDevice

Description

Send URL command to device and get return result.

Syntax

```
bool KSendURLCommandToDevice(HANDLE h, char* IP, unsigned long HTTPPort, char*
URLCommand, DWORD dwURLCommandLen, char* ResultBuffer, DWORD&
dwResultBufferLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by `KOpenInterface`.
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port
URLCommand	**char***	[in] URL command.
dwURLCommandLen	**DWORD**	[in] URL command length.
ResultBuffer	**char***	[in/out] The buffer prepare for get return data
ResultBufferLen	**DWORD&**	[in/out] The length of buffer and will return actual length of retuen bytes

Returns

If function succeeds, then parse ResultBuffer for return URL result.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
if(NULL != h)
```

```
{
    char* ps = "http://172.16.1.82:80/cgi-bin/system?
    USER=Admin&PWD=123456&VIDEO_FPS\0";
    char szResult[2048] = {0};
    DWORD dwResult = 2048;
    KSendURLCommandToDevice(h, "172.16.1.82", 80, ps, strlen(ps),
    szResult, dwResult);
}
```

See Also

KSendControlCommand, KSendURLCommand, (Back To Connection List)

KSetEvent_NetworkLoss

Description

Set event structural for network loss.

Syntax

```
void KSetEvent_NetworkLoss(HANDLE h, NOTIFY_NETWORK_LOSS* nNetworkLoss);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nNetworkLoss	**NOTIFY_NETWORK_LOSS***	[in] Event structural for network loss.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Connection List)

KSetMediaConfig

Description

Set media configuration setting

Syntax

```
bool KSetMediaConfig(HANDLE h, MEDIA_CONNECTION_CONFIG* MediaConfig);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by `KOpenInterface`.
MediaConfig	**MEDIA_CONNECTION_CONFIG***	[in] Structure for connection setting.

Returns

If function succeeds, then media configuration set to SDK.

If function fails, call function KGetLastError to retrieve error code.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
```

```
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            . . . . .
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetLastError, KResetMediaConfig, (Back To Connection List)

KSetNetworkLossCallback

Description

Set callback function for network loss.

Syntax

```
void KSetNetworkLossCallback(HANDLE h, DWORD UserParam,
NETWORK_LOSS_CALLBACK fnNetworkLossCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] User parameter carry with callback.
fnNetworkLossCallback	**NETWORK_LOSS_CALLBACK**	[in] Pointer for callback function.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK NetWorkLossCB( DWORD UserParam )
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
```

```
    KSetNetworkLossCallback(h, (DWORD)this, NetworkLossCB);
    . . . . .
}
```

See Also

([Back To Connection List])

Stream

Name	*Description*
KEnableDecoder	Enable/Disable decoder.
KGetDeviceTypeByHTTP	Get device type using HTTP
KGetNumberOfChannelByHTTP	Get number of channel using HTTP.
KGetPortInfoByHTTP	Get video server port information using HTTP.
KGetTCPTypeByHTTP	Get stream format type using HTTP
KSetAfterRenderCallback	Set the callback to get the handle after SDK paints the video on the window.
KSetCODECType	Set CODEC type.
KSetControlDataCallback	Set callback function for control data.
KSetDecodeIFrameOnly	Set Flag to decode I frame only.
KSetEvent_AfterRender	Set event structural for after render.
KSetEvent_ImageRefresh	Set event structural for image refresh.
KSetEvent_RawDataRefresh	Set event structural for raw data refresh.
KSetEvent_ResolutionChange	Set event structural for resolution change.
KSetEvent_VideoStatus	Set event structural for video status.
KSetImageCallback	Set the callback to get the Image per Frame
KSetRawDataCallback	Set the CallBack Function to get the MPEG-4 raw data
KSetResolutionChangeCallback	Set the CallBack Function when the resolution changes
KSetSequenceHeaderChecker	Enable/Disable sequence header checker.
KSetTCPType	Set TCP type to SDK.
KSetVideoLossCallback	Set callback function for video loss.
KSetVideoRecoveryCallback	Set callback function for video recovery.
KStartStreaming	Start the Stream
KStop	Stop displaying.
KStopStreaming	Stop the Stream

KEnableDecoder

Description

To Enable/Disable decoder.

Syntax

```
void KEnableDecoder(HANDLE h, bool bEnableDecoder);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnableDecoder	**bool**	[in] Flag to enable/disable

Returns

No return value.

Remarks

True – Enable decoder.

False – Disable decoder.

If you don't need decoder in your program then it is recommend to call this function after KOpenInterface.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KEnableDecoder(h, true);
}
```

See Also

(**Back To Stream List**)

KGetDeviceTypeByHTTP

Description

Get device type using HTTP.

Syntax

```
int KGetDeviceTypeByHTTP (HANDLE h, char* IP, unsigned long HTTPPort
char* UID, char* PWD);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port number.
UID	**char***	[in] User account for login.
PWD	**char***	[in] Password for login.

Returns

Return value	**Description**
0	Fail to get device Type
1	StandAlong
2	RackMouont
3	Blade

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
```

```
strcpy(mcc.UniCastIP, "172.16.1.105\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
    int nType=KGetDeviceTypeByHTTP(h,mcc.UniCastIP,mcc.HTTPPort,mcc.UserID,
     mcc.Password);
}
```

See Also

(Back To Stream List)

KGetNumberOfChannelByHTTP

Description

Get number of channel on video server using HTTP.

Syntax

```
int KGetNumberOfChannelByHTTP (HANDLE h, char* IP, unsigned long HTTPPort,
char* UID, char* PWD);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port number.
UID	**char***	[in] User account for login.
PWD	**char***	[in] Password for login.

Returns

If function succeeds, then number of channel on video server returned.

Return 0 if function fails.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
```

```
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
    int nNo = KGetNumberOfChannelByHTTP(h, mcc.UniCastIP, mcc.HTTPPort,
     mcc.UserID, mcc.Password);
}
```

See Also

(Back To Stream List)

KGetPortInfoByHTTP

Description

Get port information on video server using HTTP.

Syntax

```
bool KGetPortInfoByHTTP (HANDLE h, char* IP, MEDIA_PORT_INFO* mri, unsigned long
HTTPPort, char* UID, char* PWD, unsigned int ChannelNO = 0);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
mri	**MEDIA_PORT_INFO**	[out] Structure to contain port information.
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port number.
UID	**char***	[in] User account for login.
PWD	**char***	[in] Password for login.
ChannelNO	**unsigned int**	[in] Channel number. Default is 0.

Returns

If function succeeds, then mri will contain port information.

Return false if function fails.

Remarks

Port information for different channel.

1 Channel				
Channel No.	Channel ID	TCP		RTP
		Video Port	Control Port	RTSP Port
1	N/A	6002	6001	7070

2 Channel			
Channel No.	Channel ID	TCP	RTP

		Video Port	Control Port	RTSP Port
1	1	6002	6001	7070
2	2	6004	6003	7072

4 Channel				
Channel No.	Channel ID	TCP		RTP
		Video Port	Control Port	RTSP Port
1	1	6050	6010	7070
2	2	6051	6011	7072
3	3	6052	6012	7074
4	4	6053	6013	7076

8 Channel				
Channel No.	Channel ID	TCP		RTP
		Video Port	Control Port	RTSP Port
1	1	6050	6010	7070
2	2	6051	6011	7072
3	3	6052	6012	7074
4	4	6053	6013	7076
5	5	6054	6014	7078
6	6	6055	6015	7080
7	7	6056	6016	7082
8	8	6057	6017	7084

16 Channel				
Channel No.	Channel ID	TCP		RTP
		Video Port	Control Port	RTSP Port
1	1	6050	6010	7070
2	2	6051	6011	7072
3	3	6052	6012	7074
4	4	6053	6013	7076
5	5	6054	6014	7078
6	6	6055	6015	7080
7	7	6056	6016	7082
8	8	6057	6017	7084
9	9	6058	6018	7086

10	10	6059	6019	7088
11	11	6060	6020	7090
12	12	6061	6021	7092
13	13	6062	6022	7094
14	14	6063	6023	7096
15	15	6064	6024	7098
16	16	6065	6025	7100

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
    MEDIA_PORT_INFO mpi;
    memset(&mpi, 0x00, sizeof(MEDIA_PORT_INFO));
    KGetPortInfoByHTTP(h, &mpi, mcc.UniCastIP, mcc.HTTPPort,
     mcc.UserID, mcc.Password, mcc.ChannelNumber);
}
```

See Also

(Back To Stream List)

KGetTCPTypeByHTTP

Description

Get stream format type

Syntax

```
int KGetTCPTypeByHTTP (HANDLE h, char* IP, unsigned long HTTPPort
char* UID, char* PWD, unsigned int ChannelNO = 0);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port number.
UID	**char***	[in] User account for login.
PWD	**char***	[in] Password for login.
ChannelNO	**unsigned int**	[in] Channel number. Default is 0.

Returns

Return value	**Description**
0	Fail to get TCP Type
1	TCP 1.0
2	TCP 2.0

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
```

```
strcpy(mcc.UniCastIP, "172.16.1.105\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
     int nType = KGetTCPTypeByHTTP(h, mcc.UniCastIP, mcc.HTTPPort, mcc.UserID,
      mcc.Password);
}
```

See Also

(Back To Stream List)

KSetAfterRenderCallback

Description

Set the callback to get the handle after SDK paints the video on the window

Syntax

```
void KSetAfterRenderCallBack(HANDLE h, DWORD UserParam, AFTERRENDER_CALLBACK
fnAfterRenderCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DOWRD**	[in] Custom param for carry to callback function
fnAfterRenderCallback	**AFTER_RENDER_CALLBACK**	[in] The pointer to the callback function

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
void CALLBACK AfterRenderCB( DWORD UserParam )
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
```

```
if(NULL != h)
{
    KSetAfterRenderCallback(h, (DWORD)this, AfterRenderCB);
    . . . . .
}
```

See Also

(Back To Stream List)

KSetCODECType

Description

Set CODEC type.

Syntax

```
void KSetCODECType(HANDLE h, int nType, int nChannel);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nType	**int**	[in] CODEC type.
nChannel	**int**	[in] Channel number.

Returns

No return value.

Remarks

CODEC Type	Description
XVIDCODEC (0)	XVID CODEC
FFMCODEC (1)	FFMPEG CODEC
P51CODEC (2)	PCI51 CODEC
IPPCODEC (3)	IPP CODEC
MJPEGCODEC (4)	Motion JPEG CODEC
IH264CODEC (5)	H.264 CODEC

Setting CODEC type will overwrite system auto detection. If the codec isn't match video source, that could lead crash.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetCODECType(h, XVIDCODEC, 0);
}
```

See Also

(Back To Stream List)

KSetControlDataCallback

Description

Set callback function for control data.

Syntax

```
void KSetControlDataCallback(HANDLE h, DWORD UserParam,
CONTROL_DATA_CALLBACK fnControlDataCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] User parameter carry with callback.
fnControlDataCallback	**CONTROL_DATA_CALLBACK**	[in] Pointer for callback function.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK ControlDatayCB(DWORD UserParam, DWORD dwDataType, BYTE* buf,
DWORD len)
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
```

```
{
    KSetControlDataCallback(h, (DWORD)this, ControlDataCB);
    . . . . .
}
```

See Also

(Back To Stream List)

KSetDecodeIFrameOnly

Description

Set flag to decode I frame only.

Syntax

```
void KSetDecodeIFrameOnly(HANDLE h, bool bDecodeIOnly);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bDecodeIOnly	**bool**	[in] Flag for decode

Returns

No return value.

Remarks

True – Decode I frame only.

False – Decode all frames.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetDecodeIFrameOnly(h, true);
}
```

See Also

(Back To Stream List)

KSetEvent_AfterRender

Description

Set event structural for after render.

Syntax

```
void KSetEvent_AfterRender(HANDLE h, NOTIFY_AFTER_RENDER* nAfterRender);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAfterRender	**NOTIFY_AFTER_RENDER***	[in] Event structural for after render.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Stream List)

KSetEvent_ImageRefresh

Description

Set event structural for ImageRefresh.

Syntax

```
void KSetEvent_ImageRefresh(HANDLE h, NOTIFY_IMAGE_REFRESH* nImageRefresh);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nImageRefresh	**NOTIFY_IMAGE_REFRESH***	[in] Event structural for image refresh.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Stream List)

KSetEvent_RawDataRefresh

Description

Set event structural for raw data refresh.

Syntax

```
void KSetEvent_RawDataRefresh(HANDLE h, NOTIFY_RAWDATA_REFRESH* nRawData);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nRawData	**NOTIFY_RAWDATA_REFRESH***	[in] Event structural for raw data refresh.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Stream List)

KSetEvent_ResolutionChange

Description

Set event structural for resolution change.

Syntax

```
void KSetEvent_ResolutionChange(HANDLE h, NOTIFY_RESOLUTION_CHANGE*
nResolutionChange);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nResolutionChange	**NOTIFY_RESOLUTION_CHANGE***	[in] Event structural for resolution change.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Stream List)

KSetEvent_VideoStatus

Description

Set event structural for video status.

Syntax

```
void KSetEvent_VideoStatus(HANDLE h, NOTIFY_VIDEO_STATUS* nVideoStatus);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nVideoStatus	**NOTIFY_VIDEO_STATUS***	[in] Event structural for video status.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Stream List)

KSetImageCallback

Description

Set the callback to get the Image per Frame

Syntax

```
void KSetImageCallback(HANDLE h, DWORD UserParam, IMAGE_CALLBACK
fnImageCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DOWRD**	[in] Custom param for carry to callback function
fnImageCallback	**IMAGE_CALLBACK**	[in] The pointer to the callback function

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
void CALLBACK ImageCB( DWORD UserParam, BYTE* buf, DWORD dwlen,  DWORD dwWidth,
DWORD dwHeight)
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
```

```
    KSetAfterRenderCallback(h, (DWORD)this, ImageCB);
    . . . . .
}
```

See Also

(Back To Stream List)

KSetRawDataCallback

Description

Set the CallBack Function to get the MPEG-4/MJPEG/H.264 raw data .

Syntax

```
void KSetRawDataCallback(HANDLE h, DWORD UserParam, RAW_DATA_CALLBACK
fnStatusCallback)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DOWRD**	[in] Custom param for carry to callback function
fnImageCallback	**IMAGE_CALLBACK**	[in] The pointer to the callback function

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
void CALLBACK RawDataCB( DWORD UserParam, DWORD dwDataType, BYTE* buf,
DWORD len)
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
```

```
    KSetAfterRenderCallback(h, (DWORD)this, RawDataCB);
    . . . . .
}
```

See Also

(Back To Stream List)

KSetResolutionChangeCallback

Description

Set the CallBack Function when the resolution changes.

Syntax

```
void  KSetResolutionChangeCallback(HANDLE h, DWORD UserParam,
RESOLUTION_CHANGE_CALLBACK fnResolutionChangeCallback)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DOWRD**	[in] Custom param for carry to callback function
fnResolutionChangeCallback	**RESOLUTION_CHANGE_CALLBACK**	[in] The pointer to the callback function

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
void CALLBACK ResolutionChangeCB( DWORD UserParam, int nResolution);
{
    . . . . . .
}
```

```
. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetAfterRenderCallback(h, (DWORD)this, ResolutionChangeCB);
    . . . . .
}
```

See Also

(Back To Stream List)

KSetSequenceHeaderChecker

Description

To Enable/Disable sequence header checker.

Syntax

```
void KSetSequenceHeaderChecker(HANDLE h, bool bEnable, DWORD dwTimerInSec);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnable	**bool**	[in] Flag to enable/disable
dwTimerInSec	**DWORD**	[in] Check period in second.

