RTSP Streaming Server Reference Manual

Thu Dec 13 11:55:24 2007

Contents

1 RTSP Streaming Server Data Structure Index

1.1 RTSP Streaming Server Data Structures

Here are the data structures with brief descriptions:

2 RTSP Streaming Server File Index

2.1 RTSP Streaming Server File List

Here is a list of all documented files with brief descriptions:

3 RTSP Streaming Server Data Structure Documentation

3.1 TRTSPSTREAMING_AUDENCODING_PARAM Struct Reference

```
#include <rtspstreamingserver.h>
```

Data Fields

- int **iBitRate**
- int **iClockRate**
- int **iPacketTime**
- int **iOctetAlign**
- int **iAMRcrc**
- int **iRobustSorting**
- int **iAudioCodecType**
- int **iM4AProfileLevel**
- char **acM4ASpecConf** [16]
- int **iM4ASpecConfLen**
- int **iChanNum**
- char **acTrackName** [RTSPSTREAMING_TRACK_NAME_LEN]
- int **iCIReady**

3.1.1 Detailed Description

The data structure describes the parameters of AMR audio encoder in RTSPStreaming object. Used in **RTSPStreaming_SetAudioParameters** function to set the AMR audi encoder parameters.

The documentation for this struct was generated from the following file:

- rtspstreamingserver.h

3.2 TRTSPSTREAMING_VIDENCODING_PARAM Struct Reference

```
#include <rtspstreamingserver.h>
```

Data Fields

- int **iProfileLevel**
- int **iBitRate**
- int **iClockRate**
- char **acMPEG4Header** [50]
- int **iMPEG4HeaderLen**
- int **iWidth**
- int **iHeight**
- int **iDecoderBufferSize**
- char **acTrackName** [RTSPSTREAMING_TRACK_NAME_LEN]
- int **iCIReady**

3.2.1 Detailed Description

This structure maintains current parameters for video stream

The documentation for this struct was generated from the following file:

- rtspstreamingserver.h

3.3 TRTSPSTREAMING_DYNAMIC_PARAM Struct Reference

```
#include <rtspstreamingserver.h>
```

Data Fields

- char acAccessName [MULTIPLE_STREAM_NUM][ACCESSNAME_LENGTH]
- int iRTSPStreamingMediaType [MULTIPLE_STREAM_NUM]
- int iRTSPAuthentication

3.3.1 Detailed Description

This data structure states the parameters that can be modified at runtime

3.3.2 Field Documentation

char TRTSPSTREAMING_DYNAMIC_PARAM::acAccessName[MULTIPLE_STREAM_NUM][ACCESSNAME_LENGTH]

The access name of RTSP server for AV media. Maximun 20 characters .

int TRTSPSTREAMING_DYNAMIC_PARAM::iRTSPAuthentication

Authentication method of RTSP server

int TRTSPSTREAMING_DYNAMIC_PARAM::iRTSPStreamingMediaType[MULTIPLE_STREAM_NUM]

RTSP streaming server type. Now, not support video or audio only mode.

- RTSPSTREAMING_MEDIATYPE_AUDIOVIDEO => Support audio & video streaming.
- RTSPSTREAMING_MEDIATYPE_AUDIOONLY => Just support audio streaming.
- RTSPSTREAMING_MEDIATYPE_VIDEOONLY => Just support video streaming.

The documentation for this struct was generated from the following file:

- rtspstreamingserver.h

3.4 TRTSPSTREAMING_PARAM Struct Reference

```
#include <rtspstreamingserver.h>
```

Data Fields

- unsigned short usRTSPPort
- unsigned short usRTPVPort
- unsigned short usRTPAPort
- unsigned short usRTCPVPort
- unsigned short usRTCPAPort
- unsigned long ulLocalIP
- unsigned long ulLocalSubnetMask
- unsigned long ulNATIP
- char acAccessName [MULTIPLE_STREAM_NUM][ACCESSNAME_LENGTH]
- int iRTSPStreamingMediaType [MULTIPLE_STREAM_NUM]
- int iRTSPAuthentication
- DWORD dwRTSPServerPriority
- DWORD dwVideoChannelPriority
- DWORD dwAudioChannelPriority
- DWORD dwVideoPacketizerPriority
- DWORD dwAudioPacketizerPriority
- int iRTSPMaxConnectionNum

3.4.1 Detailed Description

The data structure describes the parameters of RTSPStreaming object. Used in **RTSPStreaming_Create** and **RTSPStreaming_SetParameters** functions to set the object parameters.

3.4.2 Field Documentation

char TRTSPSTREAMING_PARAM::acAccessName[MULTIPLE_STREAM_NUM][ACCESSNAME_LENGTH]

The access name of RTSP server for AV media. Maximun 20 characters .

DWORD TRTSPSTREAMING_PARAM::dwAudioChannelPriority

Audio Media Channel Thread Priority

DWORD TRTSPSTREAMING_PARAM::dwAudioPacketizerPriority

Audio Packetizer Thread Priority

DWORD TRTSPSTREAMING_PARAM::dwRTSPServerPriority

RTSP Signaling Server Thread Priority

DWORD TRTSPSTREAMING_PARAM::dwVideoChannelPriority

Video Media Channel Thread Priority

DWORD TRTSPSTREAMING_PARAM::dwVideoPacketizerPriority

Video Packetizer Thread Priority

int TRTSPSTREAMING_PARAM::iRTSPAuthentication

Authentication method of RTSP server

int TRTSPSTREAMING_PARAM::iRTSPMaxConnectionNum

Maximum number of allowed connections

int TRTSPSTREAMING_PARAM::iRTSPStreamingMediaType[MULTIPLE_STREAM_NUM]

RTSP streaming server type. Now, not support video or audio only mode.

- RTSPSTREAMING_MEDIATYPE_AUDIOVIDEO => Support audio & video streaming.
- RTSPSTREAMING_MEDIATYPE_AUDIOONLY => Just support audio streaming.
- RTSPSTREAMING_MEDIATYPE_VIDEOONLY => Just support video streaming.

unsigned long TRTSPSTREAMING_PARAM::ulLocalIP

Local IP address in host order.

unsigned long TRTSPSTREAMING_PARAM::ulLocalSubnetMask

Local subnet mask in host order.

unsigned long TRTSPSTREAMING_PARAM::ulNATIP

Public IP address in host order when server inside NAT environment.

unsigned short TRTSPSTREAMING_PARAM::usRTCPAPort

fixed RTCP port for audio

unsigned short TRTSPSTREAMING_PARAM::usRTCPVPort

fixed RTCP port for video

unsigned short TRTSPSTREAMING_PARAM::usRTPAPort

fixed RTP port for audio

unsigned short TRTSPSTREAMING_PARAM::usRTPVPort

fixed RTP port for video

unsigned short TRTSPSTREAMING_PARAM::usRTSPPort

RTSP streaming server port number in host order.

The documentation for this struct was generated from the following file:

- rtspstreamingserver.h

3.5 RTSPSERVER_SESSIONINFORMATION Struct Reference

```
#include <rtspserver.h>
```

Data Fields

- DWORD dwSessionID
- SOCKET sktRTP [2]
- SOCKET sktRTCP [2]
- DWORD dwInitialTimeStamp [2]
- DWORD dwSSRC [2]
- WORD wInitialSequenceNumber [2]
- char cMediaName [2][25]
- unsigned long ulClientIP
- int iRTCPTimeOut
- int iRTPStreamingType
- int iEmbeddedRTPID [2]
- int **iEmbeddedRTCPID** [2]
- SOCKET sktRTSPSocket
- int iVivotekClient
- int iSDPIndex
- sockaddr_in NATRTPAddr [2]
- sockaddr_in NATRTCPAddr [2]

3.5.1 Detailed Description

a structure that store the parameter of session information which is delivered to system control by callback function after PLAY message is done. (Except Session ID and client IP address, All the other session informations are in pairs for video and audio respectively.

3.5.2 Field Documentation

char RTSPSERVER_SESSIONINFORMATION::cMediaName[2][25]

Request media name of each session

DWORD RTSPSERVER_SESSIONINFORMATION::dwInitialTimeStamp[2]

TimeStamp of the first media packet

DWORD RTSPSERVER_SESSIONINFORMATION::dwSessionID

Session ID of 32 bit long number

DWORD RTSPSERVER_SESSIONINFORMATION::dwSSRC[2]

SSRC of each session of the server

int RTSPSERVER_SESSIONINFORMATION::iEmbeddedRTPID[2]

RTP/RTCP channel ID

int RTSPSERVER_SESSIONINFORMATION::iRTCPTimeOut

Timeout value for RTCP receiver report

int RTSPSERVER_SESSIONINFORMATION::iRTPStreamingType

RTP packet streaming type

int RTSPSERVER_SESSIONINFORMATION::iSDPIndex

SDP Index

int RTSPSERVER_SESSIONINFORMATION::iVivotekClient

1 means Vivotek client

struct sockaddr_in RTSPSERVER_SESSIONINFORMATION::NATRTCPAddr[2]

Address for RTCP in NAT

struct sockaddr_in RTSPSERVER_SESSIONINFORMATION::NATRTPAddr[2]

Address for RTP in NAT

SOCKET RTSPSERVER_SESSIONINFORMATION::sktRTCP[2]

Client UDP socket which is connected and ready to exchange RTCP sender and receiver report.

SOCKET RTSPSERVER_SESSIONINFORMATION::sktRTP[2]

Client UDP socket which is connected and ready to receive RTP media data.

