VNDP VIVOTEK Network Development Platform

VIVOTEK NETWORK DEVELPMENT PLATFORM

ModuleName

Version 6.3.1.0

2017/01/20

TABLE OF CONTENTS

1. Overview

1.1 Introduction

This document describes how to use DataBroker module to get video and audio frames from remote servers. On the other hand, you can get video and audio packets using your own network module and use this DataBroker module to help unpack and parse the packets you have received.

1.2 File Structure

File	Description
doc\VNDP_DataBroker_API.pdf	This manual document
lib\d_DataBroker.lib	The dynamic linking library
lib\DataBroker.dll	The dynamic runtime library
inc\DataBroker.h	Header file
inc\SrvTypeDef.h	Common definition file
inc\datapacketdef.h	Data packet definition file

2. Programmer's Guide

You can use DataBroker module in two ways.

2.1 DataBroker and Connection

In the first situation, if you can't get packets from server through network by yourself, you can use the network client embedded in DataBroker module to help you. DataBroker will unpack and parse the received packets automatically. So it's easy for you to get audio and video frames from server through the DataBroker module.

Call DataBroker_Initial function to acquire a DataBroker handle, you should specify the maximum connections that the DataBroker can handle when initializing. Using this DataBroker handle, you can call DataBroker_CreateConnection to create as many connections as you need (not exceed the maximum number of connections you specified). Every connection instance can establish a connection to a remote server (call DataBroker_Connect). In one hand, through the AV callback function, you can get contiguous frame data from the remote server. On the other hand, through the status callback function, you can receive various kind of connection status (connection established, starting to receive data, error conditions, etc).

2.2 Input Network Packet

The second situation is, if you want to implement the network client in your own way, using DataBroker module can help you unpack and parse media packets which you received.

Call DataBroker_CreateInputEx to acquire an input handle. Call DataBroker_InputPacketEx using this Input handle and feeding received media packets into it, then you can get parsed frame data through the AV callback function. And through status callback function, you can get status while DataBroker processing input media packets.

3. Sample Code

In this chapter we will use sample code to introduce the basic functions of DataBroker, they demonstrate the most important concepts while using DataBroker.

If you want to use DataBroker in your application, you can refer to the sample codes in the SDK package.

3.1 ConnectToDevice

Description

Connect to a device which can be a network camera or a video server.

Tips

The **DataBroker_Connect** function uses asynchronous (non-blocking) method to handle connection, and various kind of connection status will be carried through a status callback function rather than return code. You can refer to the SDK package for full sample code.

3.2 ConnectToMultipleStreamDevice

Description

Connect to a device which has multiple streams (for example, IP7161), and show how to connect to the specified stream.

Tips

The stream index is zero-based. The stream index 0 is for stream 1 and so on.

3.3 ConnectToVideoServer

Description

Connect to a video server which has multiple channels (for example, VS2403), and show how to connect to the specified channel.

Tips

The channel index is not zero-based. The value of wCam starts from 1.

3.4 GetDIDONotify

Description

Get the DI/DO information from status callback.

Tips

The DI/DO value also can be get from TMediaDataPacketInfo in the AV callback function.

3.5 TalkWithMicPhone

Description

Besides talking to camera with a microphone, this sample shows how to open a WAV file and send audio data to the device.

3.6 TalkWithWAVFile

Description

Besides talking to camera with a microphone, this sample shows how to open a WAV file and send audio data to the device.

Tips

After calling the **DataBroker_StartTxConnection**, DataBroker will use **TDataBrokerTxCallback** callback function to get audio bitstream.

3.7 SaveToFile

Description

This sample shows how to save the TMediaDataPacketInfoV3 data to a file. You can playback this file using “PlayRawFile” sample which resides in AVSynchronizer directory.

3.8 ConnectMoreThan32Channels

Description

For better performance, we recommend that application uses one DataBroker module to handle at most 32 connections. This sample shows how to connect more than 32 channels using two DataBroker instances.

Tips

A DataBroker can handle multiple connections simultaneously, so the **DataBroker_Initial** should be called only once for initialization purpose and then use this handle to create connection handles.

3.9 Reconnect

Description

This sample shows how to re-connect to devices when the connection is disconnected.

Tips

Do not call serveral DataBroker APIs in the status callback function or AV callback function, such as **DataBroker_Disconnect**, **DataBroker_DeleteConnection**, **DataBroker_Release** and so on. This kind of usage is unsafe and will probably cause module fail or crash.

3.10 GetV3Information

Description

This sample shows how to get media packet information from AV callback function. By casting the ptMediaDatapacket to TMediaDataPacketInfoV3 data type, application can get more information in distinct fields.

Tips

For detail about parsing data in TMediaDataPacketInfoV3, please refer to the document of DataPacketParser for more information.

3.11 SpecifyMediaType

Description

This sample shows how to connect to a device and receive the specified media type (Video only, Audio only or both).

Tips

Application can use dwMediaType in **TDataBrokerConnectionOptions** specfy media type.

3.12 ForceIFrame

Description

This sample shows how to force a device to send an I-Frame immediately. However, if the device’s resource is exhausting, it may ignore this request.

3.13 SSLSupport

Description

This sample shows how to connect to a device over SSL (Secure Sockets Layer).

Tips

Application can successfully connect to a device over SSL only if the device supports SSL based connection.

4. API Reference

This chapter describes the API functions for the DataBroker.

Note that in Windows CE all the interface functions use Unicode strings. And for Linux orWindows 32 platform, non-Unicode strings are used.

4.1 Enumeration

The enumeration used is depicted here.

- EFrameRateOption
- TDataBrokerChannelType
- TDataBrokerConnOptionFlag
- TDataBrokerOptionFlag
- TDataBrokerStatusType
- TExtraOptions
- TMediaAudioMode
- TMediaChangeReason
- TsdrAudioCodec
- TsdrMediaType
- TsdrProtocolType
- TUserPrivilege
- TVideoSignalState

4.1.1 EFrameRateOption

Defines frame rate options for SVC.

```
typedef enum {
                                        eNoneSetting = 0,
                                        eFrameOnly = 1,
                                        eFramelevel1 = 2,
                                        eFramelevel2 = 3,
                                        eFramelevel3 = 4,
                                        eFramelevel4 = 5,
                                        eFramelevel5 = 6,
                                        eFramelevel6 = 7,
                                        eFramelevel7 = 8,
                                        eFramelevel8 = 9
} EFrameRateOption;
```

Values

eNoneSetting

Use original frame rate.

eFrameOnly

Use I-frame only.

eFramelevel1 - eFramelevel8

Frame rate levels.

Remarks

This enumeration is used when connecting to a camera which has SVC capability. It can force the camera to send streaming with different frame rate.

Requirements

DataBroker.h

4.1.2 TDataBrokerChannelType

Defines DataBroker channel type.

```
typedef enum {
                                        eAudioChannel,
                                        eVideoChannel
} TDataBrokerChannelType;
```

Values

eAudioChannel

Audio channel.

eVideoChannel

Video channel.

Remarks

Requirements

DataBroker.h

4.1.3 TDataBrokerConnOptionFlag

Defines option flags which are used in **TDataBrokerConnectionOptions**.

```
typedef enum {
                                        eConOptCam = 0x0001,
                                        eConOptVSize = 0x0002,
                                        eConOptQuality = 0x0004,
                                        eConOptHttpPort = 0x0008,
                                        eConOptProtocolAndMediaType = 0x0100,
                                        eConOptVideoCodec = 0x0200,
                                        eConOptAudioCodec = 0x0400,
                                        eConOptStatusCallback = 0x0800,
                                        eConOptAVCallback = 0x1000,
                                        eConOptStatusContext = 0x2000,
                                        eConOptAVContext = 0x4000,
                                        eConOptTxCallback = 0x0800,
                                        eConOptTxContext = 0x1000,
                                        eConOptAudioEncCodec = 0x2000,
                                        eConOptVideoCodecPri = 0x4000,
                                        eConOptAudioCodecPri = 0x8000,
                                        eConOptAudioSample = 0x10000,
                                        eConOptAudioEncSample = 0x20000,
                                        eConOptSBCallback = 0x40000,
                                        eConOptSBContext = 0x80000
} TDataBrokerConnOptionFlag;
```

Values

eConOptCam

Indicate the wCam field.

eConOptVSize

Indicate the zVSize field.

eConOptQuality

Indicate the dwQuality field.

eConOptHttpPort

Indicate the wHttpPort field.

eConOptProtocolAndMediaType

Indicate the dwProtocolType and dwMediaType fields. You should specify these two values at the same time.

eConOptVideoCodec

Indicate the dwVideoCodec field.

eConOptAudioCodec

Indicate the dwAudioCodec field.

eConOptStatusCallback

Indicate the pfStatus field.

eConOptAVCallback

Indicate the pfAV field.

eConOptStatusContext

Indicate the dwStatusContext field.

eConOptAVContext

Indicate the dwAVContext field.

eConOptTxCallback

Indicate the pfTx field.

eConOptTxContext

Indicate the dwTxContext field.

eConOptAudioEncCodec

Indicate the dwAudioEnc field.

eConOptVideoCodecPri

Indicate the dwVCodecOrder filed.

eConOptAudioCodecPri

Indicate the dwACodecOrder filed.

eConOptAudioSample

Indicate the dwAudioSample field.

eConOptAudioEncSample

Indicate the dwAudioEncSample filed.

eConOptSBCallback

Indicate the pfSB field.

eConOptSBContext

Indicate the dwSBContext field.