Returns

No return value.

Remarks

If flag set to true then raw data sequence header will be checked.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetSequenceHeaderChecker(h, true, 1);
}
```

See Also

(Back To Stream List)

KSetTCPType

Description

Set TCP type to SDK.

Syntax

```
void KSetTCPType (HANDLE h, int Type);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
Type	**int**	[in] TCP type

Returns

No return value.

Remarks

TCP Type	Description
1	TCP 1.0
2	TCP 2.0

Call this function if you know the TCP type of video server..

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetTCPType(h, 2);        // TCP 2.0
```

}

See Also

(Back To Stream List)

KSetVideoLossCallback

Description

Set callback function for video loss.

Syntax

```
void KSetVideoLossCallback(HANDLE h, DWORD UserParam, VIDEO_LOSS_CALLBACK
fnVideoLossCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] User parameter carry with callback.
fnVideoLossCallback	**VIDEO_LOSS_CALLBACK**	[in] Pointer for callback function.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK VideoLossCB(DWORD UserParam)
{
    . . . . . .
}

. . . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
```

```
    KSetVideoLossCallback(h, (DWORD)this, VideoLossCB);
    . . . . .
}
```

See Also

[KSetVideoRecoveryCallback](), ([Back To Stream List]())

KSetVideoRecoveryCallback

Description

Set callback function for video recovery.

Syntax

```
void KSetVideoRecoveryCallback(HANDLE h, DWORD UserParam,
VIDEO_RECOVERY_CALLBACK fnVideoRecoveryCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] User parameter carry with callback.
fnVideoRecoveryCallback	**VIDEO_RECOVERY_CALLBACK**	[in] Pointer for callback function.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK VideoRecoveryCB(DWORD UserParam)
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
```

```
{
    KSetVideoRecoveryCallback(h, (DWORD)this, VideoRecoveryCB);
    . . . . .
}
```

See Also

KSetVideoLossCallback, (Back To Stream List)

KStartStreaming

Description

Start the Stream

Syntax

```
bool KStartStreaming (HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

If the function succeeds, start receive stream.

If the function fails, no data receiving.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
```

```
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStreaming(h))
               {
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStreaming(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KStopStreaming, (Back To Stream List)

KStop

Description

Stop displaying.

Syntax

```
void KStop(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;
```

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                    KPlay(h);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KPlay, (Back To Stream List)

KStopStreaming

Description

Stop the Stream

Syntax

```
void netStopStreaming (HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
```

```
mcc.ConnectTimeOut = 3;

HANDLE h = KOpenInterface();
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStreaming(h))
               {
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStreaming(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KStartStreaming, (Back To Stream List)

Record

Name	*Description*
KSetFileWriterType	Set recorder write type to raw or avi.
KSetPrerecordTime	Set the Pre Recording Time
KStartRecord	Start the normal recording
KStopRecord	Stop the Normal Recording

KSetFileWriterType

Description

Set recorder write type to raw or avi.

Syntax

```
void KSetFileWriterType (HANDLE h, int nType);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nType	**int**	[in] Write type

Returns

No return value.

Remarks

nType	Description
FRAW (0)	Set write type to raw.
FAVI (1)	Set write type to avi.
FRAW2 (2)	Set write type to raw and generate index.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, FRAW.dll, FAVI.dll**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetFileWriterType(h, FRAW);
    . . . . .
}
```

See Also

(Back To Record List)

KSetPrerecordTime

Description

Set the Pre Recording Time

Syntax

```
void KSetPrerecordTime(HANDLE h, int nInSecond);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nInSecond	**DWORD**	[in] the pre recording time by second.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetPrerecordTime(h, 3);
    . . . . .
}
```

See Also

(Back To Record List)

KStartRecord

Description

Start the normal recording

Syntax

```
bool KStartRecord (HANDLE h, char* FileName);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
FileName	**char***	[in] the file name that save the recording data

Returns

If the function succeeds, then it is recording..

If the function fails, no file will create.

Remarks

In order to complete the recording KStopRecord must perform at end.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, FRAW.dll, FAVI.dll**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
```

```
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

HANDLE h = KOpenInterface();
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
                KPlay(h);
                KStartRecord(h, “c:\\rec.raw”);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    MP4FILE_RECORD_INFO mri;
    memset(&mri, 0x00, sizeof(MP4FILE_RECORD_INFO));
    KStopRecord(h, &mri);
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KStopRecord, (Back To Record List)

KStopRecord

Description

Stop the Normal Recording

Syntax

```
void KStopRecord (HANDLE h, MP4FILE_RECORD_INFO* mri);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
mri	**MP4FILE_RECORD_INFO***	[out] The record file information.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, FRAW.dll, FAVI.dll**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
```

```
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

HANDLE h = KOpenInterface();
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
                KPlay(h);
                KStartRecord(h, “c:\\rec.raw”);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    MP4FILE_RECORD_INFO mri;
    memset(&mri, 0x00, sizeof(MP4FILE_RECORD_INFO));
    KStopRecord(h, &mri);
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KStartRecord, (Back To Record List)

Audio

Name	*Description*
KFreeAudioToken	To release speak out session of audio
KGetAudioToken	To creat speak out session of audio to video server
KGetVolume	Get sound volume value from video server
KPlayTheAudioFromPCI5100ToPCSoundDevice	Play sound from PCI5100 to PC sound device.
KReadAudioFromPCI5100	Read audio from PCI5100.
KSendAudio	Function for send PCM data to video server
KSetMute	Set to change mute status to video server
KSetVolume	Set to change volume value to video server
KStartAudioTransfer	Send audio to video server.
KStopAudioTransfer	Stop send audio to video server.

KFreeAudioToken

Description

To release speak out session of audio.

Syntax

```
void KFreeAudioToken(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
char holderip[16] = {0};
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
```

```
KSetMediaConfig(h, &mcc);
if( h )
{
    if( KGetAudioToken( h, holderip ) )
    {
        if( KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . . .
if( h )
{
        KStopAudioTransfer( h );
        KFreeAudioToken( h );
}
```

See Also

KGetAudioToken, (Back To Audio List)

KGetAudioToken

Description

To creat speak out session of audio to video server.

Syntax

```
bool KGetAudioToken(HANDLE h, char* holder);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
holder	**char***	[out] Current user information.

Returns

If the function succeeds, then Aduio Token is get by current user.

If the function fails, holder holds the information of current user.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
char holderip[16] = {0};
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
```

```
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
KSetMediaConfig(h, &mcc);
if( h )
{
    if( KGetAudioToken( h, holderip ) )
    {
        if( KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . . .
if( h )
{
        KStopAudioTransfer( h );
        KFreeAudioToken( h );
}
```

See Also

KFreeAudioToken, (**Back To Audio List**)

KGetVolume

Description

Get sound volume value from video server

Syntax

```
bool KGetVolume(HANDLE h, int& nLeftVolume, int& nReigtVolume);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by netOpenInterface()
nLeftVolume	**int**	[out] Possible values for this property are from 100 to zero for left audio in channel. 100 specifies full volume and Zero specifies no volume.
nRightVolume	**int**	[out] Possible values for this property are from 100 to zero for right audio in channel. 100 specifies full volume and Zero specifies no volume.

Returns

If the function succeeds, then Left and Right Volume are returned

If the function fails, both Left and Right volume return zero.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
HANDLE h = KOpenInterface();
int nLeft;
int nRight;
if(NULL != h)
{
```

```
    KGetVolume(h, nLeft, nRight);
    . . . . .
}
```

See Also

KSetVolume, **KSetMute**, **(Back To Audio List)**

KPlayTheAudioFromPCI5100ToPCSoundDevice

Description

Play Audio from PCI5100 to PC sound device.

Syntax

```
bool  KPlayTheAudioFromPCI5100ToPCSoundDevice(HANDLE h, bool bPlay);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bPlay	**bool**	[in] Flag to play

Returns

If function return succeeds, then audio play otherwise no audio playing.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

See Also

(Back To Audio List)

KReadAudioFromPCI5100

Description

Read audio from PCI5100.

Syntax

```
bool  KReadAudioFromPCI5100(HANDLE h, BYTE* pBuffer, LONG& lBufferLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pBuffer	**BYTE***	[out] Buffer contain audio data.
lBufferLen	**LONG&**	[in/out] Buffer length and return data length.

Returns

If function return succeeds, then audio read from PCI5100 otherwise no audio read.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

See Also

(Back To Audio List)

KSendAudio

Description

Enable can send PCM data to video server.

Syntax

```
bool  KSendAudio(HANDLE h, BYTE* pAudioBuffer, int nlen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
pAudioBuffer	**BYTE***	[in] The buffer about 8K mono format PCM data
nLen	**int**	[in] The length about buffer

Returns

If the function succeeds, then Audio data is sent.

If the function fails, then no Audio data been send.

Remarks

KGetAudioToken() must called before this function.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
char holderip[16] = {0};
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
```

```
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
KSetMediaConfig(h, &mcc);
if( h )
{
    if( KGetAudioToken( h, holderip ) )
    {
        KSendAudio(h, pAdudioData, dwAudioDataLen);
    }
}

. . . . . .
if( h )
{
    KFreeAudioToken( h );
}
```

See Also

KGetAudioToken, KFreeAudioToken, (Back To Audio List)

KSetMute

Description

Set to change mute status to video server.

Syntax

```
void KSetMute(HANDLE h, bool bMute);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bMute	**bool**	[in] true for set to mute and false not

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetMute(h, true);
    . . . . .
}
```

See Also

KGetVolume, **KSetVolume**, **(Back To Audio List)**

KSetVolume

Description

Set to change volume value to video server.

Syntax

```
void KSetVolume(HANDLE h, int nLeftVolume , int nReigtVolume);
```

Parameters

Name	*Type*	*Description*
P	**HANDLE**	[in] The handle returned by KOpenInterface()
nLeftVolume	**int**	[in] Possible values for this property are from 100 to zero for set to left audio in channel. 100 specifies full volume and Zero specifies no volume.
nReigtVolume	**int**	[in] Possible values for this property are from 100 to zero for set to right audio in channel. 100 specifies full volume and Zero specifies no volume.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetVolume(h, 50, 50);
    . . . . .
}
```

See Also

KGetVolume, **KSetMute**, **(Back To Audio List)**

KStartAudioTransfer

Description

Send audio data to video server.

Syntax

```
bool  KStartAudioTransfer(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

If the function succeeds, then Audio data is sent.

If the function fails, then no Audio data been send.

Remarks

KGetAudioToken() must called before this function. To stop audio transfer, function KStopAudioTransfer must called.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
char holderip[16] = {0};
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
```

```
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
KSetMediaConfig(h, &mcc);
if( h )
{
    if( KGetAudioToken( h, holderip ) )
    {
        if( KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . . .
if( h )
{
        KStopAudioTransfer( h );
        KFreeAudioToken( h );
}
```

See Also

KStopAudioTransfer, KGetAudioToken, KFreeAudioToken,

(Back To Audio List)

KStopAudioTransfer

Description

Stop send audio data to video server.

Syntax

```
void  KStopAudioTransfer(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

No return value.

Remarks

KFreeAudioToken() should call if the token is no longer use by the user.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
char holderip[16] = {0};
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
```

```
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
KSetMediaConfig(h, &mcc);
if( h )
{
    if( KGetAudioToken( h, holderip ) )
    {
        if( KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . . .
if( h )
{
        KStopAudioTransfer( h );
        KFreeAudioToken( h );
}
```

See Also

KStartAudioTransfer, KGetAudioToken, KFreeAudioToken,

(Back To Audio List)

Playback

Name	*Description*
KAddMultipleMedia	Add a media file for multiple file playback.
KClearAllMultipleMedia	Remove all media files from “multiple file playback”.
KEnableFullScreen	To enable/disable the full screen mode.
KEnableStretchMode	To enable/disable the stretch mode for playback
KGetBeginTime	Get the begin time of the media file.
KGetCurrentReadingAbsTimeFromMultipleMedia	Get current absolute reading time.
KGetCurrentReadingFileIDFromMultipleMedia	Get current reading file ID.
KGetEndTime	Get the end time of the media file.
KGetNextIFrame	Step to next I frame.
KGetNthBeginTimeFromMultipleMedia	Get begin time from certain file by ID.
KGetNthEndTimeFromMultipleMedia	Get end time from certain file by ID.
KGetPrevIFrame	Step to previous I frame.
KGetTotalIFramesOfMultipleMedia	Get the number of total I frames from added files.
KPause	Pause playback
KPlay	Start to play the media file
KRemoveMultipleMedia	Remove a media file from “multiple file playback”.
KSetCurrentTime	Set the current file’s playback time (in seconds)
KSetEvent_TimeCode	Set event structural for time code.
KSetFilePlayCompeleteCallback	Set callback function for complete file playing.
KSetFilePlayCompleteCllback	Set function for while playback completed to do callback
KSetMultipleMediaConfig	Set media configure and enable multiple file playback mod.
KSetPlayDirection	Set playback direction.
KSetPlayRate	Set playback rate
KSetTimeCodeCallback	Set function for while playback on a new time to do callback
KSetTimeCodeCallbackEx	

KStepNextFrame	Set playback step next frame
KStepPrevFrame	Set playback step previous frame.

KAddMultipleMedia

Description

Add a media file for multiple file playback.

Syntax

```
bool KAddMultipleMedia( HANDLE h, DWORD Nth, char* strFileName);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
Nth	**DWORD**	[in] ID of file.
strFileName	**char***	[in] File name.

Returns

Return true, if success.

Remarks

1. The playing order is equal to ID order.
2. ID must be unique, and doesn't have to be continuous.
3. Must call Kconnect() after calling this API, or playback functions won't work.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

```
KSetMultipleMediaConfig(m_hKMpeg4 , &m_mcc);
KAddMultipleMedia( m_hKMpeg4, 3, filename2);
KAddMultipleMedia( m_hKMpeg4, 7, filename4);
KConnect(m_hKMpeg4)
```

See Also

KRemoveMultipleMedia (Back To Playback List)

KClearAllMultipleMedia

Description

Remove all media files from “multiple file playback”.

Syntax

```
void KClearAllMultipleMedia( HANDLE h );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

No return value.