SOCKET RTSPSERVER_SESSIONINFORMATION::sktRTSPSocket

Embedded RTSP socket

unsigned long RTSPSERVER_SESSIONINFORMATION::ulClientIP

Client IP address in network byte order

WORD RTSPSERVER_SESSIONINFORMATION::wInitialSequenceNumber[2]

Sequence number of the first media packet

The documentation for this struct was generated from the following file:

- rtspserver.h

3.6 THTTPServer_AcceptData Struct Reference

```
#include <httpserver.h>
```

Data Fields

- DWORD **dwClientID**
- DWORD **dwClientIPAddress**
- USHORT **usClientPort**
- DWORD **dwErrorCode**

3.6.1 Detailed Description

callback on accept data structure

The documentation for this struct was generated from the following file:

- httpserver.h

3.7 TBitstreamBuffer Struct Reference

a data structure about media bitstream buffer.

```
#include <bitstreambufdef.h>
```

Data Fields

- BYTE ∗ pbyBuffer

  pointer of buffer storing bitstream.

- DWORD dwBufSize

  total size of bistream buffer.

- DWORD dwBytesUsed

  a length of used media data size.

- TMediaDBFrameType tFrameType

  the frame type in the bitstream buffer.

- DWORD **dwSecond**
- DWORD **dwMilliSecond**
- DWORD **dwStreamType**

- DWORD ∗ **pdwPacketSize**
- DWORD **dwOffset**
- DWORD **dwIntelligentVideoLength**
- BOOL **bChangeSetting**
- DWORD **dwStreamIndex**
- DWORD **dwIsBoundary**

3.7.1 Detailed Description

a data structure about media bitstream buffer.

The documentation for this struct was generated from the following file:

- bitstreambufdef.h

3.8 THTTPServer_Settings Struct Reference

```
#include <httpserver.h>
```

Data Fields

- DWORD dwSettingsFlag
- int iThreadPriority
- BOOL bKeepAlive
- int iAuthorizationType
- DWORD dwConnectionTimeout

3.8.1 Detailed Description

Settings data structure for **HTTPServer_SetParameters**

3.8.2 Field Documentation

BOOL THTTPServer_Settings::bKeepAlive

connection keep-alive or close selection (0 for close, 1 for keep-alive), default: unchanged

DWORD THTTPServer_Settings::dwConnectionTimeout

set connection timeout value in millisecond(s), default: unchanged

DWORD THTTPServer_Settings::dwSettingsFlag

combinations of settings flags to choose which field is valid in the structure

int THTTPServer_Settings::iAuthorizationType

authorization or not selection (0: No, 1: Yes), default: unchanged

int THTTPServer_Settings::iThreadPriority

set the service thread's priority, default: unchanged

The documentation for this struct was generated from the following file:

- httpserver.h

3.9 THTTPServer_InitSettings Struct Reference

```
#include <httpserver.h>
```

Data Fields

- DWORD dwVersion
- DWORD dwInitSettingsFlag
- DWORD dwReceiveBufferSize
- CHAR acServerIPAddress [MAX_IPADDR_LEN]
- USHORT usPort
- ULONG ulMaxConnections
- DWORD dwStackSize
- int iThreadPriority
- BOOL bKeepAlive
- int iAuthorizationType
- DWORD dwConnectionTimeout
- char * pcHostName
- char * pcLanguage

3.9.1 Detailed Description

Initial settings data structure for **HTTPServer_Initial**

3.9.2 Field Documentation

CHAR THTTPServer_InitSettings::acServerIPAddress[MAX_IPADDR_LEN]

set server IP address in dotted decimal notation (default: bind with INADDR_ANY)

BOOL THTTPServer_InitSettings::bKeepAlive

connection keep-alive or close selection (0 for close, 1 for keep-alive), default is 1

DWORD THTTPServer_InitSettings::dwConnectionTimeout

set connection timeout value in millisecond(s), default is 900000 (15 minutes)

DWORD THTTPServer_InitSettings::dwInitSettingsFlag

combinations of settings flags to choose which field is valid in the structure

DWORD THTTPServer_InitSettings::dwReceiveBufferSize

set each receive buffer's size, default is 1460 bytes (Ethernet MTU)

DWORD THTTPServer_InitSettings::dwStackSize

set the service thread's stack size

DWORD THTTPServer_InitSettings::dwVersion

set this value to **HTTPServer_VERSION**

int THTTPServer_InitSettings::iAuthorizationType

authorization or not selection (0: No, 1: Yes), default is 0

int THTTPServer_InitSettings::iThreadPriority

set the service thread's priority, default: THREAD_PRIORITY_NORMAL under Windows, 150 under pSOS

char∗ THTTPServer_InitSettings::pcHostName

the host name, show in server header and realm, the maximum length is MAX_HOST_LEN

char∗ THTTPServer_InitSettings::pcLanguage

the language type of webpage, default is en, the maximum length is 15

ULONG THTTPServer_InitSettings::ulMaxConnections

set maximum number of connections concurrently, default is 10

USHORT THTTPServer_InitSettings::usPort

set local port (in host byte order), default is 80

The documentation for this struct was generated from the following file:

- httpserver.h

3.10 THTTPServer_ClientSettings Struct Reference

```
#include <httpserver.h>
```

Data Fields

- DWORD dwClientSettingsFlag
- BOOL bKeepAlive
- int iAuthorizationType
- ULONG ulConnectionTimeout

3.10.1 Detailed Description

Client settings data structure for **HTTPServer_SetClientParameters**

3.10.2 Field Documentation

BOOL THTTPServer_ClientSettings::bKeepAlive

connection keep-alive or close selection (0 for close, 1 for keep-alive), default: unchanged

DWORD THTTPServer_ClientSettings::dwClientSettingsFlag

combinations of settings flags to choose which field is valid in the structure

int THTTPServer_ClientSettings::iAuthorizationType

authorization or not selection (0: No, 1: Yes), default: unchanged

ULONG THTTPServer_ClientSettings::ulConnectionTimeout

set connection timeout value in millisecond(s), default: unchanged

The documentation for this struct was generated from the following file:

- httpserver.h

3.11 THTTPServer_Disconnect Struct Reference

```
#include <httpserver.h>
```

Data Fields

- DWORD dwClientID
- BOOL bTimedOut
- DWORD dwErrorCode

3.11.1 Detailed Description

callback on disconnect data structure

3.11.2 Field Documentation

BOOL THTTPServer_Disconnect::bTimedOut

indicate if the disconnect is caused by timed out

DWORD THTTPServer_Disconnect::dwClientID

client ID

DWORD THTTPServer_Disconnect::dwErrorCode

error code, reserved

The documentation for this struct was generated from the following file:

- httpserver.h

3.12 RTSPSERVER_SDPREQUEST Struct Reference

```
#include <rtspserver.h>
```

Data Fields

- unsigned long ulIP
- unsigned short usPort
- char ∗ pcDescribe
- char ∗ pSDPBuffer
- int iSDPBuffLen
- int iSDPindex
- int iVivotekClient

3.12.1 Detailed Description

a structure that store the parameter of Describe information which is delivered to system control by callback function

3.12.2 Field Documentation

int RTSPSERVER_SDPREQUEST::iSDPBuffLen

length of SDP buffer

int RTSPSERVER_SDPREQUEST::iSDPindex

index of SDP file

int RTSPSERVER_SDPREQUEST::iVivotekClient

1 means Vivotek client

char* RTSPSERVER_SDPREQUEST::pcDescribe

describe file name (*.sdp)

char* RTSPSERVER_SDPREQUEST::pSDPBuffer

pointer to the buffer to store SDP file

unsigned long RTSPSERVER_SDPREQUEST::ulIP

Parameter of client IP which is 32bit unsigned long of network byte order

unsigned short RTSPSERVER_SDPREQUEST::usPort

Parameter of client port number which is 16bit unsigned integer of network byte order

The documentation for this struct was generated from the following file:

- rtspserver.h

3.13 RTSPSERVER_PARAM Struct Reference

```
#include <rtspserver.h>
```

Data Fields

- unsigned short rtsp_port
- unsigned short usRTPVPort
- unsigned short usRTPAPort
- unsigned short usRTCPVPort
- unsigned short usRTCPAPort
- int iUDPRTPVSock
- int iUDPRTPASock
- unsigned long ulThreadPriority
- unsigned long ulIP

3.13.1 Detailed Description

a structure that store the parameter of RTSP sever

3.13.2 Field Documentation

int RTSPSERVER_PARAM::iUDPRTPASock

fixed UDP audio socket for Symmetric RTP

int RTSPSERVER_PARAM::iUDPRTPVSock

fixed UDP video socket for Symmetric RTP

unsigned short RTSPSERVER_PARAM::rtsp_port

Parameter that define the port number of RTSP server

unsigned long RTSPSERVER_PARAM::ulIP

IP address of RTSP Server

unsigned long RTSPSERVER_PARAM::ulThreadPriority

Thread Priority for RTSP Server

unsigned short RTSPSERVER_PARAM::usRTCPAPort

port number of RTCP audio port

unsigned short RTSPSERVER_PARAM::usRTCPVPort

port number of RTCP video port

unsigned short RTSPSERVER_PARAM::usRTPAPort

port number of RTP auido port

unsigned short RTSPSERVER_PARAM::usRTPVPort

port number of RTP video port

The documentation for this struct was generated from the following file:

- rtspserver.h

3.14 rtsps_initial_options Struct Reference

```
#include <rtsps.h>
```

Data Fields

- DWORD dwVersion
- DWORD dwRTSPPort
- const char szIPAddr [20]
- const char szSubnetMask [20]
- DWORD dwStreamNumber
- char szAccessName [MULTIPLE_STREAM_NUM][255]

3.14.1 Detailed Description

RTSPS initial options

3.14.2 Field Documentation

DWORD rtsps_initial_options::dwRTSPPort

RTSP streaming server port, default : 554

DWORD rtsps_initial_options::dwStreamNumber

RTSP streaming server stream number

DWORD rtsps_initial_options::dwVersion

This is a version control parameter. Set this value as **RTSPS_VERSION**.

char rtsps_initial_options::szAccessName[MULTIPLE_STREAM_NUM][255]

RTSP streaming server access names

const char rtsps_initial_options::szIPAddr[20]

RTSP streaming server ip address

const char rtsps_initial_options::szSubnetMask[20]