Remarks

Option flags are used to indicate which field in TDataBrokerConnectionOptions should take effect. Applications can use a logical OR with appropriate flags.

Requirements

DataBroker.h

4.1.4 TDataBrokerOptionFlag

Defines option flags which are used in TDataBrokerOptions.

```
typedef enum {
                                        eOptEnableProxy = 0x0001,
                                        eOptProxyPort = 0x0002,
                                        eOptProxyName = 0x0004,
                                        eOptEnableIPRestrict = 0x0008,
                                        eOptIPRestrictNum = 0x0010,
                                        eOptIPRestrictList = 0x0020,
                                        eOptConnUseIE = 0x0040,
                                        eOptConnNotUseIE = 0x0080,
                                        eOptConnTimeout = 0x0100,
                                        eOptRWTimeout = 0x0200,
                                        eCreateMonitorThread = 0x0400,
                                        eCreateWokerThread = 0x0800,
                                        eLogFunction = 0x1000

} TDataBrokerOptionFlag;
```

Values

eOptEnableProxy

Indicate the bEnableProxy field.

eOptProxyPort

Indicate the dwProxyPort field.

eOptProxyName

Indicate the szProxyName field.

eOptEnableIPRestrict

Indicate the bEnableIPRestrict field.

eOptIPRestrictNum

Indicate the dwIPRestrictNum field.

eOptIPRestrictList

Indicate the pszIPRestrictList field.

eOptConnUseIE

Use Wininet to connect server. This flag only takes effect in Windows or Windows CE platform.

eOptConnNotUseIE

Not use Wininet to connect server. This flag only takes effect in Windows or Windows CE platform.

eOptConnTimeout

Indicate the dwConnTimeout field.

eOptRWTimeout

Indicate the dwRWTimeout field.

eCreateMonitorThread

Indicate the bCreateMonitorThread field.

eCreateWokerThread

Indicate the bCreateWokerThread field.

eLogFunction

Indicate the pLogFunction field.

Remarks

Option flags are used to indicate which field in TDataBrokerOptions should take effect. Applications can use a logical OR with appropriate flags.

If both eOptConnUseIE and eOptConnNotUseIE are set, only the eOptConnNotUseIE takes effect.

Requirements

DataBroker.h

4.1.5 TDataBrokerStatusType

Defines connection status type.

```
typedef enum {
                                        eOnConnectionInfo,
                                        eOnAuthFailed,
                                        eOnStartMediaChannel,
                                        eOnChannelClosed,
                                        eOnTimeout,
                                        eOnProtocolChanged,
                                        eOnPacketLoss,
                                        eOnDiDo,
                                        eOnLocationChanged,
                                        eOnInputInfo,
                                        eOnOtherError,
                                        eOnStopped,
                                        eOnAudioMode,
                                        eOnChangeMediaType,
                                        eOnAudioDisabled,
                                        eOnAudioUpstreamOccupied,
                                        eOnGetPrivilege,
                                        eOnTxChannelStart,
                                        eOnTxChannelClosed,
                                        eOnControlChannelClosed,
                                        eOnVideoSignalChange,
                                        eOnServiceUnavailable,
                                        eOnAudioUpstreamDisabled,
                                        eOnMediaRange,
                                        eOnMP4Vconfig,
                                        eOnMP4Aconfig,
                                        eOnGAMRConfig,
                                        eOnConnectionOptionError,
                                        eOnProxyAuthFailed,
                                        eOnConnectionType,
} TDataBrokerStatusType;
```

Values

eOnConnectionInfo

Indicate connection info when connecting to the server.

eOnAuthFailed

Can not pass the authorization of server.

eOnStartMediaChannel

Begin to receive media stream.

eOnChannelClosed

Audio or Video channel closed.

eOnTimeout

Audio or Video channel receives data timeout.

eOnProtocolChanged

The protocol of receiving media changed.

eOnPacketLoss

Packet loss.

eOnDiDo

Receiving the digital input alert and digital output status.

eOnLocationChanged

Detecting the change of location.

eOnInputInfo

Indicate the width and height of the image when using Input to unpacketize and parse the receiving packets.

eOnOtherError

Other error occurs.

eOnStopped

The connection stopped.

eOnAudioMode

Notify the audio mode set on server. This notification is only available when connecting to 6000 series servers. The pvParam1 contains the integer value that indicates the audio mod. Please refer to TMediaAudioMode. This status is also notified when someone changes the server audio mode. So it is not subjected to called when connecting.

eOnChangeMediaType

Notify that due to server settings or users' permission, the media is changed. This notification is only available when connecting to 6000 series servers. The pvParam1 contains reason. Please refer to TMediaChangeReason.

eOnAudioDisabled

This status code is similar to eOnChangeMediaType. But when users get this notification, it means only control channel is established. In such case, no audio or video data would be available.

eOnChangeMediaType

Notify that due to server settings or users' permission, the media is changed. This notification is only available when connecting to 6000 series servers. The pvParam1 contains reason. Please refer to TMediaChangeReason.

eOnAudioDisabled

This status code is similar to eOnChangeMediaType. But when users get this notification, it means only control channel is established. In such case, no audio or video data would be available.

eOnAudioUpstreamOccupied

When users start talk and DataBroker finds that the talk-channel is already used by other user, this status code will be sent to users by callback. This notification is only available when connecting to 6000/7000/8000 series servers

eOnGetPrivilege

Notify the privilege for current user. This notification is only available when connecting to 6000 series servers. The privilege type is a ORed double word of the privilege define in TUserPrivilege.

eOnTxChannelStart

Notify the Talk connection is established. This notification is only available when connecting to 6000/7000/8000 series servers.

eOnTxChannelClosed

Notify the Talk connection is closed. This notification is only available when connecting to 6000/7000/8000 series servers.

eOnControlChannelClosed

Notify the control channel has been closed.

eOnVideoSignalChange

Notify that there is no signal for video input of video server model. The parameter would either be signal lost or signal restored.

eOnServiceUnavailable

Notify that the server is now serving 10 streaming clients and no more new client could request for streaming now unless one of the previous connection closed.

eOnAudioUpstreamDisabled

Notify that the upstream channel is turned off by server. The parameter is the new audio mode that server is set currently.

eOnMediaRange

This is the notification for RTSP server's playing range for media. This is only made if the requested media is file based. For live streaming, there will not be such status notified.

eOnMP4Vconfig

This is the MP4 CI value for RTSP. pvParam1 is the CI starting address and pvParam2 is the length for CI.

eOnMP4Aconfig

This is the AAC info value for RTSP. pvParam1 is the DWORD value for the type value.

eOnGAMRConfig

This is the GAMR info value for RTSP. pvParam1 is the sample rate.

eOnConnectionOptionError

This indicates that connection options are different from that in server.

eOnProxyAuthFailed

Notify AP authentication fail.

eOnConnectionType

Notify The connection type. pvParam1's upper 16 bits indicates if this is a dual-stream model or not. If pvParam1's lower 16 bits is 1 means this connection is RTSP and 2 means server push. pvParam2 is the model of server.

Remarks

Requirements

DataBrokerCallbackDef.h

4.1.6 TExtraOptions

Defines the options available when calling **DataBroker_SetConnectionExtraOption**.

```
typedef enum {
                                        eOptQueueSize,
                                        eOptAutoForceI,
                                        eOptDIDOBySet,
                                        eOptSetDIDOValue,
                                        eOpt3KAudioDIDO,
                                        eOptRtspCtrlPort,
                                        eOptServerInfo,
                                        eOptAACInfo,
                                        eOptProxyAuthen,
                                        eOptStreamIndex,
                                        eOptRtspProxyEnable,
                                        eOptRtspProxyInfo,
                                        eOptProtocolMedia,
                                        eOptProtocolRollNext,
                                        eOptUpdUserNamePwd,
                                        eOptRtspMcastProtocolRollingTimeout,
                                        eOptRtspUDPProtocolRollingTimeout,
                                        eOptTalkWithoutStreaming,
                                        eOptSSLEnable,
                                        eOptRTSPForward,
                                        eOptRTSPBackward,
                                        eOptReqFieldModeInfo,
                                        eOptUseSharedThread,
                                        eOptUseV3Callback,
                                        eOptTalkWithVideoServerMap,
                                        eOptRTSPFrameLevel,
                                        eOptHTTPFrameInterval,
                                        eOptNormalPostEnable,
                                        eOptWindowsADAuth,
                                        eOptH264NALLen,
                                        eOptServerPushWithMJPEG,
} TExtraOptions;
```