Remarks

Must call Kconnect() and add other files after calling this API, or playback functions won’t work.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

See Also

(Back To Playback List)

KEnableFullScreen

Description

To enable/disable the full screen mode.

Syntax

```
void KEnableFullScreen(HANDLE h, bool bFullScreen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
bFullScreen	**bool**	[in] True – Enable, False – Disable.

Returns

No return value.

Remarks

This function can enable/disable full screen mode on the fly.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, DGDI.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
```

```
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KEnableFullScreen(h, true);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To Playback List)

KEnableStretchMode

Description

To enable/disable the stretch mode for playback

Syntax

```
void KEnableStretchMode(HANDLE h, bool bStretchMode);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
bStretchMode	**bool**	[in] True – Enable, False – Disable.

Returns

No return value.

Remarks

By default the stretch mode is enabled.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
```

```
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KPlay(h);
                  KEnableStretchMode(h, true);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

(Back To Playback List)

KGetBeginTime

Description

Get the begin time of the media file.

Syntax

```
void KGetBeginTime(HANDLE h, DWORD& dwBeginTime);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwBeginTime	**DWORD**	[out] Begin time of the media file.

Returns

No return value.

Remarks

Time zone is included in dwBeginTime.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
```

```
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               DWORD dwBeginTime;
               DWORD  dwEndTime;
               KGetBeginTime(h, dwBeginTime);
               KGetEndTime(h, dwEndTime);
               KPlay(h);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetEndTime, (Back To Playback List)

KGetCurrentReadingAbsTimeFromMultipleMedia

Description

Get current absolute reading time. (The range is from 0 to sum of all file's duration.)

Syntax

```
void KGetCurrentReadingAbsTimeFromMultipleMedia( HANDLE h, DWORD& dwABSTime);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwABSTime	**DWORD&**	[out] The time in second

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

See Also

KGetNthEndTimeFromMultipleMedia (Back To Playback List)

KGetCurrentReadingFileIDFromMultipleMedia

Description

Get current reading file ID.

Syntax

```
void KGetCurrentReadingFileIDFromMultipleMedia( HANDLE h, DWORD& Nth);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
Nth	**DWORD&**	[out] File ID

Returns

No return value.

Remarks

Find which file be read after “Kplay” function.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

See Also

KGetCurrentReadingAbsTimeFromMultipleMedia (Back To Playback List)

KGetEndTime

Description

Get the end time of the media file.

Syntax

```
void KGetEndnTime(HANDLE h, DWORD& dwEndTime);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwEndTime	**DWORD**	[out] End time of the media file.

Returns

No return value.

Remarks

Time zone is included in dwEndTime.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
```

```
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               DWORD dwBeginTime;
               DWORD  dwEndTime;
               KGetBeginTime(h, dwBeginTime);
               KGetEndTime(h, dwEndTime);
               KPlay(h);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetBeginTime, (Back To Playback List)

KGetNextIFrame

Description

Set playback step next I frame.

Syntax

```
bool KGetNextIFrame(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

Return true, if step to next I frame, otherwise return false.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
// need to set play status pause for play step frame
KPause( h );

KGetNextFrame( h );

. . . . .
```

See Also

KStepPrevFrame, **KPause**, **KStepNextFrame**, **KGetPrevIFrame**,

(Back To Playback List)

KGetNthBeginTimeFromMultipleMedia

Description

Get begin time from certain file by ID.

Syntax

```
void KGetNthBeginTimeFromMultipleMedia( HANDLE h, DWORD Nth,  DWORD&
     dwBeginTime );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
Nth	**DWORD**	[in]File ID
dwBeginTime	**DWORD&**	[out] Begin time of Nth file.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

```
char szBegin[64];
KGetNthBeginTimeFromMultipleMedia( m_hKMpeg4, 1,  m_dwBeginTime );
GetTimeToStr(m_dwBeginTime, szBegin);
```

See Also

KGetNthEndTimeFromMultipleMedia (Back To Playback List)

KGetNthEndTimeFromMultipleMedia

Description

Get end time from certain file by ID.

Syntax

```
void KGetNthEndTimeFromMultipleMedia( HANDLE h, DWORD Nth, DWORD& dwEndTime );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
Nth	**DWORD**	[in]File ID
dwEndTime	**DWORD&**	[out] End time of Nth file.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

```
char szEnd[64];
KGetNthEndTimeFromMultipleMedia( m_hKMpeg4, 3, m_dwEndTime );
GetTimeToStr(m_dwEndTime, szEnd);
```

See Also

KGetNthBeginTimeFromMultipleMedia (Back To Playback List)

KGetPrevIFrame

Description

Set playback step previous I frame.

Syntax

```
bool KGetPrevIFrame(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

Return true, if step to prev I frame, otherwise return false.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
// need to set play status pause for play step frame
KPause( h );

KGetPrevIFrame( h );

. . . . .
```

See Also

KStepPrevFrame, **KPause**, **KStepNextFrame**, **KGetNextIFrame**,

(Back To Playback List)

KGetTotalIFramesOfMultipleMedia

Description

Get the number of total I frames from added files.

Syntax

```
void KGetTotalIFramesOfMultipleMedia( HANDLE h, DWORD& dwIFramesNumber );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwIFramesNumber	**DWORD&**	[out] The number of I frames.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

See Also

(Back To Playback List)

KPause

Description

Pause playback.

Syntax

```
void KPause(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

No return value.

Remarks

You can re-start via calling **KPlay.**

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
```

```
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

HANDLE h = KOpenInterface();
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
                KPlay(h);
            }
        }
    }
}

if(NULL != h)
    KPause(h);
```

See Also

KPlay, (Back To Playback List)

KPlay

Description

Start to play the media file or streaming.

Syntax

```
void KPlay(HANDLE  h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

No return value.

Remarks

You can pause the playback by calling **KPause.**

If it is streaming then call this function to start display.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105 \0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
//mcc.ContactType = CONTACT_TYPE_PLAYBACK;  // Use this type for playback.
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
```

```
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeOut = 3;
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

HANDLE h = KOpenInterface();
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
                KPlay(h);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KPause, (Back To Playback List)

KRemoveMultipleMedia

Description

Remove a media file from “multiple file playback”.

Syntax

```
void KRemoveMultipleMedia( HANDLE h, DWORD Nth);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
Nth	**DWORD**	[in] ID of file

Returns

No return value.

Remarks

Must call Kconnect() after calling this API, or playback functions won’t work.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

```
KSetMultipleMediaConfig(m_hKMpeg4 , &m_mcc);
KAddMultipleMedia( m_hKMpeg4, 3, filename2);
KAddMultipleMedia( m_hKMpeg4, 7, filename4);
KConnect(m_hKMpeg4)
.....
KRemoveMultipleMedia(m_hKMpeg4, 7);
KConnect(m_hKMpeg4)
```

See Also

KAddMultipleMedia (Back To Playback List)

KSetCurrentTime

Description

Set the current file's playback time (in seconds).

Syntax

```
void KSetCurrentTime(HANDLE h, DWORD dwTimeCode);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwTimeCode	**DWORD**	[in] The time in seconds.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
```

```
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
DWORD dwBeginTime;
KGetBeginTime(h, dwBeginTime);
KSetCurrentTime(h, dwBeginTime);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetBeginTime, (Back To Playback List)

KSetEvent_TimeCode

Description

Set event structural for time code.

Syntax

```
void KSetEvent_TimeCode(HANDLE h, NOTIFY_TIMECODE* nTimeCode);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
ntimeCode	**NOTIFY_TIMECODE***	[in] Event structural for time code.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Playback List)

KSetFilePlayCompeleteCallback

Description

Set callback function for complete file playing.

Syntax

```
Void KSetFilePlayCompeleteCallback( HANDLE h,
        DWORD UserParam,
        FILE_PLAY_COMPLETE_CALLBACK fnFilePlayCompeleteCallback );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
UserParam	**DWORD**	[in] Custom param for carry to callback function
fnFilePlayCompeleteCallback	**FILE_PLAY_COMPLETE_CALLBACK**	[in] function pointer for callback

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Playback List)

KSetFilePlayCompleteCallback

Description

Set function for while playback completed to do callback

Syntax

```
void KSetFilePlayCompleteCallback(HANDLE h, DWORD UserParam,
FILE_PLAY_COMPLETE_CALLBACK fnFilePlayCompleteCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
UserParam	**DWORD**	[in] Custom param for carry to callback function
fnFilePlayCompleteCallback	**FILE_PLAY_COMPLETE_CALLBACK**	[in] function pointer for callback

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
void CALLBACK FilePlayCompleteCB(DWORD UserParam)
{
    . . . . . .
}

. . . . . .
HANDLE h = KOpenInterface();
```

```
if(NULL != h)
{
    KSetFilePlayCompleteCallback(h, (DWORD)this, FilePlayCompleteCB);
    . . . . .
}
```

See Also

(Back To Playback List)

KSetMultipleMediaConfig

Description

Set media configure and enable multiple file playback mod. (Use this function instead of "KSetMediaConfig" . Multiple files playback only work with “raw” file and “.idx” file in pairs by now. V1.2)

Syntax

```
bool KSetMultipleMediaConfig( HANDLE h, MEDIA_CONNECTION_CONFIG* pMediaconfig );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pMediaConfig	**MEDIA_CONNECTION_CONFIG***	[in] Structure for connection setting.

Returns

Return true, if success.

Remarks

In “multiple file playback mod”, The KGetEndTime() returns sum of added files’ duration (in second). And the KGetBeginTime() returns 0 always.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib, AMRAW.lib**

Runtime DLL: **KMpeg4.dll, AMRAW.dll**

Example

```
MEDIA_CONNECTION_CONFIG m_mcc;
memset(&m_mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
m_mcc.ContactType = CONTACT_TYPE_PLAYBACK;

KSetMultipleMediaConfig(m_hKMpeg4 , &m_mcc);
KAddMultipleMedia( m_hKMpeg4, 1, filename);
KConnect(m_hKMpeg4)
```

See Also

(Back To Playback List)

KSetPlayDirection

Description

Set playback direction.

Syntax

```
void KSetPlayDirection(HANDLE h, bool bForward);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bForward	**bool**	[in] True – Forward, False – Backward.

Returns

No return value.

Remarks

Only I frame will display when play backward direction.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
```

```
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KSetPlayDirection(h, false);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To Playback List)

KSetPlayRate

Description

Set playback rate.

Syntax

```
void KSetPlayRate(HANDLE h, int nRate);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nRate	**int**	[in] RATE_0_5 (0) – 1/2 Speed. RATE_1_0 (1) – 1.0 Speed. RATE_2_0 (2) – 2.0 Speed. RATE_4_0 (3) – 4.0 Speed. RATE_8_0 (4) – 8.0 Speed.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
```

```
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KSetPlayRate(h, RATE_2_0);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To Playback List)

KSetSmoothFastPlayback

Description

Set smooth fast playback.

Syntax

```
void KSetSmoothFastPlayback (HANDLE h, bool bIsSmoothFastPlayback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
bIsSmoothFastPlayback	**bool**	[in] Flag to enable/disable.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
```

```
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KPlay(h);
               }
          }
      }
}
. . . . .
KSetSmoothFastPlayback(h,true);
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

(Back To Playback List)

KSetTimeCodeCallback

Description

Set function for while playback on a new time to do callback

Syntax

```
void KSetTimeCodeCallback(HANDLE h, DWORD UserParam,
                          TIME_CODE_CALLBACK fnTimeCodeCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
UserParam	**DWORD**	[in] Custom param for carry to callback function.
fnTimeCodeCallback	**TIME_CODE_CALLBACK**	[in] Function point for time code callback.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
void CALLBACK TimeCodeCB(DWORD UserParam, DWORD dwTimeCode)
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
```

```
{
    KSetTimeCodeCallback(h, (DWORD)this, TimeCodeCB);
    . . . . .
}
```

See Also

(Back To Playback List)

KSetTimeCodeCallbackEx

Description

Set function for while playback on a new time to do callback. (in millisecond)

Syntax

```
void KSetTimeCodeCallbackEx( HANDLE h, DWORD UserParam, TIME_CODE_CALLBACK_EX
                             fnTimeCodeCallbackEx );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
UserParam	**DWORD**	[in] Custom param for carry to callback function.
fnTimeCodeCallbackEx	**TIME_CODE_CALLBACK_EX**	[in] Function point for time code callback.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

See Also

(Back To Playback List)

KStepNextFrame

Description

Set playback step next frame.

Syntax

```
void KStepNextFrame(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

No return value.

Remarks

Function KPause must called before this function.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
// need to set play status pause for play step frame
KPause( h );

KStepNextFrame( h );

. . . . .
```

See Also

KStepPrevFrame, **KPause**, **(Back To Playback List)**

KStepPrevFrame

Description

Set playback step previous frame.

Syntax

```
void KStepPrevFrame(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

No return value.

Remarks

Function KPause must called before this function.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

```
// need to set play status pause for play step frame
KPause( h );

KStepPrevFrame( h );

. . . . .
```

See Also

KStepNextFrame, **KPause**, **(Back To Playback List)**

RS-232/422/485 Control

Name	*Description*
KSendRS232Command	Send RS232 command.
KSendRS232Setting	Setup the Server's RS232 X81 and BaudRate
KSetEvent_RS232DataRefresh	Set event structural for RS232 data refresh.
KSetRS232DataCallback	Set the callback to receive the Server's RS232 Input

KSendRS232Command

Description

Send RS232 command.

Syntax

```
void netSendKeyPadCommand (HANDLE h, BYTE* cmd, DWORD len);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
cmd	**BYTE***	[in] RS232 command
len	**DWORD**	[in] the command length.

Returns

No return value.

Remarks

User may have to call KSendRS232Setting before perform this function.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
```

```
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KSendRS232Setting(h, RS232_SET_N81, BAUD_RATE_9600BPS, 0);
KSendRS232Command(h, szRS232command, dwRS232CommandLength);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KSendRS232Command, (Back To RS-232/422/485 Control List)

KSendRS232Setting

Description

Setup the Server's RS232 X81 and BaudRate

Syntax

```
void KsendRs232Setting(HANDLE h, BYTE c81, BYTE dwBaudRate, int nComNumber);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
c81	**BYTE**	[in] The None, Even, Odd Parity.
dwBaudRate	**BYTE**	[in] The Baudrate
nComNumber	**int**	[in] Com port number

Returns

No return value.