RTSP streaming server subnet mask

The documentation for this struct was generated from the following file:

- rtsps.h

3.15 stRTPPACKETIZERPARAM Struct Reference

```
#include <rtppacketizer.h>
```

Data Fields

- HANDLE hRTPMediaDataQueue
- HANDLE hRTPMediaEmptyQueue
- DWORD dwThreadPriority

3.15.1 Detailed Description

Structure for Packetizer Initialization parameters

3.15.2 Field Documentation

DWORD stRTPPACKETIZERPARAM::dwThreadPriority

Thread Priority for Packetizer

HANDLE stRTPPACKETIZERPARAM::hRTPMediaDataQueue

Data Queue Handle

HANDLE stRTPPACKETIZERPARAM::hRTPMediaEmptyQueue

Empty Queue Handle

The documentation for this struct was generated from the following file:

- rtppacketizer.h

3.16 RTSPSERVER_CLIENTIP Struct Reference

```
#include <rtspserver.h>
```

Data Fields

- unsigned long ulIP
- unsigned short usPort

3.16.1 Detailed Description

a structure that store the parameter of client socket information which is delivered to system control by callback function

3.16.2 Field Documentation

unsigned long RTSPSERVER_CLIENTIP::ulIP

Parameter of client IP which is 32bit unsigned long of network byte order

unsigned short RTSPSERVER_CLIENTIP::usPort

Parameter of client port number which is 16bit unsigned integer of network byte order

The documentation for this struct was generated from the following file:

- rtspserver.h

3.17 THTTPCONNINFO Struct Reference

```
#include <rtspserver.h>
```

Data Fields

- int iSendSock
- int iRecvSock
- char acSessionCookie [RTSP_HTTP_COOKIE_LEN]
- char acMessageBuffer [RTSP_HTTP_MESSAGE_LEN]

3.17.1 Detailed Description

a structure that store the parameters of HTTP socket for RTSP over HTTP mode

3.17.2 Field Documentation

char THTTPCONNINFO::acMessageBuffer[RTSP_HTTP_MESSAGE_LEN]

RTSP Message buffer

char THTTPCONNINFO::acSessionCookie[RTSP_HTTP_COOKIE_LEN]

Session cookie

int THTTPCONNINFO::iRecvSock

Socket for Receive RTSP over HTTP

int THTTPCONNINFO::iSendSock

Socket for Send RTSP over HTTP

The documentation for this struct was generated from the following file:

- rtspserver.h

3.18 RTPRTCPCHANNEL_CONNECTION Struct Reference

```
#include <mediachannel.h>
```

Data Fields

- DWORD dwSessionID
- SOCKET sktRTP
- SOCKET sktRTCP
- HANDLE hRTPRTCPComposerHandle
- int iRTPStreamingType
- int iEmbeddedRTPID
- int iEmbeddedRTCPID
- SOCKET sktRTSPSocket
- int iVivotekClient
- int iCodecIndex
- sockaddr_in RTPNATAddr
- sockaddr_in RTCPNATAddr

3.18.1 Detailed Description

This structure detail the session information for each connection

3.18.2 Field Documentation

DWORD RTPRTCPCHANNEL_CONNECTION::dwSessionID

Session ID

HANDLE RTPRTCPCHANNEL_CONNECTION::hRTPRTCPComposerHandle

RTPRTCP composer handle

int RTPRTCPCHANNEL_CONNECTION::iCodecIndex

Codec Index

int RTPRTCPCHANNEL_CONNECTION::iEmbeddedRTCPID

RTCP ID

int RTPRTCPCHANNEL_CONNECTION::iEmbeddedRTPID

RTP ID

int RTPRTCPCHANNEL_CONNECTION::iRTPStreamingType

RTP-over-UDP or RTP-over-TCP or RTP-over-HTTP

int RTPRTCPCHANNEL_CONNECTION::iVivotekClient

1 means vivotek client

struct sockaddr_in RTPRTCPCHANNEL_CONNECTION::RTCPNATAddr

Address for client when Server in NAT

struct sockaddr_in RTPRTCPCHANNEL_CONNECTION::RTPNATAddr

Address for client RTP when Server in NAT

SOCKET RTPRTCPCHANNEL_CONNECTION::sktRTCP

socket to send RTCP

SOCKET RTPRTCPCHANNEL_CONNECTION::sktRTP

socket to send RTP

SOCKET RTPRTCPCHANNEL_CONNECTION::sktRTSPSocket

RTSP socket for RTP-over-TCP mode

The documentation for this struct was generated from the following file:

- mediachannel.h

3.19 RTPRTCPCHANNEL_PARAM Struct Reference

```
#include <mediachannel.h>
```

Data Fields

- int iRTPRTCPMediaType
- BYTE ∗ pbyMPEG4StartBitStream
- int iMPEG4StartBitStreamLength
- int iRTCPTimeOut
- unsigned long ulThreadPriority
- int iUDPRTPSock
- int iUDPRTCPSock

3.19.1 Detailed Description

This is the RTPRTCP intial parameters

3.19.2 Field Documentation

int RTPRTCPCHANNEL_PARAM::iMPEG4StartBitStreamLength

length of bit stream

int RTPRTCPCHANNEL_PARAM::iRTCPTimeOut

timeout value of RTCP

int RTPRTCPCHANNEL_PARAM::iRTPRTCPMediaType

Media Channel Type (Must be audio or video)

int RTPRTCPCHANNEL_PARAM::iUDPRTCPSock

UDP mode RTSP server behind NAT sockets /Socket for RTCP

int RTPRTCPCHANNEL_PARAM::iUDPRTPSock

UDP mode RTSP server behind NAT sockets /Socket for RTP

BYTE∗ RTPRTCPCHANNEL_PARAM::pbyMPEG4StartBitStream

Pointer to the head of bit stream

unsigned long RTPRTCPCHANNEL_PARAM::ulThreadPriority

Priority of media channel thread

The documentation for this struct was generated from the following file:

- mediachannel.h

4 RTSP Streaming Server File Documentation

4.1 bitstreambufdef.h File Reference

header file of bitstream buffer structure.

```
#include "typedef.h"
#include "mediatypedef.h"
```

Data Structures

- struct TBitstreamBuffer

  a data structure about media bitstream buffer.

4.1.1 Detailed Description

header file of bitstream buffer structure.

Copyright 2000-2003 Vivotek, Inc. All rights reserved.

bitstreambufdef.h

Date:
2003/04/29

Author:
May Hsu

4.2 hs_parse.c File Reference

HTTP Server Module API.

```
#include "httpserver_local.h"
```

Functions

- void **ConvertHeaderToLowerCase** (PTParsingControl ptParsing)
- void **ConvertTokenToLowerCase** (char ∗pszToken, UINT uiTokenLength)
- UINT **FindLineEnd** (char ∗pszStartOfToken)
- UINT **FindTokenEnd** (char ∗pszStartOfToken)
- char ∗ **FindTokenStart** (char ∗pszBeginLine)
- DWORD **ParseDate** (char ∗pszDateString)
- TLineState **GetLineFromBuffer** (PTParsingControl ptParsing)
- char ∗ **FindValueStart** (char ∗pszValue)
- UINT **FindValueLength** (char ∗pszBeginLine)
- UINT **FindTokenDelimited** (char ∗pszStartOfToken, char cDelimiter)
- char ∗ **FindTokenDelimitedPtr** (char ∗pszStartOfToken, char cDelimiter)
- int **ParseAuthorDigestInfo** (char ∗pcMsgBuffer, THTTPRAWAUTHORINFO ∗ptHttpRawAuthInfo)

Variables

- const char ∗ **gMonthTable** [ ]

4.2.1 Detailed Description

HTTP Server Module API.

Copyright (C) 2003 Vivotek, Inc. All rights reserved.

parse.c

Date:
2003/05/26

Author:
Jason Yang

4.2.2 Variable Documentation

const char∗ gMonthTable[ ]

Initial value:

```
 { "Jan", "Feb", "Mar", "Apr", "May", "Jun",
                              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" }
```

4.3 httpserver.c File Reference

HTTP Server Module API.

```
#include "osisolate.h"
#include "common.h"
#include "httpserver.h"
#include "httpserver_local.h"
#include "hs_file.h"
```

Functions

- SCODE **HTTPServer_GetVersion** (BYTE ∗byMajor, BYTE ∗byMinor, BYTE ∗byBuild, BYTE ∗byRevision)
- SCODE HTTPServer_Initial (HANDLE ∗phHTTPServerObj, THTTPServer_InitSettings ∗pstInitSettings)

  Initialize a HTTPServer instance.

- SCODE HTTPServer_SetParameters (HANDLE hHTTPServerObj, THTTPServer_Settings ∗pstSettings)

  Set a HTTPServer instance's parameters.

- SCODE HTTPServer_Start (HANDLE hHTTPServerObj)

  Start a HTTPServer instance's service.

- SCODE HTTPServer_Stop (HANDLE hHTTPServerObj)

  Stop a HTTPServer instance.

- SCODE HTTPServer_Release (HANDLE ∗phHTTPServerObj)

  Release a HTTPServer instance.

- SCODE HTTPServer_SetCallback (HANDLE hHTTPServerObj, HTTPServer_Callback pfnCallback, DWORD dwInstance)

  set a HTTPServer instance's callback function.

- SCODE HTTPServer_Disconnect (HANDLE hHTTPServerObj, DWORD dwClientID)

  Disconnect a HTTPServer instance's client.

- SCODE **HTTPServer_SendData** (HANDLE hHTTPServerObj, DWORD dwClientID, PCHAR pSendBuffer, DWORD dwSendLength, BOOL bLastData, DWORD ∗pdwByteSent)
- SCODE HTTPServer_TakeClientOut (HANDLE hHTTPServerObj, DWORD dwClientID, SOCKET ∗psSocket)

  Set a HTTPServer instance's individual client's parameters Get a HTTPServer instance's client's socket descriptor and take the client out.