Values

eOptQueueSize

To set queue size for connection or input. It is effecttive before the queue is created. For input, please set this before DataBroker_SetInputOptions is called. For connection, set it before DataBroker_SetConnectionOptions is called. The dwParam1 is for Video Q size, and dwParam2 is for Audio Q Size. The maximum value for video is 60, and the maximum for Audio is 20.

eOptAutoForceI

To enable or disable the auto-force I function. It's only workable for 3000 server and the protocol must not be HTTP. When the dwParam1 is true means to enable and in such case, the dwParam2 is the period in milliseconds. The minimum value could be set is 100 ms. When disable period, param2 is not used. This option is only applicable for connection, not input.

eOptDIDOBySet

For the old model of server, the video bit stream does not carry DI/DO information. If application could get DI/DO from other channel (such as by http command), the application could set the value manually. This option enables the module to embed the set value into video packet. The dwParam1 contain True or False to enable or disable the function. It only works for 2000 and 3000 server. The caller must take care for if the firmware already sends DI/DO. If so, it's not necessary to use this path to set DI/DO value.

eOptSetDIDOValue

Set DI/DO value for a connection or input. When set, dwParam1 contains the DI value, and dwParam2 contains DO value. Each bit represents a single DI or DO. The LSB means DI 0 or DO 0, bit 1 means DI 1 or DO 1, and so on.

eOpt3KAudioDIDO

Should the module synchronize the DI/DO value in audio with that in video? For 3000 server, even if the sever sends DI/DO. The values are contains only in the video packets. With this option, the module will retrieve the DI/DO values from video and set them in audio packet. The dwParam1 contains True or False to enable or disable this function.

eOptRtspCtrlPort

Set the control port for the RTSP server and dwParam1 is the control port.

eOptServerInfo

Set server information. dwParam1 points to a TSrvDepResource_SysInfo structure. dwParam2 is the options to set. See EOptSysInfo in SrvDepResource.h.

eOptAACInfo

Set AAC information for Input-channel. dwParam1 points to TAACExtInfo.

eOptProxyAuthen

Settings for proxy authentication, dwParam1 is the user name, dwParam2 is password eOptStreamIndex. For dual-stream models, indicates which connection to connect. dwParam1 is the index of stream. The index starts from zero.

eOptRtspProxyEnable

The dwParam1 is TRUE for enable and FALSE for disable RTSP proxy.

eOptRtspProxyInfo

dwParam1 is proxy address, and dwParam2 is the proxy port for RTSP.

eOptProtocolMedia

Specifying the protocol and media type. dwParam1 is protocol, and dwParam2 is media type. Value of 0xFFFFFFFF means do not set it

eOptProtocolRollNext

Specifies the next protocol to be tried when protocol rolling. dwParam1 the protocoltype.

eOptUpdUserNamePwd

The dwParam1 is user name, and dwParam2 is password.

eOptRtspMcastProtocolRollingTimeout

Assign the timeout value of multicast protocol rolling. dwParam1 is timeout value.

eOptRtspUDPProtocolRollingTimeout

Assign the timeout value of UDP protocol rolling. dwParam1 is timeout value.

eOptTalkWithoutStreaming

DataBroker can support talk to camera without request the live streaming. Application should specified the camera series in dwParam1. dwParam1 is 0 for 7k/8k cameras and 1 for 6k cameras.

eOptSSLEnable

Applications can set dwParam1 toTRUE to enable SSL ecryption, or FALSE to disable. dwParam2 is reserved.

eOptRTSPForward

Applications can use both dwParam1 and dwParam2 to control the play speed in forward direction. For example, when setting dwParam1 to 2 and dwParam2 to 1, then the play speed is 2 which is dwParam1 devided by dwParam2. The dwParam2 should not be 0.

eOptRTSPBackward

Applications can use both dwParam1 and dwParam2 to control the play speed in backward direction. For example, when setting dwParam1 to 2 and dwParam2 to 1, then the play speed is 2 which is dwParam1 devided by dwParam2. The dwParam2 should not be 0.

eOptReqFieldModeInfo

When connecting to VS series cameras and users need to know whether the incoming frames'field/frame mode information, they should set dwParam1 to TRUE. DataBroker will callback the information.

eOptUseSharedThread

If dwParam1 is TRUE, DataBroker uses the shared thread to connect, otherwise create its own thread(default behavior).

eOptUseV3Callback

Reserved. Applications can pass TDataBrokerAVCallbackV3 callback function through dwParam1.

eOptTalkWithVideoServerMap

When connecting 8k VS servers, uses dwParam1 to specify the channel to talk. Each bit represents the channel number. The LSB means channel 0, bit 1 means channel 1, and so on.

eOptRTSPFrameLevel

Set frame rate level. Applications can pass EFrameRateOption through dwParam1.

eOptHTTPFrameInterval

Set interval value, which is between 0 and 6000. Applications can pass the value through dwParam1.

eOptNormalPostEnable

Applications can set dwParam1 to TRUE to enable normal POST; or FALSE to use tunnel POST. Applcations can set dwParam2 to TRUE to enable keep-alive function; or FALSE to disable it.

eOptWindowsADAuth

Applications can set dwParam1 to TRUE to enable Windows AD authentication.

eOptH264NALLen

If applications set dwParam1 to TRUE, DataBroker will modify the start code to 00 00 00 01 in NAL.

eOptServerPushWithMJPEG

In MJPEG streaming, applications can set dwParam1 to TRUE to force DataBroker use server-push method.

Remarks

Requirements

DataBroker.h, SrvDepResource.h

4.1.7 TMediaAudioMode

Defines the audio mode currently set on server. The notification of this mode is only available when connecting to 6000 series servers.

```
typedef enum {
                                        eFullDuplex = 0x0001,
                                        eHalfDuplex = 0x0002,
                                        eTalkOnly = 0x0003,
                                        eListenOnly = 0x0004
} TMediaAudioMode;
```

Values

eFullDuplex

Server is in full-duplex mode, which enables users to listen and talk simultaneously.

eHalfDuplex

Server is in half-duplex mode, which allows users to be either talk or listen. When talking, the downstream will be disabled.

eTalkOnly

Server is in talk-only mode, which allows users to talk but not listen. Users might get eOnChangeMediaType/ eOnAudioDisabled status callback if selects video/audio or audio only mode when connecting.

eListenOnly

Server is in listen-only mode, which allows users to listen but not talk.

Remarks

For 7000/8000 series servers, the audo mode is always in Full Duplex.

Requirements

DataBroker.h

4.1.8 TMediaChangeReason

Defines the reason why media type is changed. This reason code is only available when connecting to 6000 series servers.

```
typedef enum {
        eNoPermission,
        eModeNotSupport
} TMediaChangeReason;
```

Values

eNoPermission

The permission of current user is not allowed to open downstream or upstream audio connection.

eModeNotSupport

The server’s audio mode is set to talk-only or audio disabled mode so users are not allowed to establish downstream audio connection. Or the server is set to listen-only or disabled mode so users are not allowed to establish upstream audio connection.

Remarks

Requirements

DataBroker.h

4.1.9 TsdrAudioCodec

Defines audio codec types.

```
typedef enum {
                                        eACodecNone = 0x0000,
                                        eACodecG7221 = 0x0100,
                                        eACodecG729A = 0x0200,
                                        eACodecAAC = 0x0400,
                                        eACodecGAMR = 0x0800,
                                        eACodecG711 = 0x1000
} TsdrAudioCodec;
```

Values

eACodecNone

No audio codec.

eACodecG7221

G.722.1.

eACodecG729A

G.729A.

eACodecAAC

AAC (stereo).

eACodecGAMR

GAMR, used in RTSP IP camera and server.

eACodecG711

G.711.

Requirements

SrvTypeDef.h

4.1.10 TsdrMediaType

Defines media types.

```
typedef enum {
                                        emtAudio = 1,
                                        emtVideo = 2,
                                        emtTransmitAudio = 4,
                                        emtMetaData = 8
} TsdrMediaType;
```

Values

emtAudio

Audio.

emtVideo

Video.

emtTransmitAudio

Transmitted audio.

emtMetaData

Metadata.

Remarks

Requirements

SrvTypeDef.h

4.1.11 TsdrProtocolType

Defines protocol types.

```
typedef enum {
                                        eptHTTP,
                                        eptTCP,
                                        eptUDP
                                        eptScalableMULTICAST = eptMULTICAST,
                                        eptBackchannelMULTICAST
} TsdrProtocolType;
```

Values

eptHTTP

HTTP.

eptTCP

TCP.

eptUDP

UDP.

eptMULTICAST

Multicast.

eptScalableMULTICAST

Multicast.

eptBackchannelMULTICAST

Multicast.