Remarks

c81	**Description**
RS232_SET_N81 (0x00)	RS232 Setting N81
RS232_SET_O81 (0x08)	RS232 Setting O81
RS232_SET_E81 (0x18)	RS232 Setting E81

BaudRate	**Description**
BAUD_RATE_1200BPS (0)	1200 BPS
BAUD_RATE_2400BPS (1)	2400 BPS
BAUD_RATE_4800BPS (2)	4800 BPS
BAUD_RATE_9600BPS (3)	9600 BPS
BAUD_RATE_19200BPS (4)	19200 BPS
BAUD_RATE_38400BPS (5)	38400 BPS

BAUD_RATE_57600BPS	(6)	57600 BPS
BAUD_RATE_115200BPS	(7)	115200 BPS
BAUD_RATE_230400BPS	(8)	230400 BPS

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KSendRS232Setting(h, RS232_SET_N81, BAUD_RATE_9600BPS, 0);
KSendRS232Command(h, szRS232command, dwRS232CommandLength);
. . . . .
if(NULL != h)
{
    KStop(h);
```

```
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(**Back To RS-232/422/485 Control List**)

KSetEvent_RS232DataRefresh

Description

Set event structural for RS232 data refresh.

Syntax

```
void KSetEvent_RS232DataRefresh(HANDLE h, NOTIFY_RS232DATA_REFRESH*
nRS232Data);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nRS232Data	**NOTIFY_RS232DATA_REFRESH***	[in] RS232 Data event refresh structural.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

See Also

(Back To RS-232/422/485 Control List)

KSetRS232DataCallback

Description

Set the callback to receive the Server's RS232 Input

Syntax

```
void KSetRS232DataCallback(HANDLE h, DWORD UserParam, RS232_DATA_CALLBACK fnRS232Callback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] User parameter carry with callback function.
fnRS23Callback	**RS232_DATA_CALBACK**	[in] The pointer to the callback function

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
void CALLBACK RS232DataCB(DWORD UserParam, BYTE* pbuf, DWORD dwBufLen)
{
    . . . . . .
}

. . . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
```

```
{
    KSetRS232DataCallback(h, (DWORD)this, RS232DataCB);
    . . . . .
}
```

See Also

(Back To RS-232/422/485 Control List)

PTZ

These functions were removed from SDK10000 v1.2

Name	*Description*
PTZOpenInterface	PTZOpenInterface is used to open PTZ SDK's interface
PTZOpenInterfaceWithFile	PTZOpenInterface is used to open PTZ SDK's interface with file name input.
PTZCloseInterface	PTZCloseInterface is used to close PTZ SDK's interface
PTZGetString	Get hex command string from PTZ SDK and can be sent by serial port.
PTZGetStringURL	Get a PTZ command string by PTZ protocol file command.
GetURLCommand	Concate URL command with PTZ command.
PTZEnumerate	Enumerate PTZ information.
PTZEnumerateProtocol	Enumerate vender protocol information.

SDK10000 v1.2 provide these PTZ functions

Name	*Description*
KEnablePTZProtocol	Set PTZ (Pan Tilt Zoom) Protocol enabled or disabled.
KPTZBLC	PTZ (Pan Tilt Zoom) Back light compensation function.
KPTZDayNight	PTZ (Pan Tilt Zoom) Day/Night Mode switch function.
KPTZDegreeToUnit	Change drgrees to the units of hardware.
KPTZEnumerateFunctions	Return true when function success.
KPTZEnumerateProtocol	Get the protocol by the name of vender from ptz file.
KPTZEnumerateVender	Get the name of vender from ptz file.
KPTZFocus	PTZ (Pan Tilt Zoom) Focus function.
KPTZGetAbsPTZCommand	Get Absolute PTZ command string from PTZ protocol file by degrees.
KPTZGetAbsPTZCommandByUnit	Get PTZ command string from PTZ protocol file by the unit on hardware.
KPTZGetCommand	Get PTZ command string from PTZ protocol file
KPTZGetRequestAbsPTZCommand	Get Request PTZ status command. Send the command to device, the camera will send back PTZ status buffer from RS232 callback.
KPTZGetUnitFromBuffer	Get camera PTZ status from buffer.
KPTZIris	PTZ (Pan Tilt Zoom) Iris function.

KPTZLoadProtocol	Load PTZ (Pan Tilt Zoom) Protocol.
KPTZMove	PTZ (Pan Tilt Zoom) Move function.
KPTZOSD	PTZ (Pan Tilt Zoom) OSD (On Screen Display) function.
KPTZPreset	PTZ (Pan Tilt Zoom) Preset function.
KPTZUnitToDegree	Change the units of hardware to drgrees.
KPTZUnloadProtocol	Unload PTZ (Pan Tilt Zoom) Protocol.
KPTZZoom	PTZ (Pan Tilt Zoom) Zoom function.
KSendPTZCommand	Send PTZ command.

KEnablePTZProtocol

Description

Set PTZ (Pan Tilt Zoom) Protocol enabled or disabled.

Syntax

```
bool KEnablePTZProtocol(HANDLE h, bool bEnable);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
bEnable	**bool**	[in] Set bEnable true for enabling, false for disabling.

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

(Back To PTZ List)

KPTZBLC

Description

PTZ (Pan Tilt Zoom) Back light compensation function.

Syntax

```
bool KPTZBLC(HANDLE h, int nAddrID,  PTZ_BLC_OPERATION PTZBLCOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] Specify the address ID.
PTZBLCOP	**PTZ_BLC_OPERATION**	[in] On/Off

Returns

Return true when function success.

Remarks

```
enum PTZ_BLC_OPERATION
{
    PTZ_BLC_ON,
    PTZ_BLC_OFF
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZDayNight ,(Back To PTZ List)

KPTZDayNight

Description

PTZ (Pan Tilt Zoom) Day/Night Mode switch function.

Syntax

```
bool KPTZDayNight(HANDLE h, int nAddrID,  PTZ_DN_OPERATION PTZDNOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] Specify the address ID.
PTZDNOP	**PTZ_DN_OPERATION**	[in] 4 options, see the Remark section.

Returns

Return true when function success.

Remarks

```
enum PTZ_DN_OPERATION
{
    PTZ_DN_ON,
    PTZ_DN_OFF,
    PTZ_DN_AUTO_ON,
    PTZ_DN_AUTO_OFF
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZBLC ,(Back To PTZ List)

KPTZDegreeToUnit

Description

Change drgrees to the units of hardware.

Syntax

```
void KPTZDegreeToUnit( HANDLE h, float fPanDegree, float fTiltDegree, float
fZoomRatio, int& iPanUnit, int& iTiltUnit, int& iZoomUnit );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
fPanDegree	**float**	[in] Pan degree
fTiltDegree	**float**	[in] Tilt degree
fZoomRatio	**float**	[in] Zoom degree
iPanUnit	**int&**	[out] Pan unit
iTiltUnit	**int&**	[out] Tilt unit
iZoomUnit	**int&**	[out] Zoom unit

Returns

Return true when function success.

Remarks

Change drgrees to the units of hardware by Linear interpolation method.

The detail is defined in ptz file.

For example:

#DynaColor_DynaColor.ptz

PMAX; 1600

PMIN; 1

PMAXDEGREE; 360

TMAX is set at 90 degree

TMAXDEGREE; 90

TMAX; 223

TMIN is set at 0 degree

TMIN; 23

ZMAX; 37

ZMIN; 1

Requirements

Header file: `SDK10000.h`

Import library: `KMpeg4.lib`

Runtime DLL: `KMpeg4.dll, PTZParser.dll`

Example

See Also

KPTZUnitToDegree , (Back To PTZ List)

KPTZEnumerateFunctions

Description

Get functions from ptz file.

Syntax

```
bool KPTZEnumerateFunctions(HANDLE h, char* pFunctions, DWORD& dwLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pFunctions	**char***	[in/out] in: NULL string buffer. out: functions from ptz.
dwLen	**DWORD&**	[in/out] in: The size of input NULL string. out: The used size of pFunctions.

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

(Back To PTZ List)

KPTZEnumerateProtocol

Description

Get the protocol by the name of vender from ptz file.

Syntax

```
bool KPTZEnumerateProtocol(HANDLE h, char* pVender, char* pProtocol, DWORD& dwLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pVender	**char***	[in] The name of vender.
pProtocol	**char***	[out]The name of protocol.
dwLen	**DWORD&**	[out]The string length of pProtocol.

Returns

Return true when function success.

Remarks

For example : Color_yRoll.ptz

The name of vender is “Color” , and the protocol is “yRoll”.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZEnumerateVender ,(Back To PTZ List)

KPTZEnumerateVender

Description

Get the name of vender from ptz file.

Syntax

bool KPTZEnumerateVender(HANDLE h, char* pVender, DWORD& dwLen);

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pVender	**char***	[out] Get the name of vender
dwLen	**DWORD&**	[out]The string length of pVender

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZEnumerateProtocol ,(Back To PTZ List)

KPTZFocus

Description

PTZ (Pan Tilt Zoom) Focus function.

Syntax

```
bool KPTZFocus(HANDLE h, int nAddrID,  PTZ_FOCUS_OPERATION PTZFocusOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] Specify the address ID
PTZFocusOP	**PTZ_FOCUS_OPERATION**	[in] in/out/stop

Returns

Return true when function success.

Remarks

```
enum PTZ_FOCUS_OPERATION
{
    PTZ_FOCUS_IN,
    PTZ_FOCUS_OUT,
    PTZ_FOCUS_STOP
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

(Back To PTZ List)

KPTZGetAbsPTZCommand

Description

Get Absolute PTZ command string from PTZ protocol file by degrees.

Syntax

```
bool KPTZGetAbsPTZCommand( HANDLE h, char* pPTZCmd, int iParam1, int iParam2,
bool bPanCounterClock, float fPanDegree, float fTiltDegree, float fZoomRatio,
BYTE* pCommand, DWORD& dwCommandLen );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pPTZCmd	**char***	[in]The string indicate PTZ operation
iParam1	**int**	[in]
iParam2	**int**	[in] Always 0. It's reserved parameter.
bPanCounterClock	**bool**	[in]Pan the camera in ccw or not.
fPanDegree	**float**	[in]The destination of Pan.
fTiltDegree	**float**	[in]The destination of Tile.
fZoomRatio	**float**	[in]The destination of Zoom. (0~100)
pCommand	**BYTE***	[in/out]empty buffer/BYTEs of PTZ command
dwCommandLen	**DWORD&**	[in/out]size of empty buffer/size of PTZ command bytes

Returns

Return true when function success.

Remarks

Absolute PTZ commands only work with DynaColor protocols at present (V1.2),

and the nParam1 is always 0 in DynaColor ptz files.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

```
BYTE bCommand[128] = {0};
DWORD dwLen = 0;
bool IsCCW = false;
KPTZGetAbsPTZCommand(m_hNet,
      "SETABSOLUTEPTZ",
      0,
      0,
      IsCCW,
      m_fPanDegree,
      m_fTilteDegree,
      m_fZoomDegree,
      bCommand,
      dwLen);
KSendPTZCommand(m_hNet, bCommand, dwLen);
```

See Also

KPTZGetAbsPTZCommandByUnit , (**Back To PTZ List**)

KPTZGetAbsPTZCommandByUnit

Description

Get PTZ command string from PTZ protocol file by the unit on hardware.

Syntax

```
bool KPTZGetAbsPTZCommandByUnit( HANDLE h, char* pPTZCmd, int iParam1, int iParam2,
bool bPanCounterClock, int iPanUnit, int iTiltUnit, int iZoomUnit, BYTE* pCommand,
DWORD& dwCommandLen );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pPTZCmd	**char***	[in]The string indicate PTZ operation
iParam1	**int**	[in]
iParam2	**int**	[in] Always 0. It's reserved parameter.
bPanCounterClock	**bool**	[in]Pan the camera in ccw or not.
iPanUnit	**int**	[in] The destination of Pan.
iTiltUnit	**int**	[in] The destination of Tilt.
iZoomUnit	**int**	[in] The destination of Zoom.
pCommand	**BYTE***	[in/out]empty buffer/BYTEs of PTZ command
dwLen	**DWORD&**	[in/out]size of empty buffer/size of PTZ command bytes

Returns

Return true when function success.

Remarks

Absolute PTZ commands only work with DynaColor protocols at present (V1.2),

and the nParam1 is always 0 in DynaColor ptz files.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

```
BYTE bCommand[128] = {0};
DWORD dwLen = 0;
bool IsCCW = false;
KPTZGetAbsPTZCommandByUnit(m_hNet,
         "SETABSOLUTEPTZ",
         0,
         0,
         IsCCW,
         89,
         40,
         3,
         bCommand,
         dwLen);

KSendPTZCommand(m_hNet, bCommand, dwLen);
```

See Also

KPTZGetAbsPTZCommand , (Back To PTZ List)

KPTZGetCommand

Description

Get PTZ command string from PTZ protocol file.

Syntax

```
bool KPTZGetCommand(HANDLE h, char* pPTZCmd, int nAddrID, int nParam1, int nParam2,
BYTE* bCmd, DWORD& dwLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pPTZCmd	**char***	[in]The string indicate PTZ operation
nAddrID	**int**	[in]Camera address ID
nParam1	**int**	[in]
nParam2	**int**	[in]
bCmd	**BYTE***	[in/out]empty buffer/BYTEs of PTZ command
dwLen	**DWORD&**	[in/out]size of empty buffer/size of PTZ command bytes

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

(Back To PTZ List)

KPTZGetRequestAbsPTZCommand

Description

Get Request PTZ status command. Send the command to device, the camera will send back PTZ status buffer from RS232 callback.

Syntax

```
bool KPTZGetRequestAbsPTZCommand(HANDLE h, int iParam1,BYTE* pCommand,DWORD&
dwCommandLen);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
iParam1	**int**	[in]
pCommand	**BYTE***	[in/out]empty buffer/command buffer
dwCommandLen	**DWORD&**	[in/out]size of empty buffer/size of command

Returns

Return true when function success.

Remarks

Gather the buffer from RS232CallBack , analyse the buffer by KPTZGetUnitFromBuffer later.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZGetUnitFromBuffer , (Back To PTZ List)

KPTZGetUnitFromBuffer

Description

Get camera PTZ status from buffer.

Syntax

```
bool KPTZGetUnitFromBuffer( HANDLE h, BYTE* pDataBufferFromRS232CallBack, DWORD
dwLengthOfBuffer, int& iPanUnit, int& iTiltUnit, int& iZoomUnit );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pDataBufferFromRS232CallBack	**BYTE***	[in] Collected buffer from RS232CallBack
dwLengthOfBuffer	**DWORD**	[in]The length of input buffer
iPanUnit	**int&**	[out]Pan status of camera
iTiltUnit	**int&**	[out]Tilt status of camera
iZoomUnit	**int&**	[out]Zoom status of camera

Returns

Return true when function success.

Remarks

Gather the buffer from RS232CallBack first, analyse the buffer by this function second. Concatenate the buffer long enough, this function could parse out latest Pan, Tilt, and Zoom status. (To filter out other information, ex: iPanUnit &= 0x00007fff.)