4.3.1 Detailed Description

HTTP Server Module API.

Copyright (C) 2003 Vivotek, Inc. All rights reserved.

HTTPServer.c

Date:
2003/05/26

Author:
Jason Yang

4.3.2 Function Documentation

SCODE HTTPServer_Disconnect (HANDLE *hHTTPServerObj*, DWORD *dwClientID*)

Disconnect a HTTPServer instance's client.

Parameters:

hHTTPServerObject (i) the HTTPServer instance's handle

dwClientID (i) the client id

Return values:

S_OK disconnect request queued successfully.

others queue disconnect request failed. check error codes.

Note:

It is a non-blocking function.

SCODE HTTPServer_Initial (HANDLE * *phHTTPServerObj*, THTTPServer_InitSettings * *pstInitSettings*)

Initialize a HTTPServer instance.

Parameters:

phHTTPServerObject (i/o) handle pointer to receive the created instance

pstInitialSettings (i) pointer to a **THTTPServer_InitialSettings** structure variable storing initial settings to create the HTTPServer instance

Return values:

S_OK HTTPServer instance created successfully.

others HTTPServer instance creation failed. check error codes.

See also:

HTTPServer_Release

SCODE HTTPServer_Release (HANDLE * *phHTTPServerObj*)

Release a HTTPServer instance.

Parameters:

phHTTPServerObject (i) handle pointer pointing to a HTTPServer instance to release

Return values:

S_OK HTTPServer instance released successfully.

others release HTTPServer instance failed. check error codes.

See also:

HTTPServer_Initial

SCODE HTTPServer_SetCallback (HANDLE *hHTTPServerObj*, HTTPServer_Callback *pfnCallback*, DWORD *dwInstance*)

set a HTTPServer instance's callback function.

Parameters:

hHTTPServerObject *(i)* handle of the HTTPServer instance to set callback

pfnCallback *(i)* the callback function

dwInstance *(i)* the parent instance

Return values:

S_OK callback function set successfully.

others set callback function failed. check error codes.

SCODE HTTPServer_SetParameters (HANDLE *hHTTPServerObj*, THTTPServer_Settings ∗ *pstSettings*)

Set a HTTPServer instance's parameters.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle to set parameters

pstSettings *(i)* pointer to a **THTTPServer_Settings** structure variable storing settings for the HTTPServer instance

Return values:

S_OK parameters set successfully.

others set parameters failed. check error codes.

SCODE HTTPServer_Start (HANDLE *hHTTPServerObj*)

Start a HTTPServer instance's service.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle

Return values:

S_OK HTTPServer instance starts successfully.

others start HTTPServer instance failed. check error codes.

See also:

HTTPServer_Stop

SCODE HTTPServer_Stop (HANDLE *hHTTPServerObj*)

Stop a HTTPServer instance.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle

Return values:

S_OK instance stopped successfully.

others instance stop failed. check error codes.

See also:

HTTPServer_Start

SCODE HTTPServer_TakeClientOut (HANDLE *hHTTPServerObj*, DWORD *dwClientID*, SOCKET ∗ *psSocket*)

Set a HTTPServer instance's individual client's parameters Get a HTTPServer instance's client's socket descriptor and take the client out.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle

dwClientID *(i)* the client id

psSocketID *(i/o)* pointer to a SOCKET to receive the returned socket descriptor

Return values:

S_OK function done successfully.

others function failed. check error codes.

4.4 httpserver.h File Reference

HTTP Server Module API.

```
#include "netdef.h"
#include "vssdef.h"
#include "common.h"
```

Data Structures

- struct THTTPServer_AcceptData
- struct **THTTPServer_Author_Info**
- struct THTTPServer_ClientSettings
- struct THTTPServer_Disconnect
- struct THTTPServer_InitSettings
- struct **THTTPServer_RequestData**
- struct **THTTPServer_Send**
- struct THTTPServer_Settings

Defines

- #define **HTTPSERVER_VERSION** MAKEFOURCC(1, 0, 0, 0)
- #define **MAX_USERNAME_LEN** (MAX_NAME_LEN + 1)
- #define **MAX_PASSWORD_LEN** (MAX_PASS_LEN + 1)
- #define **MAX_HTTPMETHOD_LEN** 8
- #define **MAX_HTTPVERSION_LEN** 6
- #define **MAX_URL_LEN** 256
- #define **MAX_QUERY_LEN** MAX_EXTRAINFO_LEN
- #define **MAX_FILENAME_LEN** (MAX_PATH_LEN + 1)
- #define **ERR_HTTPSERVER_VERSION** 0x80010001
- #define **ERR_HTTPSERVER_MALLOC** 0x80010002
- #define **S_HTTPSERVER_SEND_COMPLETE** 0x00010001
- #define **S_HTTPSERVER_SEND_PENDING** 0x00010002
- #define HTTPServer_ReceiveBufferSizeFlag 0x001
- #define HTTPServer_ServerIPAddressFlag 0x002
- #define HTTPServer_PortFlag 0x004
- #define HTTPServer_MaxConnectionsFlag 0x008
- #define HTTPServer_ThreadPriorityFlag 0x010
- #define HTTPServer_StackSizeFlag 0x020
- #define HTTPServer_KeepAliveFlag 0x040
- #define HTTPServer_AuthorizationFlag 0x080
- #define HTTPServer_ConnectionTimeoutFlag 0x100
- #define HTTPServer_HostNameFlag 0x200
- #define HTTPServer_LanguageFlag 0x400
- #define **HTTPServer_AuthorizationType_Basic** 0x001
- #define **HTTPServer_AuthorizationType_Digest** 0x002
- #define **HTTPServer_Callback_Accept** 1
- #define **HTTPServer_Callback_Authorize** 2
- #define **HTTPServer_Callback_Request** 3
- #define **HTTPServer_Callback_Disconnect** 4
- #define **HTTPServer_Callback_Send** 5
- #define **HTTPServer_Callback_Multipart_Head** 6
- #define **HTTPServer_Callback_Multipart_Data** 7
- #define **HTTPServer_Callback_Multipart_Request** 8
- #define **HTTPServer_Callback_Alive** 9
- #define **HTTPServer_Callback_GetServerName** 10
- #define **HTTPServer_Callback_Digest_Auth_Request** 11

Typedefs

- typedef int(∗ HTTPServer_Callback)(DWORD dwInstance, DWORD dwCallbackType, void ∗pvCallbackData)

  callback function definiton.

Enumerations

- enum **THttpRequestCommand** { **HTTP_NO_COMMAND**, **HTTP_GET_COMMAND**, **HTTP_HEAD_COMMAND**, **HTTP_POST_COMMAND**, **HTTP_OPTIONS_COMMAND**, **HTTP_PUT_COMMAND**, **HTTP_DELETE_COMMAND**, **HTTP_TRACE_COMMAND** }
- enum **THttpUrlResponse** { **HTTP_URL_OK**, **HTTP_URL_OK_STATIC**, **HTTP_URL_REDIRECT**, **HTTP_URL_NOT_MODIFIED**, **HTTP_URL_UNAUTHORIZED**, **HTTP_URL_NOT_FOUND**, **HTTP_URL_UNAVAILABLE**, **HTTP_URL_NOT_RESPONSE** }
- enum **THttpDataType** { **DATA_TYPE_TEXT**, **DATA_TYPE_HTML**, **DATA_TYPE_IMAGE_GIF**, **DATA_TYPE_IMAGE_JPEG**, **DATA_TYPE_TUNNELLED**, **DATA_TYPE_APPLICATION_OCTETSTREAM** }
- enum **THttpDataSource** { **DATA_SOURCE_MEMORY**, **DATA_SOURCE_FILE**, **DATA_SOURCE_PENDING** }

Functions

- SCODE **HTTPServer_GetVersion** (BYTE ∗byMajor, BYTE ∗byMinor, BYTE ∗byBuild, BYTE ∗byRevision)
- SCODE HTTPServer_Initial (HANDLE ∗phHTTPServerObj, THTTPServer_InitSettings ∗pstInitSettings)

  Initialize a HTTPServer instance.

- SCODE HTTPServer_SetParameters (HANDLE hHTTPServerObj, THTTPServer_Settings ∗pstSettings)

  Set a HTTPServer instance's parameters.

- SCODE HTTPServer_Start (HANDLE hHTTPServerObj)

  Start a HTTPServer instance's service.

- SCODE HTTPServer_Stop (HANDLE hHTTPServerObj)

  Stop a HTTPServer instance.

- SCODE HTTPServer_Release (HANDLE ∗phHTTPServerObj)

  Release a HTTPServer instance.

- SCODE HTTPServer_SetCallback (HANDLE hHTTPServerObj, HTTPServer_Callback pfnCallback, DWORD dwInstance)

  set a HTTPServer instance's callback function.

- SCODE HTTPServer_Disconnect (HANDLE hHTTPServerObj, DWORD dwClientID)

  Disconnect a HTTPServer instance's client.

- SCODE **HTTPServer_SendData** (HANDLE hHTTPServerObj, DWORD dwClientID, PCHAR pSendBuffer, DWORD dwSendLength, BOOL bLastData, DWORD ∗pdwByteSent)
- SCODE HTTPServer_TakeClientOut (HANDLE hHTTPServerObj, DWORD dwClientID, SOCKET ∗psSocket)

  Set a HTTPServer instance's individual client's parameters Get a HTTPServer instance's client's socket descriptor and take the client out.

4.4.1 Detailed Description

HTTP Server Module API.

Copyright (C) 2003 Vivotek, Inc. All rights reserved.