Remarks

Requirements

SrvTypeDef.h

4.1.12 TUserPrivilege

Defines the privilege for a user. This privilege value is only available when connecting to 6000 series servers.

```
typedef enum {
                                        ePrivelegeDIDO = 0x00000001
                                        eModeNotSupport = 0x00000002
                                        ePrivelegeTALK = 0x00000004
                                        ePrivelegeCAMCTRL = 0x00000008
                                        ePrivelegeCONF = 0x00000080
                                        ePrivelegeAll =0xFFFFFFFF
} TUserPrivilege;
```

Values

ePrivilegeDIDO

Users could set DO and retrieve DI value from server. Note: this flag might be renamed to DO only because the DI is carried in video stream. So there is no reason to limit users to get DI value.

ePrivilegeLISTEN

Users could open the downstream audio connection to server. Hence they could listen to the live audio sent via server.

ePrivilegeTALK

Users could open the upstream audio connection to server. Hence they could send server audio data. Note: Talk capability is determined by this privilege and the server’s audio mode. Both must be turn on to enable talking.

ePrivelegeCAMCTRL

Users could control the camera control. This includes Pan/Tilt/Zoom/Focus. It depends on the server model.

ePrivelegeCONF

Users could set the configuration of the servers.

ePrivelegeAll

Users have the full access to the camera.

Remarks

Requirements

DataBroker.h

4.1.13 TVideoSignalState

Definees the signal state when callback notify that the video signal changes. The callback would be called whenever the signal state changes, so if the signal lost at certain time, the signal will be called once for signal lost, and would not be called until signal restored.

```
typedef enum {
                                                  eSignalRestored = 0,
                                                  eSignalLost = 1
} TVideoSignalState;
```

Values

eSignalRestored

The signal restored.

eSignalLost

The signal is lost.

Remarks

Requirements

DataBroker.h

4.2 Callback Function

The Callback function is depicted here.

- TDataBrokerAVCallback
- TDataBrokerNetPacketCallback
- TDataBrokerStatusCallback
- TDataBrokerTxCallback
- TDataBrokerAVCallbackV3
- TDataBrokerLogFunction

4.2.1 TDataBrokerAVCallback

The TDataBrokerAVCallback function is the callback function which is used to receive audio and video frames. The name TDataBrokerAVCallback is a placeholder for the application-specified function.

```
typedef SCODE (*TDataBrokerAVCallback)(
                                DWORD_PTR             dwContext,
                                TMediaDataPacketInfo  *pMediaDataPacket
);
```

Parameters

dwContext

[in] Data to be passed to callback function.

pMediaDataPacket

[in] Pointer to a received media packet.

Return Values

S_OK

Receive the audio or video frame successfully.

DATABROKER_S_FRAME_NOT_HANDLED

Could not handle this frame at this time.

Remarks

Requirements

DataBrokerCallbackDef.h

4.2.2 TDataBrokerNetPacketCallback

The TDataBrokerNetPacketCallback is the callback function which is used to notify applications a media packet comes, and let applications can directly access its data. The name TDataBrokerNetPacketCallback is a placeholder for the application-specified function.

```
typedef SCODE (*TDataBrokerNetPacketCallback) (
                                            DWORD_PTR       dwContext,
                                            DWORD           dwMediaType,
                                            DWORD           dwLen,
                                            BYTE            *pbyPacket
);
```

Parameters

dwContext

[in] Data to be passed to callback function.

dwMediaType

[in] The media type of this packet, possible types are defined in TsdrMediaType.

dwLen

[in] The data length of the packet

pbyPacket

[in] The packet data. Note the pointer is no longer valid after return, application needs to copy the content to its own buffer

Return Values

Return S_OK if successful or an error value otherwise.

Remarks

Requirements

DataBrokerCallbackDef.h

4.2.3 TDataBrokerStatusCallback

The TDataBrokerStatusCallback is the callback function which is used to report the status of DataBroker object. The name TDataBrokerStatusCallback is a placeholder for the application-specified function.

```
typedef SCODE (*TDataBrokerStatusCallback)(
                                        DWORD_PTR                   dwContext,
                                        TDataBrokerStatusType       tStatusType,
                                        PVOID                       pvParam1,
                                        PVOID                       pvParam2
);
```

Parameters

dwContext

[in] Data to be passed to callback function.

tStatusType

[in] The status type

pvParam1

[in] The first parameter for corresponding status type . The meaning of this value is determined by tStatusCode parameter.

pvParam2

[in] The second parameter for corresponding status type. The meaning of this value is determined by tStatusCode parameter.

Return Values

Return S_OK for most cases. When the status code is eOnProtocolChanged, return DATABROKER_S_STOPCONNECTION to inform DataBroker to stop The connection, else to allow server to change to HTTP.

Error code

DATABROKER_E_HTTP_READ_ERROR

Read web page from server error.

DATABROKER_E_HTTP_CONNECT_FAILED

Could not connect to server by http protocol.

DATABROKER_E_CONTROL_CHANNEL_CONNECT_FAILED

Control channel could not be established

DATABROKER_E_INVALID_ID

the remote ID passed in is not a correct ID assigned by 3000 servers. (Only applied to 3000 servers)

DATABROKER_E_OUT_OF_MEMORY

The module doesn’t have enough memory to create resources.

DATABROKER_E_CONTROL_CHAN_VER

The server’s control channel supports older message version from the version this module is used.

Remarks

Status code	pvParam1	pvParam2
eOnConnectionInfo	Connection information (*TDataBrokerConnInfo)	(None)
eOnAuthFailed	(None)	(None)
eOnStartMediaChannel	(None)	(None)
eOnChannelClosed	Channel type (TDataBrokerChannelType)	(None)
eOnTimeout	Channel type (TDataBrokerChannelType)	(None)
eOnProtocolChanged	Original protocol(TsdrProtocolType)	New protocol(TsdrProtocolType)
eOnPacketLoss	Numbers lost (DWORD)	Media type(TsdrMediaType)
eOnDiDo	DI/DO changes (DWORD)	DI/DO values (DWORD)
eOnLocationChanged	New location (char*)	(None)
eOnInputInfo	Image width (DWORD)	Image height (DWORD)
eOnOtherError	Error code	(None)

eOnStopped	(None)	(None)
eOnAudioMode	Audio mode (TMediaAudioMode)	(None)
eOnChangeMediaType	Change reason (TMediaChangeReason)	(None)
eOnAudioDisabled	Disabled reason (TMediaChangeReason)	(None)
eOnAudioUpstreamOccupied	(None)	(None)
eOnGetPrivilege	Privilege (or-ed valued of TUserPrivilege)	(None)
eOnTxChannelStart	(None)	(None)
eOnTxChannelClosed	(None)	(None)
eOnControlChannelClosed	(None)	(None)
eOnVideoSignalChange	(None)	(None)
eOnServiceUnavailable	(None)	(None)
eOnAudioUpstreamDisabled	Audio mode(TMediaAudioMode)	(None)
eOnMediaRange	The start time for the period in second (relative value)	The end time for theperiod in second (relative value)
eOnMP4Vconfig	The stating address of CI.	The length for CI.
eOnMP4Aconfig	This is the AAC info value for RTSP	(None)
eOnGAMRConfig	This is the sample rate for GARM	(None)
eOnConnectionOptionError	(None)	(None)
eOnProxyAuthFailed	(None)	(None)
eOnConnectionType	ConnectionType(DWORD)	The model name for server.

DI/DO changes

Each bit is used to indicate the change of DI alert and DO, DI alert is the lower 16 bits, DO is the higher 16 bits. The LSB indicates the first one. It supports four digital input sources and two digital outputs at most in the present.

DI/DO values

Each bit is used to indicate the value (H/L) of DI alert and DO, DI alert is the lower 16 bits, DO is the higher 16 bits. The LSB indicates the first one. It supports four digital input sources and two digital outputs at most in the present.

Requirements

DataBroker.h

4.2.4 TDataBrokerTxCallback

The TDataBrokerTxCallback is the callback function which is used to used to transmit media stream. It's only used for 6000/7000/8000 servers. The name TDataBrokerTxCallback is a placeholder for the application-specified function.

```
typedef SCODE (*TDataBrokerTxCallback)(
                                        DWORD_PTR          dwContext,
                                        BYTE               **ppbyDataBuffer,
                                        DWORD              *pdwLen,
                                        DWORD              *pdwDataTimePeriod
);
```

Parameters

dwContext

[in] Data to be passed to callback function.

ppbyDataBuffer

[out] The data to be sent

pdwLen

[out] The length of the valid data held in ppbyDataBuffer

pdwDataTimePeriod

[out] The total time length for the data held in ppbyDataBuffer. This value is used tocontrol the data rate sent to server

Return Values

Return S_OK for normal case. If the return value is DATABROKER_S_NOMORE_CALLBACK, this module won't make callback any more. And the ppbyDataBuffer,pdwLen, and pdwDataTimePeriod are all ignored. Users must push the to be transmitted data by calling DataBroker_InputTxPacket.

Remarks

Users must give correct pdwDataTimePeriod to avoid server from being flooded by audio data packet. The value is counted like this: the buffer contains 5 frames and each frame contains 10 milliseconds encoded audio data. Then the pdwDataTimePeriod should be 5 * 10 = 50 milliseconds.

Requirements

DataBrokerCallbackDef.h

4.2.5 TDataBrokerAVCallbackV3

The TDataBrokerAVCallbackV3 function is the callback function which is used to receive audio and video frames. The name TDataBrokerAVCallbackV3 is a placeholder for the application-specified function.

```
typedef SCODE (*TDataBrokerAVCallbackV3)(
                                        DWORD_PTR               dwContext,
                                        TMediaDataPacketInfoV3  *pMediaDataPacket
);
```

Parameters

dwContext

[in] Data to be passed to callback function.

pMediaDataPacket

[in] Pointer to a received media packet.