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZGetRequestAbsPTZCommand , (Back To PTZ List)

KPTZIris

Description

PTZ (Pan Tilt Zoom) Iris function.

Syntax

```
bool KPTZIris(HANDLE h, int nAddrID,  int nParam1, PTZ_IRIS_OPERATION PTZIrisOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] Specify the address ID
nParam1	**int**	[in]
PTZIrisOP	**PTZ_IRIS_OPERATION**	[in]4 options, see the Remark section

Returns

Return true when function success.

Remarks

```
enum PTZ_IRIS_OPERATION
{
    PTZ_IRIS_OPEN,
    PTZ_IRIS_CLOSE,
    PTZ_IRIS_STOP,
    PTZ_IRIS_AUTO
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

(**Back To PTZ List**)

KPTZLoadProtocol

Description

Load PTZ (Pan Tilt Zoom) Protocol.

Syntax

```
bool KPTZLoadProtocol(HANDLE h, MEDIA_PTZ_PROTOCOL* pMPP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
pMPP	**MEDIA_PTZ_PROTOCOL***	[in] Which specify the protocol resource.

Returns

Return true when function success.

Remarks

```
typedef struct structural_MEDIA_PTZ_PROTOCOL
{
     int nSourceType;    // [in]Specify the source type is  inside resource
                         //or a PTZ protocol file
    char szVender[32];         // [in]Vender Name
    char szProtocol[32];  // [in]Protocol Name
    char szProtocolFileName[512]; // [in]Specify the PTZ protocol file name
    DWORD dwAddressID;         // Address ID
} MEDIA_PTZ_PROTOCOL;
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZUnloadProtocol ,(Back To PTZ List)

KPTZMove

Description

PTZ (Pan Tilt Zoom) Move function.

Syntax

```
bool KPTZMove(HANDLE h, int nAddrID, int nSpeed, PTZ_MOVE_OPERATION PTZMoveOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] Which specify the address ID.
nSpeed	**int**	[in] Which specify the moving speed.
PTZMoveOP	**PTZ_MOVE_OPERATION**	[in] 8 directions and stop.

Returns

Return true when function success.

Remarks

```
enum PTZ_MOVE_OPERATION
{
    PTZ_MOVE_UP,
    PTZ_MOVE_DOWN,
    PTZ_MOVE_LEFT,
    PTZ_MOVE_RIGHT,
    PTZ_MOVE_UP_LEFT,
    PTZ_MOVE_UP_RIGHT,
    PTZ_MOVE_DOWN_LEFT,
    PTZ_MOVE_DOWN_RIGHT,
    PTZ_MOVE_STOP
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZZoom ,(Back To PTZ List)

KPTZOSD

Description

PTZ (Pan Tilt Zoom) OSD (On Screen Display) function.

Syntax

```
bool KPTZOSD(HANDLE h, int nAddrID,  PTZ_OSD_OPERATION PTZOSDOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] Specify the address ID.
PTZOSDOP	**PTZ_OSD_OPERATION**	[in] PTZ OSD operation.

Returns

Return true when function success.

Remarks

```
enum PTZ_OSD_OPERATION
{
    PTZ_OSD_ON,
    PTZ_OSD_OFF,
    PTZ_OSD_UP,
    PTZ_OSD_DOWN,
    PTZ_OSD_LEFT,
    PTZ_OSD_RIGHT,
    PTZ_OSD_ENTER,
    PTZ_OSD_LEAVE
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

(Back To PTZ List)

KPTZPreset

Description

PTZ (Pan Tilt Zoom) Preset function.

Syntax

```
bool KPTZPreset(HANDLE h, int nAddrID, int nPresetPos, PTZ_RESEST_OPERATION
PTZPresetOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] which specify the address ID
nPresetPos	**int**	[in] preset position
PTZPresetOP	**PTZ_RESEST_OPERATION**	[in] Set/Goto

Returns

Return true when function success.

Remarks

```
enum PTZ_RESEST_OPERATION
{
    PTZ_PRESET_SET,
    PTZ_PRESET_GOTO
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

(Back To PTZ List)

KPTZUnitToDegree

Description

Change the units of hardware to drgrees.

Syntax

```
void KPTZUnitToDegree( HANDLE h, int iPanUnit, int iTiltUnit, int iZoomUnit, float&
fPanDegree, float& fTiltDegree, float& fZoomRatio );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
iPanUnit	**int**	[in]Pan unit
iTiltUnit	**int**	[in]Tilt unit
iZoomUnit	**int**	[in]Zoom unit
fPanDegree	**float&**	[out]Pan degree
fTiltDegree	**float&**	[out]Tilt degree
fZoomRatio	**float&**	[out]Zoom degree

Returns

Return true when function success.

Remarks

Change the units of hardware to drgrees by Linear interpolation method.

The detail is defined in ptz file.

For example:

#DynaColor_DynaColor.ptz

PMAX; 1600

PMIN; 1

PMAXDEGREE; 360

TMAX is set at 90 degree

TMAXDEGREE; 90

TMAX; 223

TMIN is set at 0 degree

TMIN; 23

ZMAX; 37

ZMIN; 1

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZDegreeToUnit , (Back To PTZ List)

KPTZUnloadProtocol

Description

Unload PTZ (Pan Tilt Zoom) Protocol.

Syntax

```
bool KPTZUnloadProtocol(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZLoadProtocol ,(Back To PTZ List)

KPTZZoom

Description

PTZ (Pan Tilt Zoom) Zoom function.

Syntax

```
bool KPTZZoom(HANDLE h, int nAddrID, int nSpeed, PTZ_ZOOM_OPERATION PTZZoomOP);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nAddrID	**int**	[in] which specify the address ID
nSpeed	**int**	[in] which specify the moving speed
PTZZoomOP	**PTZ_ZOOM_OPERATION**	[in] PTZ Zoom In/Out/Stop

Returns

Return true when function success.

Remarks

```
enum PTZ_ZOOM_OPERATION
{
    PTZ_ZOOM_IN,
    PTZ_ZOOM_OUT,
    PTZ_ZOOM_STOP
};
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, PTZParser.dll**

Example

See Also

KPTZMove ,(Back To PTZ List)

KSendPTZCommand

Description

Send PTZ Command.

Syntax

```
void KSendPTZCommand (HANDLE h, BYTE* cmd, DWORD len);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
cmd	**BYTE***	[in] PTZ command.
len	**DWORD**	[in] PTZ command length

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll , PTZParser.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
```

```
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KSendPTZCommand(h, pPTZCmd, dwPTZCmdLen);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To PTZ List)

Motion Detection

Name	*Description*
KGetMotionInfo	Get the Server's Motion Detect Range and Sensitive Setting Value.
KSetEvent_MotionDetection	Set event structural for motion detection.
KSetMotionDetectionCallback	Set the callback to get the motion detect event
KSetMotionInfo	Set the Motion Detect Range

KGetMotionInfo

Description

Get the Server's Motion setting value.

Syntax

```
void KGetMotionInfo(HANDLE h, MEDIA_MOTION_INFO* MotionInfo)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
pMdSetting	**MEDIA_MOTION_INFO***	[out] the Motion information on the video server.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
```

```
                MEDIA_MOTION_INFO mmi;
                memset(&mmi, 0x00, sizeof(MEDIA_MOTION_INFO));
                KGetMotionInfo(h, &mmi);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KSetMotionInfo, (Back To Motion Detection List)

KSetEvent_MotionDetection

Description

Set event structural for motion detection

Syntax

```
void KSetEvent_MotionDetection(HANDLE h, NOTIFY_MOTION_DETECTION*
nMotionDetection);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
nMotionDetection	**NOTIFY_MOTION_DETECTION***	[in] Event structural for motion detection.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, ARAW.dll**

Example

See Also

(Back To Motion Detection List)

KSetMotionDetectionCallback

Description

Set the callback to get the motion detect event

Syntax

```
void KSetMotionDetectionCallback (HANDLE h, DWORD UserParam,
MOTION_DETECTION_CALLBACK fnMotionDetectionCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] Custom param for carry to callback function.
fnMotionDetectionCallback	**MOTION_DETECTION_CALLBACK**	[in] the pointer to the callback function

Returns

No return value.

Remarks

Below is the definition of MOTION_DETECTION_CALLBACK.

typedef void (CALLBACK *MOTION_DETECTION_CALLBACK)(DWORD UserParam, bool Motion1, bool Motion2, bool Motion3);

Motion1, Motion2 and Motion3 will trigger when there is a motion.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK MotionDetectionCB(DWORD UserParam, bool bMotion1,
bool bMotion2, bool bMotion3)
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetMotionDetectionCallback(h, (DWORD)this, MotionDetectionCB);
    . . . . .
}
```

See Also

(Back To Motion Detection List)

KSetMotionInfo

Description

Set the Motion Detect Range

Syntax

```
void KSetMotionInfo (HANDLE h, MEDIA_MOTION_INFO* MotionInfo);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
MotionInfo	**MEDIA_MOTION_INFO***	[in]The Motion Detect Range Setting

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
```

```
        }
    }
}
. . . . .
MEDIA_MOTION_INFO mmi;
mmi.dwRangeCount = 3;
mmi.dwSensitive[0] = Sensitive_for_1;
mmi.dwRange[0][0] = X_Pos1;
mmi.dwRange[0][1] = Y_Pos1;
mmi.dwRange[0][2] = Width1;
mmi.dwRange[0][3] = Height1;
mmi.dwSensitive[1] = Sensitive_for_2;
mmi.dwRange[1][0] = X_Pos1;
mmi.dwRange[1][1] = Y_Pos1;
mmi.dwRange[1][2] = Width1;
mmi.dwRange[1][3] = Height1;
mmi.dwSensitive[2] = Sensitive_for_3;
mmi.dwRange[2][0] = X_Pos1;
mmi.dwRange[2][1] = Y_Pos1;
mmi.dwRange[2][2] = Width1;
mmi.dwRange[2][3] = Height1;
mmi.dwEnable = bEnable;
KSetMotionInfo(h, &mmi);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetMotionInfo, (Back To Motion Detection List)

Digital I/O

Name	*Description*
KGetDIDefaultValueByHTTP	Get DI default using HTTP.
KGetDIOStatusByHTTP	Get DIO status using HTTP.
KSendDO	Send DO to video server.
KSetDICallback	Set the callback to get the DI Status.
KSetDICallbackEx	Set the callback to get the DI Status.
KSetDIDefaultValue	Set DI default.
KSetEvent_DI	Set DI event structural.

KGetDIDefaultValueByHTTP

Description

Get DI default value using HTTP.

Syntax

```
BYTE KGetDIDefaultValueByHTTP (HANDLE h, char* IP, unsigned long HTTPPort
char* UID, char*PWD);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port number.
UID	**char***	[in] User account for login.
PWD	**char***	[in] Password for login.

Returns

DI default value.

Remarks

DI value	**Description**
DI_DEFAULT_IS_LOW (0x00)	Default setting is low.
DI_DEFAULT_IS_HIGH (0X03)	Default setting is high.
0xFF	Error.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
```

```
if(NULL != h)
{
    BYTE bDefaultValue = KGetDIDefaultValueByHTTP(h, IP, HTTPPort, UID, PWD);
}
```

See Also

KGetDIOStatusByHTTP, (Back To Digital I/O List)

KGetDIOStatusByHTTP

Description

Get DIO status using HTTP.

Syntax

```
BYTE KGetDIOStatusByHTTP (HANDLE h, char* IP, unsigned long HTTPPort
char* UID, char*PWD);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port number.
UID	**char***	[in] User account for login.
PWD	**char***	[in] Password for login.

Returns

DIO Status value.

Remarks

DIO value	**Description**
BIT 0	DI1 Status
BIT 1	DI2 Status
BIT 2	Reserved
BIT 3	Reserved
BIT 4	DO1 Status
BIT 5	DO2 Status
BIT 6	Reserved
BIT 7	Reserved

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    BYTE bDIO = KGetDIOStatusByHTTP(h, IP, HTTPPort, UID, PWD);
}
```

See Also

KGetDIDefaultValueByHTTP, (Back To Digital I/O List)

KGetDIOStatusByHTTPEx

Description

Get DIO status from multi-channel using HTTP.

Syntax

```
BYTE KGetDIOStatusByHTTPEx (HANDLE h, char* IP, unsigned long HTTPPort, unsigned long nChannel, char* UID, char*PWD);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
IP	**char***	[in] Video server IP address.
HTTPPort	**unsigned long**	[in] HTTP port number.
nChannel	**unsigned long**	[in] Channel number.
UID	**char***	[in] User account for login.
PWD	**char***	[in] Password for login.

Returns

DIO Status value.

Remarks

DIO value	**Description**
BIT 0	DI1 Status
BIT 1	DI2 Status
BIT 2	Reserved
BIT 3	Reserved
BIT 4	DO1 Status
BIT 5	DO2 Status
BIT 6	Reserved
BIT 7	Reserved

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    BYTE bDIO = KGetDIOStatusByHTTPEx(h, IP, HTTPPort, nChannel, UID, PWD);
}
```

See Also

KGetDIDefaultValueByHTTP, (Back To Digital I/O List)

KSendDO

Description

Send DO to video server.

Syntax

```
void KSendDO (HANDLE h, BYTE bDOData);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bDOData	**BYTE**	[in] the DO Event

Returns

No return value.

Remarks

DO value	Description
DO_OUTPUT_CLEAN (0X00)	Clean DO.
DO_OUTPUT_1 (0X01)	DO 1
DO_OUTPUT_2 (0X02)	DO 2
DO_OUTPUT_BOTH (0X03)	DO 1 & 2

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
```

```
    if(KSetMediaConfig(h, &mcc))
    {
         if(KConnect(h))
        {
             if(KStartStream(h))
             {
                KSendDO(h, DO_OUTPUT_1);
             }
        }
     }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To Digital I/O List)

KSetDICallback

Description

Set DI callback.

Syntax

```
void KSetDICallback(HANDLE h, DWORD UserParam, DI_CALLBACK fnDICallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] Custom param for carry to callback function
fnDICallback	**DI_CALLBACK**	[in] pointer for callback function.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK DICB(DWORD UserParam, bool bDI1, bool bDI2)
{
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetDICallback(h, (DWORD)this, DICB);
    . . . . .
```

```
}
```

See Also

(Back To Digital I/O List)

KSetDICallbackEx

Description

Set DI callback. Triggered when DI On or Off at first time.

Syntax

```
void KSetDICallbackEx(HANDLE h, DWORD UserParam, DI_CALLBACK_EX fnDIExCallback);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] Custom param for carry to callback function
fnDIExCallback	**DI_CALLBACK_EX**	[in] pointer for callback function.

Returns

No return value.