HTTPServer.h

Date:
2003/03/11

Author:
Bill, Shu-Zhe Weng

4.4.2 Define Documentation

#define HTTPServer_AuthorizationFlag 0x080

settings flag to select authorization or not

#define HTTPServer_ConnectionTimeoutFlag 0x100

settings flag to set connection timeout value

#define HTTPServer_HostNameFlag 0x200

settings flag to set host name

#define HTTPServer_KeepAliveFlag 0x040

settings flag to select connection keep-alive or close

#define HTTPServer_LanguageFlag 0x400

settings flag to set language type

#define HTTPServer_MaxConnectionsFlag 0x008

initial settings flag to set maximum number of clients

#define HTTPServer_PortFlag 0x004

initial settings flag to set local port

#define HTTPServer_ReceiveBufferSizeFlag 0x001

initial settings flag to set receive buffer size

#define HTTPServer_ServerIPAddressFlag 0x002

initial settings flag to set local IP address

#define HTTPServer_StackSizeFlag 0x020

settings flag to set stack size of service thread

#define HTTPServer_ThreadPriorityFlag 0x010

settings flag to set the service thread's priority

4.4.3 Typedef Documentation

typedef int(∗ HTTPServer_Callback)(DWORD dwInstance, DWORD dwCallbackType, void∗ pvCallbackData)

callback function definiton.

Parameters:

dwInstance *(i)* parent instance

dwCallbackType *(i)* callback type, includes

1) HTTPServer_Callback_Accept: after a new connection is accepted

2) HTTPServer_Callback_Authorize: after authorization info received, return 0 if user info is accepted, return other values if access denied.

3) HTTPServer_Callback_Request: after a HTTP request received. Note: when callback on request, an internal buffer will be called back, after callback returns, module will assume the buffer is returned to module. So save all information or process them before return from callback. To send response, assign the send buffer pointer in field pcSendBuffer and set buffer size in field dwBufLen before return from callback. After send done, the send buffer pointer will be called back on type HTTPServer_Callback_Send, you can reuse or do anything to that buffer then.

4) HTTPServer_Callback_Disconnect: after a connection is disconnected

5) HTTPServer_Callback_Send: after a buffer is sent to a client

Parameters:

pvCallbackData *(i)* pointer to a block of data, each callback type has a specific callback data structure. Cast this pointer to appropriate structure pointer according to parameter dwCallbackType.

4.4.4 Function Documentation

SCODE HTTPServer_Disconnect (HANDLE *hHTTPServerObj*, DWORD *dwClientID*)

Disconnect a HTTPServer instance's client.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle

dwClientID *(i)* the client id

Return values:

S_OK disconnect request queued successfully.

others queue disconnect request failed. check error codes.

Note:

It is a non-blocking function.

SCODE HTTPServer_Initial (HANDLE ∗ *phHTTPServerObj*, THTTPServer_InitSettings ∗ *pstInitSettings*)

Initialize a HTTPServer instance.

Parameters:

phHTTPServerObject *(i/o)* handle pointer to receive the created instance

pstInitialSettings *(i)* pointer to a **THTTPServer_InitialSettings** structure variable storing initial settings to create the HTTPServer instance

Return values:

S_OK HTTPServer instance created successfully.

others HTTPServer instance creation failed. check error codes.

See also:

HTTPServer_Release

SCODE HTTPServer_Release (HANDLE ∗ *phHTTPServerObj*)

Release a HTTPServer instance.

Parameters:

phHTTPServerObject *(i)* handle pointer pointing to a HTTPServer instance to release

Return values:

S_OK HTTPServer instance released successfully.

others release HTTPServer instance failed. check error codes.

See also:

HTTPServer_Initial

SCODE HTTPServer_SetCallback (HANDLE *hHTTPServerObj*, HTTPServer_Callback *pfnCallback*, DWORD *dwInstance*)

set a HTTPServer instance's callback function.

Parameters:

hHTTPServerObject *(i)* handle of the HTTPServer instance to set callback

pfnCallback *(i)* the callback function

dwInstance *(i)* the parent instance

Return values:

S_OK callback function set successfully.

others set callback function failed. check error codes.

SCODE HTTPServer_SetParameters (HANDLE *hHTTPServerObj*, THTTPServer_Settings ∗ *pstSettings*)

Set a HTTPServer instance's parameters.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle to set parameters

pstSettings *(i)* pointer to a **THTTPServer_Settings** structure variable storing settings for the HTTPServer instance

Return values:

S_OK parameters set successfully.

others set parameters failed. check error codes.

SCODE HTTPServer_Start (HANDLE *hHTTPServerObj*)

Start a HTTPServer instance's service.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle

Return values:

S_OK HTTPServer instance starts successfully.

others start HTTPServer instance failed. check error codes.

See also:

HTTPServer_Stop

SCODE HTTPServer_Stop (HANDLE *hHTTPServerObj*)

Stop a HTTPServer instance.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle

Return values:

S_OK instance stopped successfully.

others instance stop failed. check error codes.

See also:

HTTPServer_Start

SCODE HTTPServer_TakeClientOut (HANDLE *hHTTPServerObj*, DWORD *dwClientID*, SOCKET ∗ *psSocket*)

Set a HTTPServer instance's individual client's parameters Get a HTTPServer instance's client's socket descriptor and take the client out.

Parameters:

hHTTPServerObject *(i)* the HTTPServer instance's handle

dwClientID *(i)* the client id

psSocketID *(i/o)* pointer to a SOCKET to receive the returned socket descriptor

Return values:

S_OK function done successfully.

others function failed. check error codes.

4.5 httpserver_local.h File Reference

HTTP Server Module API.

```
#include "osisolate.h"
#include "hs_states.h"
#include "hs_parse.h"
#include "hs_tcp.h"
#include "httpserver.h"
#include "common.h"
#include "encrypt_md5.h"
#include "encrypt_base64.h"
```

Data Structures

- struct **THttpConnection**
- struct **THttpServerInfo**
- struct **TPatternTable**

Defines

- #define **DbgPrint**(x)
- #define **DbgPrint1**(x)
- #define **DEFAULT_RECEIVE_BUFFER_SIZE** 1460
- #define **DEFAULT_HTTP_PORT** 80
- #define **DEFAULT_MAX_CONNECTION** 10
- #define **DEFAULT_CONNECTION_TIMEOUT** 60
- #define **DEFAULT_HOSTNAME** "Network Camera"
- #define **DEFAULT_LANGUAGE** "en"
- #define **MAX_NAME_LENGTH** 32
- #define **MAX_VALUE_LENGTH** 256

- #define **MAX_LINE_LENGTH** 256
- #define **MAX_SESSION_LENGTH** 32
- #define **MAX_SAVE_HEADER_LENGTH** 256
- #define **HTTP_HEADER_BUFFER_SIZE** 512
- #define **HTTP_BODY_BUFFER_SIZE** 1460
- #define **MAX_BOUNDARY_LENGTH** 70
- #define **HTTP_STRING** ”http://”
- #define **STR_HTTP_CONNECTION** ”Connection: ”
- #define **STR_HTTP_DATE** ”Date: ”
- #define **STR_HTTP_SERVER** ”Server:”
- #define **STR_HTTP_WWW_AUTHENTICATE_BASIC** ”WWW-Authenticate: Basic realm=\””
- #define **STR_HTTP_WWW_AUTHENTICATE_DIGEST** ”WWW-Authenticate: Digest”
- #define **STR_SERVER_HEADER** ”Server: ”
- #define **STR_NO_CACHE** ”Pragma: no-cache\x0d\x0a”
- #define **STR_ONEONE_NO_CACHE** ”Cache-Control: no-cache\x0d\x0a”
- #define **STR_ONEONE_NO_STORE** ”Cache-Control: no-store\x0d\x0a”
- #define **STR_STREAM_ID** ”SID : ”
- #define **STR_AUDIO_MODE** ”Audio Mode : ”
- #define **STR_AUDIO_NONE** ”None”
- #define **STR_AUDIO_FULLDUPLEX** ”Full Duplex”
- #define **STR_AUDIO_HALFDUPLEX** ”Half Duplex”
- #define **STR_AUDIO_TALKONLY** ”Talk Only”
- #define **STR_AUDIO_LISTENONLY** ”Listen Only”
- #define **STR_USER_PRIVILEGE** ”Privilege : ”
- #define **HTTP_METHOD_GET** ”GET ”
- #define **HTTP_METHOD_HEAD** ”HEAD ”
- #define **HTTP_METHOD_POST** ”POST ”
- #define **STR_HTTPPATTERNACCEPT** ”accept:”
- #define **KHTTPPATTERNACCEPTLANGUAGE** ”accept-language”
- #define **HTTP_PATTERN_AUTHORIZATION** ”authorization”
- #define **KHTTPPATTERNBASIC** ”basic”
- #define **KHTTPPATTERNCONNECTION** ”connection”
- #define **HTTP_PATTERN_DISPOSITION** ”content-disposition”
- #define **HTTP_PATTERN_CONTENT_LENGTH** ”content-length”
- #define **HTTP_PATTERN_CONTENT_TYPE** ”content-type”
- #define **KHTTPPATTERNCOOKIE** ”cookie”
- #define **KHTTPPATTERNDATE** ”date”
- #define **KHTTPPATTERNEXPECT** ”expect”
- #define **KHTTPPATTERNEXPIRES** ”expires”
- #define **KHTTPPATTERNEXTENSION** ”extension”
- #define **HTTP_PATTERN_HOST** ”host”
- #define **HTTP_PATTERN_IFMODIFIED** ”if-modified-since”
- #define **KHTTPPATTERNIFNONEMATCH** ”if-none-match”
- #define **KHTTPPATTERNLASTMODIFIED** ”last-modified”
- #define **KHTTPPATTERNLOCATION** ”location”
- #define **KHTTPPATTERNNOCACHE** ”no-cache”
- #define **KHTTPPATTERNPRAGMA** ”pragma”
- #define **KHTTPPATTERNREFERER** ”referer”
- #define **KHTTPPATTERNREFRESH** ”refresh”
- #define **KHTTPPATTERNSERVER** ”server”