Return Values

S_OK

Receive the audio or video frame successfully.

DATABROKER_S_FRAME_NOT_HANDLED

Could not handle this frame at this time.

Remarks

Requirements

DataBrokerCallbackDef.h

4.2.6 TDataBrokerLogFunction

Reseerved. The TDataBrokerLogFunction function is the callback function which is used to receive internal log message from DataBroker. The name TDataBrokerLogFunction is a placeholder for the application-specified function.

```
typedef SCODE (*TDataBrokerLogFunction)(
                                        const char          *module,
                                        int                 level,
                                        const char          *format,
                                        va_list             argptr
);
```

Parameters

module

[out] Reserved.

level

[out] Reserved.

format

[out] Reserved.

argptr

[out] Reserved.

Return Values

Applications should return S_OK for normal case.

Remarks

Requirements

DataBrokerCallbackDef.h

4.3 Data Structure

The data structure is depicted here.

- TDataBrokerConnectionOptions
- TDataBrokerConnInfo
- TDataBrokerInputOptions
- TDataBrokerOptions
- TDataBrokerRTSPPlayOptions
- TDataBrokerSockFDInfo
- TDataBrokerONVIFOptions
- TDataBrokerSVCLayerInfo
- TDataBrokerH264Info
- TDataBrokerMJPEGInfo
- TDataBrokerMPEG4Info
- TAACExtInfo

4.3.1 TDataBrokerConnectionOptions

Describes the connection options.

```
typedef struct {
                        WORD                        wCam;
                        char                        zVSize[MAX_VSIZE + 1];
                        DWORD                       dwQuality;
                        TDataBrokerStatusCallback   pfStatus;
                        TDataBrokerAVCallback       pfAV;
                        TDataBrokerTxCallback       pfTx;
                        DWORD_PTR                   dwStatusContext;
                        DWORD_PTR                   dwAVContext;
                        DWORD_PTR                   dwTxContext;
                        WORD                        wHttpPort;
                        DWORD                       dwProtocolType;
                        DWORD                       dwMediaType;
                        DWORD                       dwVideoCodec;
                        DWORD                       dwAudioCodec;
                        DWORD                       dwAudioSample;
                        DWORD                       dwAudioEnc;
                        DWORD                       dwAudioEncSample;
                        char                        *pzServerType;
                        char                        *pzIPAddr;
                        char                        *pzUID;
                        char                        *pzPWD;
                        DWORD                       adwVCodecOrder[MAX_VIDEO_CODEC];
                        DWORD                       adwACodecOrder[MAX_AUDIO_CODEC];
                        DWORD                       dwFlags;
                        double                      dPlaySpeed;
                        TDataBrokerSBCallback       pfSB;
                        DWORD_PTR                   dwSBContext;
}
TDataBrokerConnectionOptions;
```

Members

wCam

Camera index.

zVSize

Vsize. (For 2K series only)

dwQuality

Quality value. (For 2K series only)

pfStatus

Pointer to a callback function which is used to report connection status.

pfAV

Pointer to a callback function which is used to receive audio and video frames.

pfTx

Pointer to a callback function which is used to transmit media packets.

dwStatusContext

An instance which is associated to the pfStatus callback function.

dwAVContext

An instance which is associated to the pfAV callback function.

dwTxContext

An instance which is associated to the pfTx callback function.

wHttpPort

HTTP port number.

dwProtocolType

Protocol type.

dwMediaType

Requested media type.

dwVideoCodec

Video codec type.

dwAudioCodec

Audio codec type.

dwAudioSample

Audio sample rate.

dwAudioEnc

Audio encoding codec type.

dwAudioEncSample

Audio encoding sample rate.

pzServerType

Server friendly name. Applications should set this parameter to either “Auto” or “Darwin” string.

pzIPAddr

Remote IP address, the maximum length is 128 bytes. Applications should set this parameter to a valid IP address.

pzUID

User login ID, the maximum length is 40 bytes. Application should set this parameter to a valid login ID.

pzPWD

User login password, the maximum length is 40 bytes. Application should set this parameter to a valid login password.

adwVCodecOrder

Server supports video codecs following this order. It’s only valid for 6K server.

adwACodecOrder

Server supports audio codecs following this order. It’s only valid for 6K server.

dwFlags

A combination of TDataBrokerConnectionOptionFlag to indicate one or more options that take effect.

dPlaySpeed

RTSP speed.

pfSB

Reserved. Pointer to a callback function which is used to input media packets.

dwSBContext

Resreved. An instance which is associated to the pfSB callback function.

Remarks

Applications should set pzServerType, pzIPAddr, pzUID, and pzPWD to specified values. Other options are optional. Applications can use dwFlags to indicate one or more options that take effect. Otherwise, they would use default values.

Applicatioins can pass status and AV callback function by calling DataBroker_Initial, DataBroker then uses the same callback functions among all connections. If you want to give different callback functions for each connection, set pfStatus and pfAV to specified function in TDataBrokerConnectionOptions when calling DataBroker_SetConnectionOptions.

For 3000 series models, the audio codec type could be one of the two modes: one is eACodecG7221, the other is eACodecG7221|eACodecG729A. Because client must always support this codec, eACodecG7221.

To use DataBroker to build connection with server in generic way, applications should set pzServerType to “Auto” and set pzIPAddr, pzUID, pzPWD, and wHttpPort.

Requirements

DataBroker.h

4.3.2 TDataBrokerConnInfo

Describes the information of a connection.

```
typedef struct {
                                        DWORD           dwWidth;
                                        DWORD           dwHeight;
                                        char            zLanguage[MAX_LANGUAGE_LEN + 1];
                                        DWORD           dwAudioCodec;
                                        DWORD           dwVideoCodec;
                                        DWORD           dwMediaType;
                                        DWORD           dwProtocol;
                                        WORD            wVideoPort;
                                        WORD            wAudioPort;
                                        char            szServerType[MAX_SERVERTYPE_LEN + 1];
                                        DWORD           dwMetadataType
} TDataBrokerConnInfo;
```

Members

dwWidth

The width of the image. Note that this value is only for reference because for HTTP mode of 3000 server, the value is not retrieved actually, so some reference value is returned. To get the exact value, please use the decoder callback from AVSynchronizer.

dwHeight

The height of the image. Note that this value is only for reference because for HTTP mode of 3000 server, the value is not retrieved actually, so some reference value is returned. To get the exact value, please use the decoder callback from AVSynchronizer.

zLanguage

The language type of the server.

dwAudioCodec

The audio codec type of the server.

dwVideoCodec

The video codec type of the server.

dwMediaType

The media type of the server.

dwProtocol

Current protocol type which is used by the connection.

wVideoPort

The server side video port which is used by the connection.

wAudioPort

The server side audio port which is used by the connection.

szServerType

The server's friendly name

dwMetadataType

The metadata type. The parameter can be mctMETX or mctMETJ which is defined in EMediaCodecType.

Remarks

The connection information would be carried through the status callback function when the status code is eConOptConnectionInfo.

Requirements

DataBroker.h, mediatypdef.h

4.3.3 TDataBrokerInputOptions

Describes the input options.

```
typedef struct TStruct1 {
                                        TDataBrokerStatusCallback    pfStatus;
                                        TDataBrokerAVCallback        pfAV;
                                        DWORD_PTR                    dwStatusContext;
                                        DWORD_PTR                    dwAVContext;
                                        char                         *pzServerType;
                                        DWORD                        dwAudioCodec;
                                        DWORD                        dwVideoCodec;
                                        DWORD                        dwProtocolType;
                                        char                         *zVSize;
} TDataBrokerInputOptions;
```

Members

pfStatus

Pointer to a callback function which is used to report connection status.

pfAV

Pointer to a callback function which is used to receive audio and video frames.

dwStatusContext

An instance which is associated to the pfStatus callback function.

dwAVContext

An instance which is associated to the pfAV callback function.

pzServerType

Server friendly name.

dwAudioCodec

Audio codec type.

dwVideoCodec

Video codec type.

dwProtocolType

Protocol type.

zVSize

Vsize. (For 2K series only)

Remarks

Requirements

DataBroker.h

4.3.4 TDataBrokerOptions

Describes the miscellaneous options.

```
typedef struct {
                        BOOL                        bEnableProxy;
                        DWORD                       dwProxyPort;
                        char                        szProxyName[MAX_PROXY_NAME_LEN + 1];
                        BOOL                        bEnableIPRestrict;
                        DWORD                       dwIPRestrictNum;
                        char                        *pszIPRestrictList[MAX_PROXY_NAME_LEN + 1];
                        DWORD                       dwFlags;
                        DWORD                       dwConnTimeout;
                        DWORD                       dwRWTimeout;
                        BOOL                        bCreateMonitorThread;
                        BOOL                        bCreateWokerThread;
                        TDataBrokerLogFunction      pLogFunction;
} TDataBrokerOptions;
```

Members

bEnableProxy

Enable proxy when connect to server by HTTP.

dwProxyPort

Proxy port.

szProxyName

IP of proxy server.

bEnableIPRestrict

Enable IP restriction check.

dwIPRestrictNum

The number of IP restriction strings you want to apply.

pszIPRestrictList

The pointer to the list of IP restriction strings. Proxy will not be applied to the IP starting by these strings.

dwFlags

A combination of TDataBrokerOptionFlag to indicate one or more options that take effect.

dwConnTimeout

This is the socket connection timeout in seconds. The default value is 20 seconds. If you are connecting to 6000 servers, the timeout value should not be less than 20 seconds.

dwRWTimeout

This is the socket read or write timeout in seconds. The default value is 30 seconds.

bCreateMonitorThread

Reserved. Set this parameter to FALSE if applications don't want DataBroker create an internal monitor thread. Otherwise, DataBroker will use an internal thread to monitor all its network connections.

bCreateWokerThread

Reserved. Set this parameter to FALSE if applications don't want DataBroker create I/O services to handle network connections.

pLogFunction

Reserved. Pointer to a TDataBrokerLogFunction function. Applications can use this parameter to get detail log messages from DataBroker.