Remarks

32 inputs total. Status 1 means On, 0 means Off, and -1 means nothing change.

```
/*
*typedef struct structural_DI_EX_CALLBACK_DATA
*{
*      int DIStatus[32];
*}DI_EX_CALLBACK_DATA;
*/
typedef void ( CALLBACK *DI_CALLBACK_EX    )
(       DWORD UserParam,
       DI_EX_CALLBACK_DATA di );
```

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

See Also

(Back To Digital I/O List)

KSetDIDefaultValue

Description

Set DI default value.

Syntax

```
void KSetDIDefaultValue(HANDLE h, BYTE bDefault);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bDefault	**BYTE**	[in] DI default value.

Returns

No return value.

Remarks

DI value	Description
DI_DEFAULT_IS_LOW (0x00)	Set DI default to low.
DI_DEFAULT_IS_HIGH (0x03)	Set DI default to high.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    KSetDIDefaultValue(h, DI_DEFAULT_IS_HIGH);
}
```

See Also

(Back To Digital I/O List)

KSetEvent_DI

Description

Set DI event.

Syntax

```
void KSetEvent_DI(HANDLE h, NOTIFY_DI* nDI);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nDI	**NOTIFY_DI***	[in] pointer for notify DI structural..

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

See Also

(Back To Digital I/O List)

QUAD

Name	*Description*
KQuadGetBrightness	Get Quad brightness setting.
KQuadGetContrast	Get Quad contrast setting.
KQuadGetDisplayMode	Get current Quad display mode.
KQuadGetHue	Get Quad hue setting.
KQuadGetMotionDetectionEnbale	Get Quad motion status.
KQuadGetMotionSensitive	Get Quad sensitive setting.
KQuadGetOSDEnable	Get Quad OSD status.
KQuadGetSaturation	Get Quad saturation setting.
KQuadGetTitleName	Get Quad channel title name.
KQuadSetBrightness	Set Quad brightness.
KQuadSetContrast	Set Quad contrast.
KQuadSetDisplayMode	Set Quad display mode.
KQuadSetHue	Set Quad hue.
KQuadSetMotionDetectionEnable	Set Quad motion.
KQuadSetMotionSensitive	Set Quad sensitive.
KQuadSetOSDEnable	Set Quad OSD.
KQuadSetSaturation	Set Quad saturation.
KQuadSetTitleName	Set Quad channel title name.
KSetQuadMotionDetectionCallback	Set motion callback for Quad
KSetQuadSetVideoLossCallback	Set video loss callback for Quad.
KSetQuadVideoLossCallback	Set callback function for Quad video loss.
KSetTargetCameraIsQuad	Set Target camera type to Quad.

KQuadGetBrightness

Description

Get Quad channel brightness value.

Syntax

```
int KQuadGetBrightness(HANDLE h, int nChannel)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.

Returns

If return greater than 0 then channel brightness returned.

Return -1 if function fails..

Remarks

Channel number from 1 to 4.

Channel brightness value from 0 to 255.

Brightness	**Description**
0	-25 IRE
……	……
128	0 IRE
……	……
255	25 IRE

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  int nBrightness = KQuadGetBrightness(h, nChannel);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetBrightness, (Back To QUAD List)

KQuadGetContrast

Description

Get Quad channel contrast value.

Syntax

```
int KQuadGetContrast(HANDLE h, int nChannel)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.

Returns

If return greater than 0 then channel contrast returned.

Return -1 if function fails..

Remarks

Channel number from 1 to 4.

Channel contrast value from 0 to 255.

Contrast	Description
0	0%
……	……
128	100%
……	……
255	200%

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  int nContrast = KQuadGetContrast(h, nChannel);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetContrast, (Back To QUAD List)

KQuadGetDisplayMode

Description

Get Quad's display mode.

Syntax

```
int KQuadGetDisplayMode(HANDLE h)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

Quad display Mode.

0 – Quad display.

1 – Display channel one.

2 – Display channel two.

3 – Display channel three.

4 – Display channel four.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
```

```
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  int nDisplayMode = KQuadGetDisplayMode(h);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetDisplayMode, (Back To QUAD List)

KQuadGetHue

Description

Get Quad channel hue value.

Syntax

```
int KQuadGetHue(HANDLE h, int nChannel)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.

Returns

If return greater than 0 then channel hue returned.

Return -1 if function fails..

Remarks

Channel number from 1 to 4.

Channel hue value from 0 to 255.

Saturation	**Description**
0	-180 Degree
……	……
128	0 Degree
……	……
255	180 Degree

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  int nHue = KQuadGetHue(h, nChannel);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetHue, (Back To QUAD List)

KQuadGetMotionDetectionEnable

Description

Get Quad motion detection status.

Syntax

```
BYTE KQuadGetMotionDetectionEnable(HANDLE h)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

Motion Detection return BYTE

Bit 0: 1 – Channel 1 motion detect enabled.

Bit 1: 1 – Channel 2 motion detect enabled.

Bit 2: 1 – Channel 3 motion detect enabled.

Bit 3: 1 – Channel 4 motion detect enabled.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
```

```
            if(KStartStream(h))
            {
               BYTE btMotion = KQuadGetMotionDetectionEnable(h);
            }
        }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetMotionDetectionEnable, (Back To QUAD List)

KQuadGetMotionSensitive

Description

Get Quad motion sensitive status.

Syntax

```
int KQuadGetMotionSensitive(HANDLE h, int nChannel)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.

Returns

If function succeeds then Quad sensitive status returned otherwise -1.

Remarks

Channel number from 1 to 4.

Quad sensitive status.

0: less sensitive.

…..

] 50: middle sensitive.

…..

100: more sensitive.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
```

```
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  int nSensitive = KQuadGetMotionSensitive(h, nChannel);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetMotionSensitive, (Back To QUAD List)

KQuadGetOSDEnable

Description

Get Quad OSD status.

Syntax

```
BYTE KQuadGetOSDEnable(HANDLE h)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

OSD status

Bit 0: 1 – Title name enable.

Bit 1: 1 – Video loss enable.

Bit 2: 1 – Motion detect enable.

Bit 3: 1 – Date time enable.

Bit 4: 1 – DIO status enable.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
```

```
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  BYTE btOSD = KQuadGetOSDEnable(h);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetOSDEnable, (Back To QUAD List)

KQuadGetSaturation

Description

Get Quad channel saturation value.

Syntax

```
int KQuadGetSaturation(HANDLE h, int nChannel)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.

Returns

If return greater than 0 then channel saturation returned.

Return -1 if function fails..

Remarks

Channel number from 1 to 4.

Channel saturation value from 0 to 255.

Saturation	**Description**
0	0%
……	……
128	100%
……	……
255	200%

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  int nSaturation = KQuadGetSaturation(h, nChannel);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetSaturation, (Back To QUAD List)

KQuadGetTitleName

Description

Get Quad channel title name.

Syntax

```
int KQuadGetTitleName(HANDLE h, int nChannel, char* pName8Bytes)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.
pName8Bytes	**char***	[out] Quad channel title name.

Returns

If function succeeds length of camera title will return otherwise -1.

Remarks

Channel number from 1 to 4.

Max length of title is 8 bytes.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
```

```
        {
             if(KStartStream(h))
             {
                char szTitleName[16] = {0};
                int nLen = KQuadGetTitleName(h, nChannel, szTitleName);
             }
        }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadSetTitleName, (Back To QUAD List)

KQuadSetBrightness

Description

Set Quad channel brightness value.

Syntax

```
bool KQuadSetBrightness(HANDLE h, int nChannel, int nBrightness)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.
nBrightness	**int**	[in] Brightness value.

Returns

If function return succeeds, then new brightness value has set to the channel.

If function return fails, then channel brightness reamin the same.

Remarks

Channel number from 1 to 4.

Channel brightness value from 0 to 255.

Brightness	**Description**
0	-25 IRE
……	……
128	0 IRE
……	……
255	25 IRE

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  bool b = KQuadSetBrightness(h, nChannel, nBrightness);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetBrightness, (Back To QUAD List)

KQuadSetContrast

Description

Set Quad channel contrast value.

Syntax

```
bool KQuadSetContrast(HANDLE h, int nChannel, int nContrast)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.
nContrast	**int**	[in] Contrast value.

Returns

If function return succeeds, then new contrast value has set to the channel.

If function return fails, then channel contrast reamin the same.

Remarks

Channel number from 1 to 4.

Channel contrast value from 0 to 255.

Contrast	**Description**
0	0%
……	……
128	100%
……	……
255	200%

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  bool b = KQuadSetContrast(h, nChannel, nContrast);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetContrast, (Back To QUAD List)

KQuadSetDisplayMode

Description

Set Quad display mode.

Syntax

```
bool KQuadSetDisplayMode(HANDLE h, int nMode)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nMode	**int**	[in] Display mode.

Returns

If function return succeeds, then new display has set to the Quad video server.

If function return fails, then display remain the same.

Remarks

Value for Display mode.

0 – Quad display.

1 – Display channel one.

2 – Display channel two.

3 – Display channel three.

4 – Display channel four.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
```

```
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KQuadSetDisplayMode(h, nMode);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetDisplayMode, (Back To QUAD List)

KQuadSetHue

Description

Set Quad channel hue value.

Syntax

```
bool KQuadSetHue(HANDLE h, int nChannel, int nHue)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.
nHue	**int**	[in] Hue value.

Returns

If function return succeeds, then new hue value has set to the channel.

If function return fails, then channel hue reamin the same.

Remarks

Channel number from 1 to 4.

Channel hue value from 0 to 255.

Hue	**Description**
0	-180 Degree
……	……
128	0 Degree
……	……
255	180 Degree

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  bool b = KQuadSetHue(h, nChannel, nHue);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetHue, (Back To QUAD List)

KQuadSetMotionDetectionEnable

Description

Set Quad motion detection enable.

Syntax

```
bool KQuadSetMotionDetectionEnable(HANDLE h, BYTE btEnable)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
btEnable	**BYTE**	[in] Channel to enable.

Returns

If function return succeeds, then new motion detect setting has set to the Quad video server.

If function return fails, then motion detect setting reamin the same.

Remarks

Motion Detection for btEnable.

Bit 0: 1 – Channel 1 motion detect enabled.

Bit 1: 1 – Channel 2 motion detect enabled.

Bit 2: 1 – Channel 3 motion detect enabled.

Bit 3: 1 – Channel 4 motion detect enabled.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
```

```
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KQuadSetMotionDetectionEnable(h, btMotion);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetMotionDetectionEnable, (Back To QUAD List)

KQuadSetMotionSensitive

Description

Set Quad motion sensitive.

Syntax

```
bool KQuadSetMotionSensitive(HANDLE h, int nChannel, int nSensitive)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.
nSensitive	**int**	[in] Sensitive value.

Returns

If function return succeeds, then new sensitive setting has set to the channel.

If function return fails, then sensitive setting reamin the same.

Remarks

Channel number from 1 to 4.

Quad sensitive status.

0: less sensitive.

…..

] 50: middle sensitive.

…..

100: more sensitive.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KQuadSetMotionSensitive(h, nChannel, nSensitive);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetMotionSensitive, (Back To QUAD List)

KQuadSetOSDEnable

Description

Set Quad OSD.

Syntax

```
bool KQuadSetOSDEnable(HANDLE h, BYTE btEnable)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
btEnable	**BYTE**	[in] OSD enable option.

Returns

If function return succeeds, then new OSD has set to the Quad video server.

If function return fails, then OSD setting remain the same.

Remarks

OSD enable BYTE

Bit 0: 1 – Title name enable.

Bit 1: 1 – Video loss enable.

Bit 2: 1 – Motion detect enable.

Bit 3: 1 – Date time enable.

Bit 4: 1 – DIO status enable.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
```

```
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KQuadSetOSDEnable(h, btOSD);
               }
          }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetOSDEnable, (Back To QUAD List)

KQuadSetSaturation

Description

Set Quad channel saturation value.

Syntax

```
bool KQuadSetSaturation(HANDLE h, int nChannel, int nSaturation)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.
nSaturation	**int**	[in] Saturation value.

Returns

If function return succeeds, then new saturation value has set to the channel.

If function return fails, then channel saturation reamin the same.

Remarks

Channel number from 1 to 4.

Channel saturation value from 0 to 255.

Saturation	**Description**
0	0%
……	……
128	100%
……	……
255	200%

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  bool b = KQuadSetSaturation(h, nChannel, nSaturation);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetSaturation, (Back To QUAD List)

KQuadSetTitleName

Description

Set Quad channel title name.

Syntax

```
bool KQuadSetTitleName(HANDLE h, int nChannel, char* pName8Bytes)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nChannel	**int**	[in] Channel number.
pName8Bytes	**char***	[in] Quad channel title name.

Returns

If function return succeeds, then new title name has set to the channel.

If function return fails, then channel title reamin the same.

Remarks

Channel number from 1 to 4.

Max length of title is 8 bytes.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
```

```
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  bool b = KQuadSetTitleName(h, nChannel, szTitleName);
               }
          }
     }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KQuadGetTitleName, (Back To QUAD List)

KSetQuadMotionDetectionCallback

Description

Set motion detection callback for Quad.

Syntax

```
void KSetQuadMotionDetectionCallback(HANDLE h, DWORD UserParam,
QUAD_MOTION_DETECTION_CALLBACK fnQuadMotionDetectionCallback)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] Custom param for carry to callback function
fnQuadMotionDetectionCallback	**QUAD_MOTION_DETECTION_CALLBACK**	[in] function pointer for callback

Returns

No return values.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK QuadMotionDetectionCB(DWORD UserParam, bool bMotion1,
bool bMotion2, bool bMotion3, bool bMotion4)
{
```

```
    . . . . .
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetQuadMotionDetectionCallback(h, (DWORD)this, QuadMotionDetectionCB);
    . . . . .
}
```

See Also

(Back To QUAD List)

KSetQuadSetVideoLossCallback

Description

Set video loss callback for Quad.

Syntax

```
void KSetQuadSetVideoLossCallback(HANDLE h, DWORD UserParam,
QUAD_VIDEO_LOSS_CALLBACK fnQuadVideoLossCallback)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
UserParam	**DWORD**	[in] Custom param for carry to callback function
fnQuadVideoLossCallback	**QUAD_VIDEO_LOSS_CALLBACK**	[in] function pointer for callback

Returns

No return values.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
void CALLBACK QuadVideoLossCB(DWORD UserParam, bool bR1VideoLoss,
bool bR2VideoLoss, bool bR3VideoLoss, bool bR4VideoLoss)
{
    . . . . .
}

. . . . .
```

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetQuadSetVideoLossCallback(h, (DWORD)this, QuadMotionDetectionCB);
    . . . . .
}
```

See Also

(Back To QUAD List)

KSetQuadVideoLossCallback

Description

Set callback function for Quad video loss.