- #define **KHTTPPATTERNSETCOOKIE** ”set-cookie”
- #define **KHTTPPATTERNTRANSFERENCODING** ”transfer-encoding”
- #define **KHTTPPATTERNUPDATE** ”update”
- #define **KHTTPPATTERNUSERAGENT** ”user-agent”
- #define **KHTTPPATTERNAUTHENTICATE** ”www-authenticate”
- #define **HTTP_PATTERN_SESSIONCOOKIE** ”x-sessioncookie”
- #define **STR_HTTP_ALLOW** ”Allow:”
- #define **STR_HTTP_CONTENT_LENGTH** ”Content-Length: ”
- #define **STR_HTTP_CONTENT_TYPE** ”Content-Type: ”
- #define **STR_HTTP_CONTENT_DISPOSITION_ATTACHMENT** ”Content-Disposition: attachment;”
- #define **STR_HTTP_CONTENT_LANGUAGE** ”Content-Language: ”
- #define **STR_HTTP_EXPIRES** ”Expires: ”
- #define **STR_HTTP_LAST_MODIFIED** ”Last-Modified: ”
- #define **STR_HTTP_KEEP_ALIVE** ”Keep-Alive”
- #define **STR_HTTP_CLOSE** ”close”
- #define **STR_HTTP_BOUNDARY** ”boundary”
- #define **STR_HTTP_FILENAME** ”filename”
- #define **STR_HTTP_NAME** ”name”
- #define **STR_HTTP_VERSION** ”HTTP/1.1”
- #define **STR_HTTP_VERSION_1DOT0** ”HTTP/1.0”
- #define **STR_100_CONTINUE** ” 100 Continue\x0d\x0a”
- #define **STR_200_OK** ” 200 OK\x0d\x0a”
- #define **STR_304_NOT_MODIFIED** ” 304 Not Modified\x0d\x0a”
- #define **STR_400_BAD_REQUEST** ” 400 Bad Request\x0d\x0a”
- #define **STR_401_UNAUTHORIZED** ” 401 Unauthorized\x0d\x0a”
- #define **STR_404_PAGE_NOT_FOUND** ” 404 Not Found\x0d\x0a”
- #define **STR_405_METHOD_NOT_ALLOWED** ” 405 Method Not Allowed\x0d\x0a”
- #define **STR_413_REQUEST_TOO_LARGE** ” 413 Request Entity Too Large\x0d\x0a”
- #define **STR_500_SERVER_ERROR** ” 500 Internal Server Error\x0d\x0a”
- #define **STR_501_NOT_IMPLEMENTED** ” 501 Not Implemented\x0d\x0a”
- #define **STR_503_SERVICE_UNAVAILABLE** ” 503 Service Unavailable\x0d\x0a”
- #define **K_CRLF** ”\x0d\x0a”
- #define **K_QUOTE_CRLF** ”\”\x0d\x0a”
- #define **K_QUOTE** ”\””
- #define **K_SPACE** ” ”
- #define **K_QUESTION** ”?”
- #define **K_COMMA** ”,”
- #define **K_PERIOD** ”.”
- #define **K_COLON** ”:”
- #define **K_EQUAL** ”=”
- #define **K_AMPERSAND** ”&”
- #define **K_OPENANGLE** ”<”
- #define **K_CLOSEANGLE** ”>”
- #define **K_EMPTYLENGTH** ”\x30\x0d\x0a”
- #define **MAX_MIME_TYPE_LENGTH** 50
- #define **STR_TYPE_HTML** ”text/html”
- #define **STR_TYPE_TEXT** ”text/plain”
- #define **STR_TYPE_GIFIMAGE** ”image/gif”
- #define **STR_TYPE_JPEGIMAGE** ”image/jpeg”

- #define **STR_TYPE_MULTIPARTFORM** ”multipart/form-data”
- #define **STR_TYPE_TUNNELLED** ”application/x-rtsp-tunnelled”
- #define **STR_TYPE_APPLICATION_OCTETSTREAM** ”application/octet-stream”
- #define **ASCII_Null** 0x00
- #define **ASCII_0** 0x30
- #define **ASCII_9** 0x39
- #define **ASCII_A** 0x41
- #define **ASCII_F** 0x46
- #define **ASCII_Z** 0x5A
- #define **ASCII_a** 0x61
- #define **ASCII_f** 0x66
- #define **ASCII_z** 0x7A
- #define **ASCII_Return** 0x0D
- #define **ASCII_Newline** 0x0A
- #define **ASCII_Space** 0x20
- #define **ASCII_Quote** 0x22
- #define **ASCII_Hash** 0x23
- #define **ASCII_Percent** 0x25
- #define **ASCII_Ampersand** 0x26
- #define **ASCII_Plus** 0x2B
- #define **ASCII_Comma** 0x2C
- #define **ASCII_Hyphen** 0x2D
- #define **ASCII_Dot** 0x2E
- #define **ASCII_Slash** 0x2F
- #define **ASCII_Colon** 0x3A
- #define **ASCII_SemiColon** 0x3B
- #define **ASCII_LeftArrow** 0x3C
- #define **ASCII_OpenAngle** 0x3C
- #define **ASCII_Equal** 0x3D
- #define **ASCII_RightArrow** 0x3E
- #define **ASCII_CloseAngle** 0x3E
- #define **ASCII_Question** 0x3F
- #define **ASCII_Backslash** 0x5C
- #define **DATA_SOURCE_INTERNAL** 99

Typedefs

- typedef ∗ **PTHttpConnection**
- typedef ∗ **PTHttpServerInfo**
- typedef void **PatternProcedure** (PTHttpConnection ptConnection, char ∗pszStartOfToken, UINT uiTokenLength)
- typedef ∗ **PTPatternTable**

Functions

- void **StrLenCpy** (char ∗pszTo, char ∗pszFrom, DWORD dwLength)
- void **StrLenCpyTruncate** (char ∗pszTo, char ∗pszFrom, DWORD dwLength)
- void **EscapeDecodeString** (char ∗pszEncodedString, DWORD dwEncodeLen, char ∗pszDecodedString, DWORD ∗pdwDecodeLen, BOOL bFormData)
- DWORD THREADAPI **HttpServerMainTask** (DWORD dwInstance)
- SCODE **InitHttpConnections** (PTHttpServerInfo pServerInfo)
- DWORD **DecodeBase64Data** (char ∗pszInputBuf, UINT uiInputLen, char ∗pszOutputBuf)
- void **HandleHttpHeader** (PTPatternTable ptPatternTable, PTHttpConnection ptConnection)
- void **InitRequestStates** (PTHttpConnection ptConnection)
- BOOL **ParseHttpHeaders** (PTHttpConnection ptConnection)
- void **PrepareHttpData** (PTHttpConnection ptConnection)
- BOOL **GetObjectData** (PTHttpConnection ptConnection)
- BOOL **ParseMultipartHeaders** (PTHttpConnection ptConnection)
- BOOL **GetMultipartObjectData** (PTHttpConnection ptConnection)
- SCODE **HttpResponseHeader** (PTHttpConnection ptConnection)
- SCODE **HttpResponseBody** (PTHttpConnection ptConnection)

Variables

- const char ∗ **gMimeTypes** [ ]
- TPatternTable **gtPatternTable** [ ]
- TPatternTable **gtMpPatternTable** [ ]
- TPatternTable **gtDispPatternTable** [ ]

4.5.1 Detailed Description

HTTP Server Module API.

Copyright (C) 2003 Vivotek, Inc. All rights reserved.

HTTPServer.h

Date:
2003/03/11

Author:
Bill, Shu-Zhe Weng

4.6 mediatypedef.h File Reference

header file of media type related.

Enumerations

- enum EMediaCodecType { mctJPEG = 0x0001, mctH263 = 0x0002, mctMP4V = 0x0004, mctG7221 = 0x0100, mctG729A = 0x0200, mctAAC = 0x0400, mctGAMR = 0x0800, mctG711U = 0x1000, mctG711A = 0x2000 }
- enum TMediaDBFrameType { MEDIADB_FRAME_INTRA = 0, MEDIADB_FRAME_PRED = 1, MEDIADB_FRAME_BIPRED = 2 }

4.6.1 Detailed Description

header file of media type related.

Copyright 2000-2002 Vivotek, Inc. All rights reserved.

mediatypedef.h

Date:
2003/08/07

Author:
May Hsu

4.6.2 Enumeration Type Documentation

enum EMediaCodecType

The media codec type

Enumeration values:

mctJPEG the codec type is JPEG (image, video)

mctH263 the codec type is H263 (video)

mctMP4V the codec type is MPEG-4 video (video)

mctG7221 the codec type is G.722.1 (audio)

mctG729A the codec type is G.729A (audio)

mctAAC the codec type is AAC (audio)

mctGAMR the codec type is GAMR (audio)

mctG711U the codec type is G.711 u-law

mctG711A the codec type is G.711 a-law

enum TMediaDBFrameType

the type of frame

Enumeration values:

MEDIADB_FRAME_INTRA the frame is intra

MEDIADB_FRAME_PRED the frame is prediction

MEDIADB_FRAME_BIPRED the frame is bi-direction prediction

4.7 netdef.h File Reference

some common definitions for network.

Defines

- #define MAX_IPADDR_LEN 15

  max length of ip address.

- #define MAX_DOMAIN_NAME_LEN 63

  max length of domain name.

- #define MAX_PATH_LEN 255

  max length of file & url path.

- #define MAX_DEVPATH_LEN 15

  max length of device name.

- #define MAX_NAME_LEN 16

  max length of user name.

- #define MAX_PASS_LEN 14

  max length of user password.

- #define **MAX_NETAP_PASS_LEN** 15
- #define MAX_HOST_LEN 40

  max length of host name.

- #define MAC_ADDR_LEN 12

  length of mac address.

- #define MAX_EXTRAINFO_LEN 8192

  max length of extra information in http.

- #define NETWORK_PACKET_SIZE 1460

  The ethernet packet size.