Remarks

Requirements

DataBroker.h, DataBrokerCallbackDef.h

4.3.5 TDataBrokerRTSPPlayOptions

Describes the RTSP PLAY options.

```
typedef struct {
                                             const char          *pszRange;
                                             float               fScale;
                                             float               fSpeed;
} TDataBrokerRTSPPlayOptions;
```

Members

pszRange

Time period.

fScale

Scale. This parameter indicates the forward or backward play.

fSpeed

Speed.

Remarks

Requirements

DataBroker.h

4.3.6 TDataBrokerSocketFDInfo

Describes the socket info.

```
typedef struct {
                                        DWORD           dwSize;
                                        int             anSocketFD[6];
} TDataBrokerSocketFDInfo;
```

Members

dwSize

Number of returned sockets in anSocketFD array.

anSocketFD

An array of sockets.

Remarks

Requirements

DataBroker.h

4.3.7 TDataBrokerONVIFOptions

Describes the Onvif options.

```
typedef struct {
                                        DWORD           dwProfileIndex;
                                        DWORD           dwAudioCodec;
                                        DWORD           dwVideoCodec;
                                        char            *pzURI;
                                        DWORD           dwVideoSource;
} TDataBrokerONVIFOptions;
```

Members

dwProfileIndex

Profile index.

dwAudioCodec

Audio codec type.

dwVideoCodec

Video codec type.

pzURI

Onvif device’s URI.

dwVideoSource

Video source.

Remarks

Requirements

DataBroker.h

4.3.8 TDataBrokerSVCLayerInfo

Reserved. Describes the SVC layer info.

```
typedef struct {
                                    int             nDependency;
                                    int             nQuality;
                                    int             nTemporal;
} TDataBrokerSVCLayerInfo;
```

Members

nDependency

Spatial scalibility.

nQuality

Quality scalibility.

nTemporal

Temporal scalibility.

Remarks

Requirements

DataBroker.h

4.3.9 TDataBrokerH264Info

Reserved. Describes the H.264 SPS and PPS info.

```
typedef struct {
                    BYTE        *pbySPS;
                    DWORD       dwSPSLength;
                    BYTE        *pbyPPS;
                    DWORD       dwPPSLength
} TDataBrokerH264Info;
```

Members

pbySPS

Pointer to SPS data.

dwSPSLength

The size, in bytes, of the SPS data.

pbyPPS

Pointer PPS data.

dwPPSLength

The size, in bytes, of the PPS data.

Remarks

Requirements

DataBroker.h

4.3.10 TDataBrokerMJPEGInfo

Reserved. Describes the motion JPEG CI info.

```
typedef struct {
                              BYTE          *pbyCI;
                              DWORD         dwCILength;
} TDataBrokerMJPEGInfo;
```

Members

pbyCI

Pointer to CI data.

dwCILength

The size, in bytes, of the CI data.

Remarks

Requirements

DataBroker.h

4.3.11 TDataBrokerMPEG4Info

Reserved. Describes the MPEG4 CI info.

```
typedef struct {
                                   BYTE          *pbyCI;
                                   DWORD         dwCILength;
} TDataBrokerMPEG4Info;
```

Members

pbyCI

Pointer to CI data.

dwCILength

The size, in bytes, of the CI data.

Remarks

Requirements

DataBroker.h

4.3.12 TAACExtInfo

Deprecated. Describes the AAC info.

```
typedef struct {
                                        DWORD           dwSamplingFrequency;
                                        DWORD           dwChannelNumber;
} TAACExtInfo;
```

Members

dwSamplingFrequency

Sample frequency of AAC.

dwChannelNumber

Channel number of the AAC.

Remarks

Requirements

DataBroker.h

4.4 API Definition

The API is depicted here.

- DataBroker_CheckIfLive
- DataBroker_Connect
- DataBroker_CreateConnection
- DataBroker_CreateInput
- DataBroker_CreateInputEx
- DataBroker_DeleteConnection
- DataBroker_DeleteInput
- DataBroker_DeleteInputEx
- DataBroker_Disconnect
- DataBroker_ForceIFrame
- DataBroker_GetVersionInfo
- DataBroker_Initial
- DataBroker_InputPacket
- DataBroker_InputPacketEx
- DataBroker_InputTxPacket
- DataBroker_JumpMediaStreaming
- DataBroker_PauseMediaStreaming
- DataBroker_Release
- DataBroker_ResumeMediaStreaming
- DataBroker_SetCodecPriority
- DataBroker_SetConnectionExtraOption
- DataBroker_SetConnectionNetPacketCallback
- DataBroker_SetConnectionOptions

- DataBroker_SetConnectionUrlsExtra
- DataBroker_ChangeFrameRate
- DataBroker_SetInputExtraOption
- DataBroker_SetInputOptions
- DataBroker_SetOptions
- DataBroker_StartTxConnection
- DataBroker_StopTxConnection

4.4.1 DataBroker_CheckIfLive

Deprecated. Check if a RTSP connection is live or file playback.

```
SCODE DataBroker_CheckIfLive(
                                        HANDLE                    hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

The connection is live.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

DATABROKER_E_FAIL

The connection is not a live channel.

Requirements

DataBroker.h

4.4.2 DataBroker_Connect

Start a connection. The media packets will start to be carried through TDataBrokerAVCallback callback function.

```
SCODE DataBroker_Connect(
                                        HANDLE                      hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

Establish The connection to the remote server successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

DATABROKER_E_RUNNINGCONNECTION

The connection is running.

DATABROKER_E_NOSETTING

No setting about this Connection.

DATABROKER_E_FAIL

Fail to establish The connection to the remote server.

Remarks

Applications should call DataBroker_SetConnectionOptions at least once before calling this function.

Requirements

DataBroker.h

4.4.3 DataBroker_CreateConnection

Create a Connection instance.

```
SCODE DataBroker_CreateConnection(
                                        HANDLE                  hDataBroker,
                                        HANDLE                  *phConn
);
```

Parameters

hDataBroker

[in] Pointer to a handle of Databroker which is returned by DataBroker_Initial.

phConn

[out] Pointer to a handle of a connection.

Return Values

DATABROKER_S_OK

Create a connection instance successfully.

DATABROKER_E_INVALID_HANDLE

The DataBroker handle is invalid.

DATABROKER_E_OUT_OF_MEMORY

The module doesn’t have enough memory to create resources.

DATABROKER_E_TOO_MANY_CONNECTIONS

Can’t create any more connection.

DATABROKER_E_FAIL

Fail to create a Connection instance.

Requirements

DataBroker.h

4.4.4 DataBroker_CreateInput

Create an input instance. This function is obsolete applications should use DataBroker_CreateInputEx instead.

```
SCODE DataBroker_CreateInput(
                                        HANDLE                  *phInput,
                                        DWORD                   dwVersion
);
```

Parameters

phInput

[out] Pointer to a handle of input instance.

dwVersion

[in] The library version.

Return Values

DATABROKER_S_OK

Create an input instance successfully.

ERR_INVALID_VERSION

Library version is invalid.

DATABROKER_E_OUT_OF_MEMORY

The module doesn't have enough memory to create resources.

Requirements

DataBroker.h

4.4.5 DataBroker_CreateInputEx

Create an Input instance. This function is meant to replace DataBroker_CreateInput.

```
SCODE DataBroker_CreateInputEx(
                                        HANDLE                  hDataBroker,
                                        HANDLE                  *phInput
);
```

Parameters

hDataBroker

[in] Pointer to a handle of Databroker which is returned by DataBroker_Initial.

phInput

[out] Pointer to a handle of Input instance.

Return Values

DATABROKER_S_OK

Create an Input instance successfully.

ERR_INVALID_VERSION

Library version is invalid.

DATABROKER_E_OUT_OF_MEMORY

The module doesn't have enough memory to create resources.

DATABROKER_E_TOO_MANY_CONNECTIONS

Can't create any more input due to maximum connection limitation is reached. Tosolve this problem, it should be assigned larger connection number when initialize Databroker object.