Syntax

```
void KSetQuadVideoLossCallback( HANDLE h, DWORD UserParam,
QUAD_VIDEO_LOSS_CALLBACK fnQuadVideolossCallback );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
UserParam	**DWORD**	[in] Custom param for carry to callback function
fnQuadVideolossCallback	**QUAD_VIDEO_LOSS_CALLBACK**	[in] function pointer for callback

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To QUAD List)

KSetTargetCameraIsQuad

Description

Set connect target camera to Quad type.

Syntax

```
void KSetTargetCameraIsQuad(HANDLE h, bool bIsQuad)
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bIsQuad	**bool**	[in] True – Camera is Quad. False – Camera is not Quad.

Returns

No return values.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
if(NULL != h)
{
    KSetTargetCameraIsQuad(h, true);
    . . . . .
}
```

See Also

(Back To QUAD List)

User Interface

Name	*Description*
KEnablePrivacyMask	Set Privacy Zone on image of video.
KEnableRender	Enable/Disable render.
KFlipImage	Inverse Image of video (Upside Down)
KMirrorImage	Inverse image of video (Left to Right)
KNotifyFullScreenWindow	Send notify to full screen window.
KSetDrawerType	Set the method to display video frames.
KSetRenderInfo	Set SDK render information.
KSetTextOut	Display text on video frame.

KEnablePrivacyMask

Description

Set Privacy Zone on image of video. (3 rects maximum)

Syntax

```
void KEnablePrivacyMask( HANDLE h, bool bEnable, RECT r1, RECT r2, RECT r3, BYTE btColor_R, BYTE btColor_G, BYTE btColor_B );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnable	**bool**	[in] Enable or disable
r1	**RECT**	[in] Privacy zone 1
r2	**RECT**	[in] Privacy zone 2
r3	**RECT**	[in] Privacy zone 3
btColor_R	**BYTE**	[in] Blocking color R
btColor_G	**BYTE**	[in] Blocking color G
btColor_B	**BYTE**	[in] Blocking color B

Returns

Windows message return code.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

See Also

(**Back To User Interface List**)

KEnableRender

Description

Enable/Disable Render.

Syntax

```
void KEnableRender (HANDLE h, bool bEnableRender);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnableRender	**bool**	[in] Flag to Enable/Disable.

Returns

No return values.

Remarks

If bEnableRender assign to true then SDK will draw video frames base on KSetRenderInfo.

If bEnableRender assign to false then SDK will not draw video frames.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
```

```
                KPlay(h);
            }
        }
    }
}
. . . . .
KEnableRender(h, false);
. . . . .
KEnableRender(h, true);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To User Interface List)

KFlipImage

Description

Inverse Image of video (Upside Down)

Syntax

```
void KFlipImage( HANDLE h, bool bFlip )
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bFlip	**bool**	[in] Enable or disable

Returns

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

See Also

(Back To User Interface List)

KMirrorImage

Description

Inverse image of video (Left to Right)

Syntax

```
void KMirrorImage( HANDLE h, bool bMirror )
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bMirror	**bool**	[in] Enable or disable

Returns

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

See Also

(Back To User Interface List)

KNotifyFullScreenWindow

Description

Send notify to full screen window.

Syntax

```
DWORD KNotifyFullScreenWindow (HANDLE h,UINT message, WPARAM wParam,
LPARAM lParam);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
message	**UNIT**	[in] windows message
wParam	**WPARAM**	[in] message
lParam	**LPARAM**	[in] message

Returns

Windows message return code.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

See Also

(Back To User Interface List)

KSetDrawerType

Description

Set the method to display video frames.

Syntax

```
void KSetDrawerType(HANDLE h, int nType);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nType	**DWORD***	[in]Drawer type

Returns

No return values.

Remarks

Drawer Type	Description
DGDI (0)	Request to use windows GDI for draw.
DXDRAW (1)	Request to use Direct Draw for draw

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, DGDI.dll & DXDRAW.dll**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    KSetDrawerType(h, DGDI);
```

```
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
     }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To User Interface List)

KSetRenderInfo

Description

Set SDK render information.

Syntax

```
void KSetRenderInfo (HANDLE h, MEDIA_RENDER_INFO* RenderInfo);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
RenderInfo	**MEDIA_RENDER_INFO***	[in] Render information.

Returns

No return values.

Remarks

MEDIARENDER_INFO	**Description**
DrawerInterface	Drawer type. DGDI or DXDRAW.
hWnd	Windows handle use to draw.
rect	Area to draw.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll, DGDI.dll & DXDRAW.dll**

Example

```
HANDLE h = KOpenInterface();
. . . . .
MEDIA_RENDER_INFO mri;
mri.rect.top = ntop;
mri.rect.right = nright;
mri.rect.left = nleft;
```

```
mri.rect.bottom = nbottm;
mri.hWnd = hWnd;
if(h)
{
     KSetRenderInfo(h, &mri);
}
```

See Also

(Back To User Interface List)

KSetTextOut

Description

Display text on video frame.

Syntax

```
void KSetTextOut(HANDLE h, int nIndex, int nX, int nY, char* Text, int nTextLen,
bool bBold, bool bItalic, bool bUnderLine, const char* pFontName, int nFontSize,
COLORREF color, int nBKMode, COLORREF BKcolor);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nIndex	**int**	[in] Index number of display text.
nX	**int**	[in] X pos of text.
nY	**int**	[in] Y pos of text.
Text	**char***	[in] Text going to display
nTextLen	**int**	[in] Text length.
bBold	**bool**	[in] True – Bold, False – Normal.
bItalic	**bool**	[in] True – Italic, False – Normal.
bUnderLine	**bool**	[in] True – Underline, False – Normal.
pFontName	**const char***	[in] Text font style.
nFontSize	**int**	[in] Text size.
color	**COLORREF**	[in] Text color.
nBKMode	**int**	[in] Background mode. 1 – TRANSPRANT. 2 – OPAQUE.
nBKcolor	**COLORREF**	[in] Background color.

Returns

No return value.

Remarks

Index value is from 0 to 9.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
            }
        }
    }
}
. . . . .
KSetTextOut(h, 0, 0, 0, “123456789\0”, 9, true, false, false, “Arial”, 100,
RGB(255, 255, 0), 2, RGB(0, 0, 255));
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To User Interface List)

Utility

Name	*Description*
KGetVersion	Get the SDK’s Version

KGetVersion

Description

Get the SDK's Version.

Syntax

```
void KGetVersion(char* Version);
```

Parameters

Name	*Type*	*Description*
message	**UNIT**	[in] windows message

Returns

SDK version.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate adaptors**

Example

```
char szSDKVersion[32] = {0};
KGetVersion(szSDKVersion);
```

See Also

(Back To Utility List)

-

Miscellaneous

Name	*Description*
KDecodeFrame	Decode a frame.
KDigitalPTZEnable	Enable the Digital PTZ Function.
KDigitalPTZTo	Digital PTZ To a image region.
KEnableJitterLessMode	Enable the Jitter Less Mode.
KGetCameraName	Get camera name.
KGetFrameRateMode	Get frame rate mode of video server.
KGetLastError	Get the error reason
KGetTotalReceiveAudioFrameCount	Get the total number of audio frame received.
KGetTotalReceiveSize	Get the size of video data received
KGetTotalReceiveVideoFrameCount	Get the total number of video frame received
KGetVideoConfig	Get video server's config
KReverseImageLeftToRight	Inverse the image left To right.
KReverseImageUpToDown	Inverse the image upside down.
KSaveReboot	Save and Reboot video server.
KSendAudioToSE	Send audio data (PCM) to Stream engine.
KSendCommand	Send a media command command to SDK kernel.
KSendCommandToSE	Send command to Stream engine, and get result of execution.
KSendCommandToStreamingEngine	Send command to Stream engine.
KSetAutoDropFrameByCPUPerformance	Set auto frame rate by CPU threshold
KSetBitRate	Set video server's bitrate
KSetBrightness	Set video server's brightness
KSetContrast	Set video server's contrast.
KSetCurrentPosition	Set current position in processing file. (sec)
KSetFPS	Set video server's FPS (Constant Frame Rate Mode)
KSetHue	Set video server's hue
KSetResolution	Set video server's resolution.
KSetSaturation	Set video server's saturation.
KSetVariableFPS	Set video server's FPS (Variable Frame Rate Mode)
KSetVideoConfig	Set video server's config.

KStartDecodeMode	Start the SDK with a decoder mode.
KStopDecodeMode	Stop the SDK decoder mode.

KDecodeFrame

Description

Decode a frame.

Syntax

```
bool KDecodeFrame(HANDLE h, BYTE* pData, int nLen, int nRawDataType );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
pData	**BYTE***	[in] Mpeg4/MJPEG/H.264 data.
nLen	**Int**	[in] The length of pData.
nRawDataType	**int**	[in] 1 for mpeg4, 2,3 for audio(PCM 8K/16Bit) 4 for MJPEG, 5 for H.264

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

KStartDecodeMode , KStopDecodeMode, (Back To Miscellaneous List)

KDigitalPTZEnable

Description

Enable the Digital PTZ Function.

Syntax

```
void KDigitalPTZEnable( HANDLE h, bool bEnableEPTFunction );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnableEPTFunction	**bool**	[in] true for enable, false for disable

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KDigitalPTZTo

Description

Digital PTZ To a image region.

Syntax

```
void KDigitalPTZTo( HANDLE h, int nXSrc, int nYSrc, int nWidth, int nHeight );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
nXSrc	**int**	[in] image left
nYSrc	**int**	[in] image top
nWidth	**int**	[in] image width
nHeight	**int**	[in]image height

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KEnableJitterLessMode

Description

Enable the Jitter Less Mode.

Syntax

```
void KEnableJitterLessMode( HANDLE h, bool bEnable );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnable	**bool**	[in] To enable or not.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KGetCameraName

Description

Get Name of camera.

Syntax

```
bool KGetCameraName( HANDLE h, char* pCameraNameBuffer, int nBufferSize );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
pCameraNameBuffer	**char***	[in/out] The string buffer
nBufferSize	**int**	[in] The size of string buffer

Returns

True if success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

See Also

(Back To Miscellaneous List)

KGetFrameRateMode

Description

Get frame rate mode of video server.

Syntax

```
int KGetFrameRateMode(HANDLE h, char* IP, unsigned long HTTPPort, char* UID, char* PWD, unsigned int ChannelNO);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
IP	**char***	[in] Video server IP address.
HTTPPort	**int**	[in] HTTP port number
UID	**char***	[in] User login name.
PWD	**char***	[in] User login password.
ChannelNO	**char***	[in] Channel number.

Returns

Result	**Description**
0	Error - Unable to get frame rate mode.
1	Success – Frame rate mode is Constant.
2	Success – Frame rate mode is Variable.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
```

```
if(NULL != h)
{
    int nMode = KGetFrameRateMode(h, IP, HTTPPort, UID, PWD, ChannelNo);
}
```

See Also

(Back To Miscellaneous List)

KGetLastError

Description

Get the Error Reason

Syntax

```
DWORD KGetLastError(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

Error code. Please refer to **Error Code**.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
            }
        }
    }
```

```
        else
        {
            DWORD dwError = KGetLastError(h);
        }
}
```

See Also

([Back To Miscellaneous List])

KGetTotalReceiveAudioFrameCount

Description

Get the total number of audio frame received

Syntax

```
DWORD KGetTotalReceiveAudioFrameCount(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

Number of audio frame received.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
            }
        }
    }
```

```
}
. . . . .
DWORD dwAudioCount = KGetTotalReceiveAudioFrameCount(h);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetTotalReceiveSize, KGetTotalReceiveVideoFrameCount,

(Back To Miscellaneous List)

KGetTotalReceiveSize

Description

Get the total size of video data received

Syntax

```
DWORD KGetTotalReceiveSize(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

Total size of data received in bytes.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
            }
        }
    }
```

```
}
. . . . .
DWORD dwSize = KGetTotalReceiveSize(h);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetTotalReceiveAudioFrameCount, KGetTotalReceiveVideoFrameCount,

(Back To Miscellaneous List)

KGetTotalReceiveVideoFrameCount

Description

Get the total number of video frame received

Syntax

```
DWORD KGetTotalReceiveVideoFrameCount(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

Number of video frame received.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
            }
        }
    }
```

```
}
. . . . .
DWORD dwSize = KGetTotalReceiveVideoFrameCount(h);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetTotalReceiveAudioFrameCount, KGetTotalReceiveSize,

(Back To Miscellaneous List)

KGetVideoConfig

Description

Get video server's config

Syntax

```
bool KGetVideoConfig(HANDLE h, MEDIA_VIDEO_CONFIG* VideoConfig);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
VideoConfig	**MEDIA_VIDEO_CONFIG***	[out] the pointer to the struct MEDIA_VIDEO_CONFIG that contain the Video Server Config.

Returns

If the function succeeds, then video server information is container in the structure...

If the function fails, then get video server information fail..

Remarks

Structure MEDIA_VIDEO_CONFIG should initialize by user before use.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
```

```
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               MEDIA_VIDEO_CONFIG mvc;
               memset(&mvc, 0x00, sizeof(MEDIA_VIDEO_CONFIG));
               KGetVideoConfig(h, &mvc);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KSetVideoConfig, (Back To Miscellaneous List)

KReverseImageLeftToRight

Description

Inverse the image left To right.

Syntax

```
void KReverseImageLeftToRight( HANDLE h, bool bEnableLeftToRight );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnableLeftToRight	**bool**	[in] To inverse or not.

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

KReverseImageUpToDown, (Back To Miscellaneous List)

KReverseImageUpToDown

Description

Inverse the image upside down.

Syntax

```
void KReverseImageUpToDown( HANDLE h, bool bEnableUpToDown );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnableUpToDown	**bool**	[in] To inverse or not

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

KReverseImageLeftToRight, (Back To Miscellaneous List)

KSaveReboot

Description

Save Reboot the video server.

Syntax

```
void KSaveReboot(HANDLE h);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().

Returns

No return value.

Remarks.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
            }
        }
    }
```

```
}
. . . . .
KSaveReboot(h);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

(Back To Miscellaneous List)

KSendAudioToSE

Description

Send audio data (PCM) to Stream engine.

Syntax

```
bool KSendAudioToSE( HANDLE h, BYTE* pAudioBuffer, int nLen );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
pAudioBuffer	**BYTE***	[in] Audio data buffer.
nLen	**int**	[in] The size of pAudioBuffer

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KSendCommand

Description

Send a media command command to SDK kernel.

Syntax

```
void KSendCommand( HANDLE h, MEDIA_COMMAND* mc );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
mc	**MEDIA_COMMAND***	[in] Media command structure.

Returns

Return a string in char* pResult;

Remarks

typedef struct structural_MEDIA_COMMAND

{

DWORD dwCommandType;

char* pCommand;

int nCommandLength;

char* pResult;

int nResultLength;

} MEDIA_COMMAND;

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KSendCommandToSE

Description

Send command to Stream engine, and get result of execution.

Syntax

```
bool KSendCommandToSE( HANDLE h, char* URLCommand, DWORD dwLen, char*
ResultBuffer, DWORD& ResultBufferLen );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
URLCommand	**char***	[in]URL command buffer.
dwLen	**DWORD**	[in]The size of URL command buffer.
ResultBuffer	**char***	[in/out] in: NULL string buffer. out: The result of execution.
ResultBufferLen	**DWORD&**	[in/out] in: The size of NULL string buffer. out: The used size of ResultBuffer.