- #define **MAX_SMTP_NAME_LEN** 63
- #define **MAX_SMTP_EMAIL_LEN** 80
- #define **MAX_FTP_FOLDER_LEN** 40
- #define **MAX_DDNS_HOST_LEN** 127
- #define **MAX_DDNS_PASS_LEN** 20
- #define MAX_ACCESS_NAME_LEN 20

  for RTSP access name.

4.7.1 Detailed Description

some common definitions for network.

Copyright 2000-2003 Vivotek, Inc. All rights reserved.

netdef.h

Date:

2003/03/21

Author:

Joe Wu

4.8 rtprtcp_utility.c File Reference

Utility routines for RTP/RTCP implement file. Moved from rtprtcp.c.

`#include "rtprtcp_local.h"`

Defines

- #define **W32Len**(l) ((l + 3) / 4)

Functions

- UINT32 **RTPRTCP_BitFieldSet** (UINT32 uiValue, UINT32 uiBitField, int iStartBit, int iBits)
- UINT32 **RTPRTCP_BitFieldGet** (UINT32 uiValue, int iStartBit, int iBits)
- void **RTPRTCP_SetSDES** (rtcpSDesType eType, rtcpSDES ∗ptSdes, char ∗pcData, int iLength)
- int **RTPRTCP_ConvertHeader2h** (UINT32 ∗plBuff, int iStartIndex, int iSize)
- int **RTPRTCP_ConvertHeader2l** (UINT8 ∗piBuff, int iStartIndex, int iSize)
- void **RTPRTCP_Time2MinSeconds** (UINT64_NTP ∗pNTP64, unsigned long ulSeconds, unsigned long ulMSeconds)
- void **RTPRTCP_MakeHeader** (rtcpHeader ∗ptHeader, UINT32 uiSsrc, UINT8 uiCount, rtcpType eType, UINT16 uiDataLen)
- int **RTPRTCP_ProcessRTCPPacket** (char ∗pcData, INT32 iDataLen, rtcpType eType, INT32 i-ReportCount, rtpSession ∗ptRTPSession)
- UINT32 **RTPRTCP_GetTimeStamp** (RTPMEDIABUFFER ∗ptBuf, rtpSession ∗ptRTPSession)

4.8.1 Detailed Description

Utility routines for RTP/RTCP implement file. Moved from rtprtcp.c.

Copyright 2007-2010 Vivotek, Inc. All rights reserved.

rtprtcp_utility.c

Date:

2007/03/30

Author:

Jeffrey Lee $Orginal draft by ShengFu$

4.9 rtsps.c File Reference

rtsp streaming server for Kilrogg.

`#include <arpa/inet.h>`

`#include <stdlib.h>`

`#include <sys/stat.h>`

`#include "osisolate.h"`

```
#include "rtsps_local.h"
#include "rtsps_fdipc.h"
#include "rtsps_ubuffer.h"
#include "rtsps_callback.h"
#include "xmlsparser.h"
```

Functions

- SCODE **CfgParser_GetUnixDomainSocket** (void *pData, void *pParam)
- SCODE **CfgParser_GetFIFO** (char *pData, int *piFIFO, int iFlag)
- SCODE **CfgParser_GetIPAddress** (void *pData, void *pParam)
- SCODE **CfgParser_GetAuthenticateMode** (void *pData, void *pParam)
- SCODE **SetUpRTSPServer** (TSTREAMSERVERINFO *pThis)
- int **StreamingServer_ParseTrackID** (char *pcTrackName)
- SCODE **StreamingServer_ParseAccessFile** (TSTREAMSERVERINFO *pThis, char *pzAccessFile)
- SCODE **StreamingServer_ParseConfigFile** (TSTREAMSERVERINFO *pThis, char *pzConfigFile)
- SCODE StreamingServer_SetMediaTrackParam (HANDLE hObject)

  Release the resources of a RTSPS object.

- void StreamingServer_AccountManagerParse (HANDLE hObject)

  Parse and load the account manager information.

- int **StreamingServer_AccountManagerInit** (TSTREAMSERVERINFO *ptStreamServerInfo)
- SCODE StreamingServer_Initial (HANDLE *phObject, TRTSPSInitOptions *pInitOpts, char *pzConfigFile, char *pzAccessFile)

  Create handle of RTSPS object.

- SCODE StreamingServer_Start (HANDLE hObject)

  Start the operation of rtsp streaming server.

- SCODE StreamingServer_Stop (HANDLE hObject)

  Stop the operation of rtsp streaming server.

- SCODE StreamingServer_Release (HANDLE *phObject)

  Release the resources of a RTSPS object.

- int StreamingServer_UpdateDynamicPamater (HANDLE hRTSPS, char *pzConfigFile)

  Update RTSP server dynamic parameters.

Variables

- char **acVideoTrackName** [VIDEO_TRACK_NUMBER][MEDIA_TRACK_NAME_LEN] = {"trackID=1","trackID=2"}
- char **acAudioTrackName** [AUDIO_TRACK_NUMBER][MEDIA_TRACK_NAME_LEN] = {"trackID=4"}

- TSTREAMSERVERINFO **tStreamServerInfo**
- TCfgParseMap **g_AcsParseMap** [ ]
- TCfgParseMap **g_CfgParseMap** [ ]

4.9.1 Detailed Description

rtsp streaming server for Kilrogg.

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps.c

Date:
2006/04/21

Author:
Rey Cheng

4.9.2 Function Documentation

void StreamingServer_AccountManagerParse (HANDLE *hObject*)

Parse and load the account manager information.

Parameters:
hObject *(i)* handle of the RTSPS object

SCODE StreamingServer_Initial (HANDLE ∗ *phObject*, TRTSPSInitOptions ∗ *pInitOpts*, char ∗ *pzConfigFile*, char ∗ *pzAccessFile*)

Create handle of RTSPS object.

Parameters:
phObject *(o)* pointer to receive the handle of the RTSPS object

Return values:
S_OK Create object ok

S_FAIL Create object failed

Remarks:
A RTSPS object shall be initialized before using it.

See also:
RTSPStreamingServer_Release

SCODE StreamingServer_Release (HANDLE ∗ *phObject*)

Release the resources of a RTSPS object.

Parameters:

phObject *(i/o)* pointer to the handle of the RTSPS object

Return values:

S_OK Release object ok

S_FAIL Release object failed

See also:

RTSPStreamingServer_Start

SCODE StreamingServer_SetMediaTrackParam (HANDLE *hObject*)

Release the resources of a RTSPS object.

Parameters:

hObject *(i)* handle of the RTSPS object

Return values:

S_OK Release object ok

S_FAIL Release object failed

See also:

RTSPStreamingServer_Start

SCODE StreamingServer_Start (HANDLE *hObject*)

Start the operation of rtsp streaming server.

Parameters:

hObject Handle of the RTSPS object

Return values:

S_OK Start rtsp streaming ok

S_FAIL Start rtsp streaming failed

Remarks:

Be sure to call RTSPStreamingServer_Initial() before calling this.

See also:

RTSPStreamingServer_Stop

SCODE StreamingServer_Stop (HANDLE *hObject*)

Stop the operation of rtsp streaming server.

Parameters:

hObject Handle of the RTSPS object

Return values:

S_OK Stop rtsp streaming ok

S_FAIL Stop rtsp streaming failed

See also:

RTSPStreamingServer_Start

int StreamingServer_UpdateDynamicPamater (HANDLE *hRTSPS*, char ∗ *pzConfigFile*)

Update RTSP server dynamic parameters.

Parameters:

hRTSPS *(i)* pointer to the handle of the RTSPS object

pzConfigFile *(i)* string which contains the config information

Return values:

0 Update RTSP server dynamic parameters ok

others Update RTSP server dynamic parameters failed

4.10 rtsps.h File Reference

rtsp streaming server for Kilrogg.

```
#include <semaphore.h>
#include "typedef.h"
#include "rtsprtpcommon.h"
#include "rtpmediaqueue.h"
#include "rtprtcp.h"
#include "mediachannel.h"
#include "rtspserver.h"
#include "rtppacketizer.h"
#include "rtspstreamingserver.h"
#include "datapacketdef.h"
#include "ubuffer.h"
#include "account_mgr_app.h"
```

Data Structures

- struct rtsps_initial_options

Defines

- #define RTSPS_VERSION MAKEFOURCC(1, 1, 0, 0)
- #define **RTSP_SYSLOG_ID_STRING** ”[RTSP SERVER]”
- #define **RTSPS_PID_FILE** ”/var/run/rtsps.pid”

Typedefs

- typedef rtsps_initial_options TRTSPSInitOptions

Functions

- SCODE StreamingServer_Initial (HANDLE ∗phObject, TRTSPSInitOptions ∗pInitOpts, char ∗pzConfigFile, char ∗pzAccessFile)

  Create handle of RTSPS object.

- SCODE StreamingServer_Start (HANDLE hObject)

  Start the operation of rtsp streaming server.

- SCODE StreamingServer_Stop (HANDLE hObject)

  Stop the operation of rtsp streaming server.

- SCODE StreamingServer_Release (HANDLE ∗phObject)

  Release the resources of a RTSPS object.

- SCODE StreamingServer_SetMediaTrackParam (HANDLE hObject)

  Release the resources of a RTSPS object.

- void StreamingServer_AccountManagerParse (HANDLE hObject)

  Parse and load the account manager information.

- int StreamingServer_UpdateDynamicPamater (HANDLE hRTSPS, char ∗pzConfigFile)

  Update RTSP server dynamic parameters.

4.10.1 Detailed Description

rtsp streaming server for Kilrogg.

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps.h

Date:

2006/04/21

Author:

Rey Cheng

4.10.2 Define Documentation

#define RTSPS_VERSION MAKEFOURCC(1, 1, 0, 0)

FOUR_CC Version code of your **RTSPS** instance.

4.10.3 Typedef Documentation

typedef struct rtsps_initial_options TRTSPSInitOptions

RTSPS initial options

4.10.4 Function Documentation

void StreamingServer_AccountManagerParse (HANDLE *hObject*)

Parse and load the account manager information.