Requirements

DataBroker.h

4.4.6 DataBroker_DeleteConnection

Delete a Connection instance.

```
SCODE DataBroker_DeleteConnection(
                                        HANDLE                  hDataBroker,
                                        HANDLE                  *phConn
);
```

Parameters

hDataBroker

[in] Pointer to a handle of Databroker which is returned by DataBroker_Initial.

phConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

Delete the connection instance successfully.

DATABROKER_E_INVALID_HANDLE

The DataBroker handle or Connection handle is invalid.

Remarks

A connection could be used to connect to different servers (of course, only connect to one server at the same moment) by setting different options. So it is more efficient to create all The connections when program starts and does not delete those connection until program ends.

Requirements

DataBroker.h

4.4.7 DataBroker_DeleteInput

Delete an input instance. This function is obsolete applications should use DataBroker_DeleteInputEx instead.

```
SCODE DataBroker_DeleteInput(
                                        HANDLE                  *phInput
);
```

Parameters

phInput

[in] Pointer to a handle of input instance which is returned by DataBroker_CreateInput.

Return Values

DATABROKER_S_OK

Delete the Input instance successfully.

DATABROKER_E_INVALID_HANDLE

The Input handle is invalid.

Requirements

DataBroker.h

4.4.8 DataBroker_DeleteInputEx

Delete an Input instance. This function is to repace DataBroker_DeleteInput.

```
SCODE DataBroker_DeleteInputEx(
                                        HANDLE                  hDataBroker,
                                        HANDLE                  *phInput
);
```

Parameters

hDataBroker

[in] Pointer to a handle of Databroker which is returned by DataBroker_Initial.

phInput

[in] Pointer to a handle of input instance which is returned by DataBroker_CreateInputEx.

Return Values

DATABROKER_S_OK

Delete the input instance successfully.

DATABROKER_E_INVALID_HANDLE

The input handle is invalid.

Requirements

DataBroker.h

4.4.9 DataBroker_Disconnect

Disconnect from remote server.

```
SCODE DataBroker_Disconnect(
                                        HANDLE                  hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

Disconnect from remote server successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Remarks

The connection is not yet closed after this function returned. Applications should wait the eOnStopped status being callback to ensure the connection is really closed.

Requirements

DataBroker.h

4.4.10 DataBroker_ForceIFrame

Force server to send an I-Frame immediately.

```
SCODE DataBroker_ForceIFrame (
                                        HANDLE                  hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

Force server to send a I-Frame successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Requirements

DataBroker.h

4.4.11 DataBroker_GetVersionInfo

Get the version information of the DataBroker library.

```
SCODE DataBroker_GetVersionInfo(
                                        BYTE                  *byMajor,
                                        BYTE                  *byMinor,
                                        BYTE                  *byBuild,
                                        BYTE                  *byRevision
);
```

Parameters

byMajor

[out] Pointer to the major version number.

byMinor

[out] Pointer to the minor version number.

byBuild

[out] Pointer to the build version number.

byRevision

[out] Pointer to the revision version number.

Return Values

DATABROKER_S_OK

Get the version information successfully.

Requirements

DataBroker.h

4.4.12 DataBroker_Initial

Create a DataBroker instance.

```
SCODE DataBroker_Initial(
                                        HANDLE                      *phDataBroker,
                                        DWORD                       dwMaxConn,
                                        TDataBrokerStatusCallback   pfStatus,
                                        TDataBrokerAVCallback       pfAV,
                                        DWORD                       dwSupportCodec
                                        DWORD                       dwFlag
                                        DWORD                       dwVersion
);
```

Parameters

phDataBroker

[out] Pointer to a handle of DataBroker object.

dwMaxConn

[in] The maximum number of connections that DataBroker can handle.

pfStatus

[in] Pointer to the status callback function. This parameter can be NULL.

pfAV

[in] Pointer to the AV callback function. This parameter can be NULL.

dwSupportCodec

[in] The codec types which client supports. For most case, this value is set to mctALLCODEC.

dwFlag

[in] Reserved.

dwVersion

[in] The library version.

Return Values

DATABROKER_S_OK

Create the DataBroker instance successfully.

ERR_INVALID_VERSION

Library version is invalid.

DATABROKER_E_OUT_OF_MEMORY

The module doesn’t have enough memory to create resources.

DATABROKER_E_FAIL

Fail to create the DataBroker instance.

Requirements

DataBroker.h

4.4.13 DataBroker_InputPacket

Input stream packet from network client to the media unpacketizer and parser. This function is obsolete please use DataBroker_InputPacketEx instead.

```
SCODE DataBroker_InputPacket(
                                        HANDLE              hInput,
                                        TsdrMediaType       dwMediaType,
                                        BYTE                *pbyData,
                                        DWORD               dwLength
);
```

Parameters

hInput

[in] Pointer to a handle of input instance which is returned by DataBroker_CreateInput.

dwMediaType

[in] The media type of the stream data.

pbyData

[in] The stream data received from network.

dwLength

[in] The length of the input stream data.

Return Values

DATABROKER_S_OK

Input stream data from network client to unpacketizer and parser successfully.

DATABROKER_E_OUT_OF_MEMORY

The module doesn’t have enough memory to create resources.

DATABROKER_E_FAIL

Fail to input stream data from network client to unpacketizer and parser.

Requirements

DataBroker.h

4.4.14 DataBroker_InputPacketEx

Input stream packet from network client to the media unpacketizer and parser. This function solves the problem for DataBroker_InputPacket that sometimes the frame could lose if the data size is small.

```
SCODE DataBroker_InputPacketEx(
                    HANDLE          hInput,
                    TsdrMediaType   dwMediaType,
                    BYTE            *pbyData,
                    DWORD           dwLength
);
```

Parameters

hInput

[in] Pointer to a handle of input instance which is returned by DataBroker_CreateInputEx.

dwMediaType

[in] The media type of the stream data.

pbyData

[in] The stream data received from network or from the TDataBrokerNetPacketCallback callback function.

dwLength

[in] The length of the input stream data.

Return Values

DATABROKER_S_OK

Input stream data from network client to unpacketizer and parser successfully.

DATABROKER_E_OUT_OF_MEMORY

The module doesn’t have enough memory to create resources.

DATABROKER_E_FAIL

Fail to input stream data from network client to unpacketizer and parser.

Requirements

DataBroker.h

4.4.15 DataBroker_InputTxPacket

Input upstream packet into network client and it will be sent to server side. This is only available for 6000/7000/8000 series servers.

```
SCODE DataBroker_InputTxPacket(
                                        HANDLE                  hConn,
                                        TMediaDataPacketInfo    *ptMediaPacketInfo,
                                        DWORD                   dwDataTimePeriod
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

ptMediaPacketInfo

[in] Pointer to a media packet. Currently, only audio packets encoded by G.729A or G.711 are allowed.

dwDataTimePeriod

[in] Time period, in millisecons, that the data in ptMediaPacketInfo needs when playing back. This time is also used by DataBroker to control the data rate sent.

Return Values

DATABROKER_S_OK

Input upstream packet successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is not correct.

Requirements

DataBroker.h

4.4.16 DataBroker_JumpMediaStreaming

Reserved. For RTSP connection and the connection is for a file playback, users can use this function to jump to arbitrary point in the period.

```
SCODE DataBroker_JumpMediaStreaming(
                                        HANDLE              hConn,
                                        DWORD               dwPercentage
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

dwPercentage

[in] This is the percentage to be set. The percentage is 10000 based. That means, when set 100, the module will move the play location to 100/10000 = 0.01 position.

Return Values

DATABROKER_S_OK

Operation successfully.

DATABROKER_E_INVALID_HANDLE

The cnnection handle is invalid.

Requirements

DataBroker.h

4.4.17 DataBroker_PauseMediaStreaming

For RTSP connection, pause the non-live connection. The server will not stream any media after pause. Users could use DataBroker_ResumeMediaStreaming to resume the streaming.

```
SCODE DataBroker_PauseMediaStreaming(
                                        HANDLE                    hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

The connection is paused.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Requirements

DataBroker.h

4.4.18 DataBroker_Release

Release a DataBroker instance.

```
SCODE DataBroker_Release(
                                        HANDLE                    *phDataBroker
);
```

Parameters

phDataBroker

[in/out] Pointer to a handle of Databroker which is returned by **DataBroker_Initial**.

Return Values

DATABROKER_S_OK

Release the DataBroker instance successfully.

DATABROKER_E_INVALID_HANDLE

The DataBroker handle is invalid.

Requirements

DataBroker.h

4.4.19 DataBroker_ResumeMediaStreaming

For RTSP connection, resume the paused non-live connection. The server will continue the streaming.

```
SCODE DataBroker_ResumeMediaStreaming(
                                        HANDLE                  hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

The connection resume successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Remarks

If the connection already timeout, the module will try to re-establish the connection and set the playing point to where paused.