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KSendCommandToStreamingEngine

Description

Send command to Stream engine.

Syntax

```
bool KSendCommandToStreamingEngine( HANDLE h, char* URLCommand );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
URLCommand	**char***	[in] URL command buffer.

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KSetAutoDropFrameByCPUPerformance

Description

Set auto frame rate by CPU threshold.

Syntax

```
void KSetAutoDropFrameByCPUPerformance( HANDLE h, bool bEnable = false, DWORD
dwCPUPerformance = 100 );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
bEnable	**bool**	[in] Enable or disable.
dwCPUPerformance	**DWORD**	[in] Drop frames when reach this CPU Performance.

Returns

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KSetBitRate

Description

Set video server's BitRate.

Syntax

```
void KSetBitRate(HANDLE h, DWORD dwBitRate);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwBitRate	**DWORD**	[in] BitRate value.

Returns

No return value.

Remarks.

Save reboot is required for some video server.

BitRate	**Description**
BITRATE_28K (0)	28K Bits per second
BITRATE_56K (1)	56K Bits per second
BITRATE_128K (2)	128K Bits per second
BITRATE_256K (3)	256K Bits per second
BITRATE_384K (4)	384K Bits per second
BITRATE_500K (5)	500K Bits per second
BITRATE_750K (6)	750K Bits per second
BITRATE_1000K (7)	1M Bits per second
BITRATE_1200K (8)	1.2M Bits per second
BITRATE_1500K (9)	1.5M Bits per second
BITRATE_2000K (10)	2M Bits per second
BITRATE_2500K (11)	2.5M Bits per second
BITRATE_3000K (12)	3M Bits per second

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KPlay(h);
               }
          }
     }
}
. . . . .
KSetBitRate(h, BITRATE_1500K);
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KGetVideoConfig, (Back To Miscellaneous List)

KSetBrightness

Description

Set video server's brightness.

Syntax

```
void KSetBrightness(HANDLE h, DWORD dwBrightness);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwBrightness	**DWORD**	[in] Brightness value.

Returns

No return value.

Remarks.

Brightness value is from 0 (low) to 100 (high).

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
```

```
            }
        }
    }
}
. . . . .
KSetBrightness(h, dwBrightness);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetVideoConfig, (Back To Miscellaneous List)

KSetContrast

Description

Set video server's contrast.

Syntax

```
void KSetContrast(HANDLE h, DWORD dwContrast);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwContrast	**DWORD**	[in] Contrast value.

Returns

No return value.

Remarks.

Contrast value is from 0 (low) to 100 (high).

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
              KPlay(h);
```

```
            }
        }
    }
}
. . . . .
KSetContrast(h, dwContrast);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KGetVideoConfig, (Back To Miscellaneous List)

KSetCurrentPosition

Description

Set current position in processing file. (sec)

Syntax

```
void KSetCurrentPosition( HANDLE h, DWORD dwPosition );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()
dwPosition	**DWORD**	[in] The assigned new position (second)

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

(Back To Miscellaneous List)

KSetFPS

Description

Set video server's FPS.

Syntax

```
void KSetFPS(HANDLE h, DWORD dwFPS);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwFPS	**DWORD**	[in] FPS value.

Returns

No return value.

Remarks.

This function is used when video server is at constant frame rate mode. Some video server may require you to save reboot to affect the change.

Constant FPS value for NTSC – 30, 15, 10, 7, 6, 5, 4, 3, 2, 1.

Constant FPS value for PAL – 25, 12, 8, 6, 5, 4, 3, 2, 1.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
```

```
        {
             if(KStartStream(h))
             {
               KPlay(h);
             }
        }
     }
}
. . . . .
KSetFPS(h, 1);
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KSetVariableFPS, KGetVideoConfig, (Back To Miscellaneous List)

KSetHue

Description

Set video server's hue.

Syntax

```
void KSetHue(HANDLE h, DWORD dwHue);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwHue	**DWORD**	[in] Hue value.

Returns

No return value.

Remarks.

Saturation	Description
0	-180 Degree
……	……
50	0 Degree
……	……
100	180 Degree

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
```

```
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KPlay(h);
               }
          }
      }
}
. . . . .
KSetHue(h, dwHue);
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KGetVideoConfig, (Back To Miscellaneous List)

KSetResolution

Description

Set video server's resolution.

Syntax

```
void KSetResolution(HANDLE h, DWORD dwResolution);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwResolution	**DWORD**	[in] Resolution value.

Returns

No return value.

Remarks.

Save reboot is required for some video server.

Resolution	Description
NTSC_720x480 (0)	NTSC - 720 x 480
NTSC_352x240 (1)	NTSC - 352 x 240.
NTSC_160x112 (2)	NTSC - 160 x 112.
PAL_720x576 (3)	PAL - 720 x 576
PAL_352x288 (4)	PAL - 352 x 288
PAL_176x144 (5)	PAL - 176 x 144.
PAL_176x120 (6)	PAL - 176 x 120
NTSC_640x480 (64)	NTSC - 640 x 480.
PAL_640x480 (192)	PAL - 640 x 480.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KPlay(h);
               }
          }
     }
}
. . . . .
KSetResolution(h, NTSC_720x480);
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KGetVideoConfig, (Back To Miscellaneous List)

KSetSaturation

Description

Set video server's saturation.

Syntax

```
void KSetSaturation(HANDLE h, DWORD dwSaturation);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwSaturation	**DWORD**	[in] Saturation value.

Returns

No return value.

Remarks.

Saturation value is from 0 (low) to 100 (high).

Saturation	**Description**
0	0%
……	……
50	100%
……	……
100	200%

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
```

```
{
     if(KSetMediaConfig(h, &mcc))
     {
          if(KConnect(h))
          {
               if(KStartStream(h))
               {
                  KPlay(h);
               }
          }
      }
}
. . . . .
KSetSaturation(h, dwSaturation);
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KGetVideoConfig, (Back To Miscellaneous List)

KSetVariableFPS

Description

Set video server's FPS (only works in Variable Frame Rate mode).

Syntax

```
void KSetVariableFPS(HANDLE h, DWORD dwVariableFPS);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
dwVariableFPS	**DWORD**	[in] FPS value.

Returns

No return value.

Remarks.

Variable FPS value for NTSC – 30, 6, 3, 1.

Variable FPS value for PAL – 25, 5, 3, 1.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
```

```
                KPlay(h);
            }
        }
    }
}
. . . . .
KSetVariableFPS(h, 1);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

See Also

KSetFPS, KGetVideoConfig, (Back To Miscellaneous List)

KSetVideoConfig

Description

Set video server's config.

Syntax

```
bool KSetVideoConfig(HANDLE h, MEDIA_VIDEO_CONFIG* VideoConfig);
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface().
VideoConfig	**MEDIA_VIDEO_CONFIG***	[in] the pointer to the strut MEDIA_VIDEO_CONFIG that contain the Video Server Config.

Returns

If the function succeeds, then video server information is set to the structure.

If the function fails, config on video server is remain unchanged.

Remarks.

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll & relate AVC adaptors**

Example

```
HANDLE h = KOpenInterface();
. . . . .
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
```

```
            {
               MEDIA_VIDEO_CONFIG mvc;
               memset(&mvc, 0x00, sizeof(MEDIA_VIDEO_CONFIG));
               . . . . .
               KSetVideoConfig(h, &mvc);
            }
         }
      }
}
. . . . .
if(NULL != h)
{
     KStop(h);
     KStopStream(h);
     KDisconnect(h);
     KCloseInterface(h);
     h = NULL;
}
```

See Also

KGetVideoConfig, (Back To Miscellaneous List)

KStartDecodeMode

Description

Start the SDK with a decoder mode.

Syntax

```
bool KStartDecodeMode( HANDLE h );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

Return true when function success.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

KStopDecodeMode , KDecodeFrame ,(Back To Miscellaneous List)

KStopDecodeMode

Description

Stop the SDK decoder mode.

Syntax

```
void KStopDecodeMode( HANDLE h );
```

Parameters

Name	*Type*	*Description*
h	**HANDLE**	[in] The handle returned by KOpenInterface()

Returns

No return value.

Remarks

Requirements

Header file: **SDK10000.h**

Import library: **KMpeg4.lib**

Runtime DLL: **KMpeg4.dll**

Example

See Also

KStartDecodeMode, KDecodeFrame, (Back To Miscellaneous List)

4 Error Code

IF Function Return is Fail, The Caller can Call `KGetLastError()` to get the Error Reason.

The Error Reason Code: (Start from `0`)

Following is the Error Code Definition:

Error Code	Description
SDK10000_ERROR_NO_ERROR (0)	No Error
SDK10000_ERROR_AVC_ADAPTOR_ATTACHED_ALREADY (1)	Adaptor already attached.
SDK10000_ERROR_CODEC_ADAPTOR_ATTACHED_ALREADY (2)	CODEC adaptor already attached.
SDK10000_ERROR_FILE_ADAPTOR_ATTACHED_ALREADY (3)	File adaptor already attached (FRAW)
SDK10000_ERROR_DRAWER_ADAPTOR_ATTACHED_ALREADY (4)	Drawer adaptor already attached (DGDI or DXDRAW)
SDK10000_ERROR_CAN_NOT_LOAD_AVC_ADAPTOR (11)	Make sure you place your adaptors at right place with KMpeg4.dll.
SDK10000_ERROR_CAN_NOT_LOAD_CODEC_ADAPTOR (12)	Make sure you place your CODEC adaptor at right place with KMpeg4.dll.
SDK10000_ERROR_CAN_NOT_LOAD_FILE_ADAPTOR (13)	Make sure you place your File adaptors at right place with KMpeg4.dll.
SDK10000_ERROR_CAN_NOT_LOAD_DRAWER_ADAPTOR (14)	Make sure you place your Drawer adaptor at right place with KMpeg4.dll.
SDK10000_ERROR_BAD_URL_COMMAND (22)	Unable to get URL result or URL command error.
SDK10000_ERROR_BAD_IP_OR_PORT (23)	Unable to create URL connection.
SDK10000_ERROR_BAD_PARAMETER (24)	Bad parameter is passing into functions.

SDK10000_ERROR_NO_CONNECTION (25)	No connection is made from client to device, KConnect must perform.
SDK10000_ERROR_TCP10_NOT_SUPPORTED_SOUND_DEVICE (26)	Sound is not support on device.
SDK10000_ERROR_AUDIO_TOKEN_WAS_TAKEN (27)	Audio token is taken by others.
SDK10000_ERROR_HAVE_NO_AUDIO_TOKEN (28)	No Audio Token.
SDK10000_ERROR_FAIL_TO_INIT_AUDIO_CAPTURE_DEVICE (29)	No Microphone.
SDK10000_ERROR_CREATE_FILE_FAIL (30)	Fail to create file, please check available disk space.
SDK10000_ERROR_CONNECT_FAIL (31)	Connect fail. AVC adaptor might not load successfully.

5 Sample Codes

Initialization

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

Preview

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

MEDIA_RENDER_INFO mri;
mri.rect.top = ntop;
mri.rect.right = nright;
mri.rect.left = nleft;
mri.rect.bottom = nbottm;
mri.hWnd = hWnd;
if(h)
{
    KSetRenderInfo(h, &mri);
}

if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
```

```
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

Record

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);
if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KStartRecord(h, “c:\\rec.raw”);
. . . . .
if(NULL != h)
{
    MP4FILE_RECORD_INFO mri;
    memset(&mri, 0x00, sizeof(MP4FILE_RECORD_INFO));
    KStopRecord(h, &mri);
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

Playback

```
HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_PLAYBACK;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

MEDIA_RENDER_INFO mri;
mri.rect.top = ntop;
mri.rect.right = nright;
mri.rect.left = nleft;
mri.rect.bottom = nbottm;
mri.hWnd = hWnd;
if(h)
{
    KSetRenderInfo(h, &mri);
}

if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KStop(h);
```

```
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

PTZ – Pan/Tilt/Zoom

```
MEDIA_PTZ_PROTOCOL m_mPTZ;
memset(&m_mPTZ, 0x00, sizeof(MEDIA_PTZ_PROTOCOL));
m_mPTZ. dwAddressID = 1;
m_mPTZ. nSourceType = 1;
strcpy(m_mPTZ.szProtocolFileName, “c:\\CAM-6100_Pelco-P.ptz”);


HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
mcc.ContactType = CONTACT_TYPE_CONTROL;
. . . . .

if(NULL != h)
{
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
KPTZLoadProtocol(m_hNet,&m_mPTZ);
KPTZMove(m_hNet, m_mPTZ.dwAddressID, 1, PTZ_MOVE_DOWN_LEFT);
. . . . .
if(NULL != h)
{
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}

if(NULL != hPTZ)
{
    PTZCloseInterface(hPTZ);
}
```

Motion Detection

```
void CALLBACK MotionDetectionCB(DWORD UserParam, bool bMotion1,
bool bMotion2, bool bMotion3)
{
    if(bMotion1)
    {
        printf(“Motion 1\n”);
    }
    if(bMotion2)
    {
         printf(“Motion 2\n”);
    }
    if(bMotion3)
    {
         printf(“Motion 3\n”);
    }
}

. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{

    . . . . .
}


HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

if(NULL != h)
{
```

```
    KSetMotionDetectionCallback(h, (DWORD)this, MotionDetectionCB);
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
     }
}
. . . . .
if(h)
{
    KSetMotionInfo(h, &mmi);
}
. . . . .
if(NULL != h)
{
    KSetMotionDetectionCallback(h, (DWORD)this, NULL);
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```

Digital I/O

```
void CALLBACK DICB(DWORD UserParam, bool bDI1, bool bDI2)
{
    if(bDI1)
    {
        printf(“DI 1\n”);
    }
    if(bDI2)
    {
        printf(“DI 2\n”);
    }
}
. . . . .
HANDLE h = KOpenInterface();
if(NULL != h)
{

    . . . . .
}


HANDLE h = KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, “c:\\rec.raw\0”);

if(NULL != h)
{
    KSetDICallback(h, (DWORD)this, DICB);
    if(KSetMediaConfig(h, &mcc))
    {
        if(KConnect(h))
        {
```

```
            if(KStartStream(h))
            {
               KPlay(h);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    KSetDICallback(h, (DWORD)this, NULL);
    KStop(h);
    KStopStream(h);
    KDisconnect(h);
    KCloseInterface(h);
    h = NULL;
}
```