Requirements

DataBroker.h

4.4.20 DataBroker_SetCodecPriority

Set the codec using priority for all connections.

```
SCODE DataBroker_SetCodecPriority(
                                        HANDLE          hDataBroker,
                                        DWORD           *pdwVideoCodec,
                                        DWORD           *pdwAudioCodec
);
```

Parameters

hDataBroker

[in] Pointer to a handle of Databroker which is returned by DataBroker_Initial.

pdwVideoCodec

[in] Pointer to an array which contains the priority order of video codecs.

pdwAudioCodec

[in] Pointer to an array which contains the priority order of audio codecs.

Return Values

DATABROKER_S_OK

Set options to a DataBroker instance successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Remarks

The priorities are only used for 6K servers..

Requirements

DataBroker.h

4.4.21 DataBroker_SetConnectionExtraOption

Set more option for a connection. These options can be set during the connection is active (connected).

```
SCODE DataBroker_SetConnectionExtraOption(
                                        HANDLE          hConn,
                                        DWORD           dwOption,
                                        DWORD           dwParam1,
                                        DWORD           dwParam2
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

dwOption

[in] The options which are listed in TExtraOptions.

dwParam1

[in] The first extra parameter for corresponding option. The meaning of this value is determined by dwOption parameter.

dwParam2

[in] The second extra parameter for corresponding option. The meaning of this value is determined by dwOption parameter.

Return Values

DATABROKER_S_OK

Set the connection options successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Requirements

DataBroker.h

4.4.22 DataBroker_SetConnectionNetPacketCallback

Set network packet callback function and the related parameters. These settings could be changed during connection, but it is recommended to set them before connecting.

```
SCODE
DataBroker_SetConnectionNetPacketCallback(    HANDLE                            hConn,
                                              TDataBrokerNetPacketCallback      pfNetPacketCallback,
                                              DWORD_PTR                         dwContext,
                                              BOOL                              bCallbackOnly
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

pfNetPacketCallback

[in] Pointer to a network packet callback function.

dwContext

[in] Data to be passed to callback function.

bCallbackOnly

[in] If this flag is set, the module would not call back the frame data. That is the TDataBrokerAVCallback would not get called. This would save some CPU resources if the computer runs as a proxy server.

Return Values

DATABROKER_S_OK

Set connection options successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Requirements

DataBroker.h

4.4.23 DataBroker_SetConnectionOptions

Set options of a connection.

```
SCODE
DataBroker_SetConnectionOptions(  HANDLE                              hConn,
                                  TDataBrokerConnectionOptions        *ptOption

);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

ptOption

[in] Pointer to a TDataBrokerConnectionOptions structure.

Returrn Values

DATABROKER_S_OK

Set the Connection options successfully.

DATABROKER_E_INVALID_HANDLE

The Connection handle is invalid.

DATABROKER_E_INVALID_ARG

The input argument is invalid.

DATABROKER_E_OUT_OF_MEMORY

The module doesn’t have enough memory to create resources.

DATABROKER_E_WRONG_CONNECTION_SETTING

Wrong settings.

DATABROKER_E_RUNNINGCONNECTION

The connection is running. You can't set a connection's options when it is running. Stop it before calling this function.

DATABROKER_E_FAIL

Failed to set the Connection options.

Remarks

The function will return DATABROKER_E_WRONG_CONNECTION_SETTING if the setting is not correct, the following list the possible reasons for this error code:

- szServerType is not specified
- For 2000 or 3000 video only model (VS3101) the protocol is not HTTP or the media includes audio.
- For 3000 series (A/V models) the protocol is HTTP but the media includes audio or the protocol is TCP or UDP but the media is not A/V.
- For 6000 servers, the protocol is TCP.
- For 7000 servers, the protocol is not TCP or not UDP.
- The szServerType contains model that is not listed in SrvDepResource module.

Requirements

DataBroker.h

4.4.24 DataBroker_SetConnectionUrlsExtra

Set extra URL information to a connection. The URLs will take effect only if it’s called before calling DataBroker_Connect.

```
SCODE DataBroker_SetConnectionUrlsExtra(
                                        HANDLE                  hConn,
                                        const char              *pszVideoUrl,
                                        const char              *pszVideoExtra,
                                        const char              *pszTxUrl,
                                        const char              *pszTxExtra,
                                        const char              *pszRxUrl,
                                        const char              *pszRxExtra
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

pszVideoUrl

[in] The actual video URL when connecting to the server. Usually it is not necessary to set the URL when connecting to the server. But if you are connecting to the IIS solution for 2000 server, the URL should be updated so that DataBroker could connect to it correctly. For 7000 servers or other RTSP streaming server, the URL could be different from those specified in SrvDepResource module. In such case, please use this function to set the video URL. The URL should not contain the IP/host name part, and should not contain the parameters part (put parameters in pszVideoExtra instead).

pszVideoExtra

[in] The extra parameters for the specific video URL.

pszTxUrl

[in] Reserved.

pszTxExtra

[in] Reserved.

pszRxUrl

[in] Reserved.

pszRxExtra

[in] Reserved.

Return Values

DATABROKER_S_OK

The connection resume successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

DATABROKER_E_INVALID_ARG

The input argument is invalid.

Requirements

DataBroker.h

4.4.25 DataBroker_ChangeFrameRate

Set the frame rate of the corresponding connection when connecting to SVC cameras.

```
SCODE DataBroker_ChangeFrameRate(
                                        HANDLE                  hConn,
                                        EFrameRateOption        eOption
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

eOption

[in] The parameter can be one of the EFrameRateOption enumerated type.

Return Values

DATABROKER_S_OK

The operation successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Requirements

DataBroker.h

4.4.26 DataBroker_SetInputExtraOption

Set more option of an input instance. The property could be changed at any time.

```
SCODE DataBroker_SetInputExtraOption(
                                        HANDLE          hInput,
                                        DWORD           dwOption,
                                        DWORD           dwParam1,
                                        DWORD           dwParam2
);
```

Parameters

dwOption

[in] The parameter can be one of the **TExtraOptions**.

dwParam1

[in] The first extra parameter for corresponding option. The meaning of this value is determined by dwOption parameter.

dwParam2

[in] The second extra parameter for corresponding option. The meaning of this value is determined by dwOption parameter.

Return Values

DATABROKER_S_OK

The operation successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is invalid.

Remarks

If The connection already timeout, the module will try to re-establish The connection and set the playing point to where paused.

Requirements

DataBroker.h

4.4.27 DataBroker_SetInputOptions

Reserved. Set options of an Input instance.

```
SCODE DataBroker_SetInputOptions(
                                        HANDLE                      hInput,
                                        TDataBrokerInputOptions     *ptInputOptions
);
```

Parameters

hInput

[in] Pointer to a handle of input instance which is returned by DataBroker_CreateInput or DataBroker_CreateInputEx.

ptInputOptions

[in] Pointer to a TDataBrokerInputOptions structure.

Return Values

DATABROKER_S_OK

Set options of an input instance successfully.

DATABROKER_E_INVALID_HANDLE

The input handle is invalid.

DATABROKER_E_INVALID_ARG

The argument is invalid.

Requirements

DataBroker.h

4.4.28 DataBroker_SetOptions

Set options to a DataBroker instance.

```
SCODE DataBroker_SetOptions(
                                        HANDLE                  hDataBroker,
                                        TDataBrokerOptions      *ptOptions
);
```

Parameters

hDataBroker

[in] Pointer to a handle of Databroker which is returned by DataBroker_Initial.

ptOptions

[in] Pointer to a TDataBrokerOptions structure.

Return Values

DATABROKER_S_OK

Set options to a DataBroker instance successfully.

DATABROKER_E_INVALID_ARG

The input argument is invalid.

DATABROKER_S_FAIL

Fail to set options to a DataBroker instance.

Remarks

You can set which servers should be applied proxy.

Note: If proxy is applied to a 3000 series video only server or a 3000 series AV server connected by HTTP, you can not get VSize and Language from callback function.

Requirements

DataBroker.h

4.4.29 DataBroker_StartTxConnection

Establish the upstream connection. This is only available for 6000/7000/8000 series servers. When the connection is established, the eOnTxChannelStart status code will be called back. If the channel has already been occupied, eOnAudioUpstreamOccupied will be called back. It might occurs when the server has already changed the audio mode, and the new audio mode does not support talk, in such case, the status eOnAudioUpstreamDisabled will be called.

```
SCODE DataBroker_StartTxConnection(
                                        HANDLE                    hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

Input upstream packet successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is not correct.

DATABROKER_E_FAIL

Unable to start the upstream connection. It might because the control connection to server is not established yet

Requirements

DataBroker.h

4.4.30 DataBroker_StopTxConnection

Close the upstream connection. This is only available for 6000/7000/8000 series servers. When the connection is closed, the eOnTxChannelClosedstatus code will be called back.

```
SCODE DataBroker_StopTxConnection(
                                        HANDLE                  hConn
);
```

Parameters

hConn

[in] Pointer to a handle of channel which is returned by DataBroker_CreateConnection.

Return Values

DATABROKER_S_OK

Input upstream packet successfully.

DATABROKER_E_INVALID_HANDLE

The connection handle is not correct.

Requirements

DataBroker.h