Parameters:

hObject *(i)* handle of the RTSPS object

SCODE StreamingServer_Initial (HANDLE ∗ *phObject*, TRTSPSInitOptions ∗ *pInitOpts*, char ∗ *pzConfigFile*, char ∗ *pzAccessFile*)

Create handle of RTSPS object.

Parameters:

phObject *(o)* pointer to receive the handle of the RTSPS object

Return values:

S_OK Create object ok

S_FAIL Create object failed

Remarks:

A RTSPS object shall be initialized before using it.

See also:

RTSPStreamingServer_Release

SCODE StreamingServer_Release (HANDLE ∗ *phObject*)

Release the resources of a RTSPS object.

Parameters:

phObject *(i/o)* pointer to the handle of the RTSPS object

Return values:

S_OK Release object ok

S_FAIL Release object failed

See also:

RTSPStreamingServer_Start

SCODE StreamingServer_SetMediaTrackParam (HANDLE *hObject*)

Release the resources of a RTSPS object.

Parameters:

hObject *(i)* handle of the RTSPS object

Return values:

S_OK Release object ok

S_FAIL Release object failed

See also:

RTSPStreamingServer_Start

SCODE StreamingServer_Start (HANDLE *hObject*)

Start the operation of rtsp streaming server.

Parameters:

hObject Handle of the RTSPS object

Return values:

S_OK Start rtsp streaming ok

S_FAIL Start rtsp streaming failed

Remarks:

Be sure to call RTSPStreamingServer_Initial() before calling this.

See also:

RTSPStreamingServer_Stop

SCODE StreamingServer_Stop (HANDLE *hObject*)

Stop the operation of rtsp streaming server.

Parameters:

hObject Handle of the RTSPS object

Return values:

S_OK Stop rtsp streaming ok

S_FAIL Stop rtsp streaming failed

See also:

RTSPStreamingServer_Start

int StreamingServer_UpdateDynamicPamater (HANDLE *hRTSPS*, char ∗ *pzConfigFile*)

Update RTSP server dynamic parameters.

Parameters:

hRTSPS *(i)* pointer to the handle of the RTSPS object

pzConfigFile *(i)* string which contains the config information

Return values:

0 Update RTSP server dynamic parameters ok

others Update RTSP server dynamic parameters failed

4.11 rtsps_callback.c File Reference

Callback functions for rtsp streaming server.

```
#include <string.h>
#include <signal.h>
#include <sys/file.h>
#include "rtsps_callback.h"
#include "encrypt_md5.h"
#include "rtsps_ubuffer.h"
```

Defines

- #define RTPM4APACK_AU_SIZE_LENGTH 13
- #define RTPM4APACK_AU_INDEX_LENGTH 3
- #define **MAX_AMR_FRAMES_PER_UBUFFER** 16
- #define **UserName** ”root”
- #define **Password** ””
- #define **AUDIO_TAG_LEN_BYTES** 4

Functions

- int **M4ABitstreamPack** (TBitstreamBuffer ∗pBitstreamBuf, TUBuffer ∗pUBuffer, int iIndex)
- SCODE **AMRBitstreamPack** (TBitstreamBuffer ∗pBitstreamBuf, TUBuffer ∗pUBuffer, int iIndex, int iFramesPerUBuffer)
- SCODE **StreamSvrWriteFile** (char ∗pzFilePathName, char ∗pWriteBuff, int iWriteLength)
- SCODE **StreamSvrAudioInCallback** (DWORD dwInstance, DWORD dwCallbackType, void ∗pvCallbackData)
- SCODE **StreamSvrVideoCallback** (DWORD dwInstance, DWORD dwCallbackType, void ∗pvCallbackData)
- SCODE **StreamSvrCheckIfMediaTrackForMulticast** (DWORD dwInstance, DWORD dwSDPIndex, DWORD dwMediaType)
- SCODE **StreamSvrCtrlChCallback** (DWORD dwInstance, DWORD dwConnectionID, DWORD dwCallbackType, DWORD dwCallbackData)

Variables

- const BYTE **g_abyFT2Size** [16] = {12, 13, 15, 17, 19, 20, 26, 31, 5, 0, 0, 0, 0, 0, 0, 0}

4.11.1 Detailed Description

Callback functions for rtsp streaming server.

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps_callback.c

Date:
2006/05/11

Author:
Rey Cheng

4.11.2 Define Documentation

#define RTPM4APACK_AU_INDEX_LENGTH 3

The number of bits on which the AU-index field is encoded in the first AU-header. AND the number of bits on which the AU-Index-delta field is encoded in any non-first AU-header. This value MUST consist with the IndexLength and IndexDeltaLength fields in SDP.

#define RTPM4APACK_AU_SIZE_LENGTH 13

The number of bits on which the AU-size field is encoded in the AU-header. This value MUST consist with the SizeLength field in SDP.

4.12 rtsps_callback.h File Reference

Callback functions for rtsp streaming server. (header file for rtsps_callback.c).

```
#include "rtsps_local.h"
#include "rtsps_ubuffer.h"
```

Functions

- SCODE **StreamSvrAudioInCallback** (DWORD dwInstance, DWORD dwCallbackType, void *pvCallbackData)
- SCODE **StreamSvrVideoCallback** (DWORD dwInstance, DWORD dwCallbackType, void *pvCallbackData)
- SCODE **StreamSvrCtrlChCallback** (DWORD dwInstance, DWORD dwConnectionID, DWORD dwCallbackType, DWORD dwCallbackData)

4.12.1 Detailed Description

Callback functions for rtsp streaming server. (header file for rtsps_callback.c).

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps_callback.h

Date:
2006/05/11

Author:
Rey Cheng

4.13 rtsps_fdipc.c File Reference

Exchange file descriptor between http server and rtsp server.

```
#include <sys/stat.h>
#include "rtspstreamingserver.h"
#include "rtsps_fdipc.h"
#include "message.h"
```

Defines

- #define **GENERAL_BUFFER_LENGTH** 1024

Functions

- SCODE **RTSPSSetupFdIPCSocket** (HANDLE hObject)
- void **RTSPHandleRTSPOverHTTPSocket** (TSTREAMSERVERINFO ∗pThis, TStreamServer_ConnectionSettings ∗ptSSConnSettings)
- int **ReadAndParseControlMessage** (int iFD, int iLength, char ∗pcBuffer)
- DWORD **RTPOverHttpSocketExchanger** (DWORD dwInstance)

4.13.1 Detailed Description

Exchange file descriptor between http server and rtsp server.

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps_fdipc.c

Date:
2006/05/11

Author:
Rey Cheng

4.14 rtsps_fdipc.h File Reference

Exchange file descriptor between http server and rtsp server. (header file for rtsps_fdipc.c).

```
#include <string.h>
#include "fdipc.h"
#include "rtsps_local.h"
```

Functions

- DWORD **RTPOverHttpSocketExchanger** (DWORD dwInstance)
- SCODE **RTSPSSetupFdIPCSocket** (HANDLE hObject)

4.14.1 Detailed Description

Exchange file descriptor between http server and rtsp server. (header file for rtsps_fdipc.c).

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps_fdipc.h

Date:
2006/05/11

Author:
Rey Cheng

4.15 rtsps_local.h File Reference

rtsp streaming server for Kilrogg (local include file).

```
#include "rtsps.h"
```

Data Structures

- struct **rtsp_info**
- struct **TIPFILTER**
- struct **TMEDIADSTINFO**
- struct **TMEDIASRCINFO**
- struct **TRTPINFO**
- struct **TRTSPINFO**
- struct **TSIPINFO**
- struct **TSTREAMINFO**

Defines

- #define **VSTREAM_SOCK_DIR** "/tmp/venc/"
- #define **ASTREAM_SOCK_DIR** "/tmp/aenc/"
- #define **HTTP_FDIPC_SOCK** "/tmp/httpfdipc.sck"
- #define **FILE_NAME_SDP1** "/tmp/livea.sdp"
- #define **FILE_NAME_SDP2** "/tmp/liveb.sdp"

- #define **SDP_PATH** ”/tmp/”
- #define **MEDIA_TYPE_VIDEO** 123
- #define **MEDIA_TYPE_AUDIO** 456
- #define **UBUFFER_PROCESS_PRIORITY** 120
- #define **UBUFFER_PROCESS_STACKSIZE** 16384
- #define **VIDEO_UBUFFER_SIZE** 64∗1024
- #define **AUDIO_UBUFFER_SIZE** 10∗1024
- #define **MAX_CONNECT_NUM** 10
- #define **MAX_MP4V_PACKET_NUM** 40
- #define **MAX_AUDIO_PACKET_NUM** 1
- #define **PRIORITY_STREAMCONTROL** 145
- #define **MAX_BITSTREAM_SIZE** (262144 + 8192)
- #define **SDP_FULL_PATH_NAME_LEN** 255
- #define **MEDIA_TRACK_NAME_LEN** 30
- #define **VIDEO_TRACK_NUMBER** 2
- #define **AUDIO_TRACK_NUMBER** 1
- #define **IPFILTER_NUMBER** 10
- #define **CONTROL_MSG_START** ”<control id=\”0\”><output>start</output></control>”
- #define **CONTROL_MSG_STOP** ”<control id=\”0\”><output>stop</output></control>”
- #define **CONTROL_MSG_FORCECI** ”<control id=\”0\”><forceCI/></control>”
- #define **CONTROL_MSG_FORCEINTRA** ”<control id=\”0\”><forceIntra/></control>”

Typedefs

- typedef rtsp_info **TSTREAMSERVERINFO**

Variables

- char **acVideoTrackName** [VIDEO_TRACK_NUMBER][MEDIA_TRACK_NAME_LEN]
- char **acAudioTrackName** [AUDIO_TRACK_NUMBER][MEDIA_TRACK_NAME_LEN]

4.15.1 Detailed Description

rtsp streaming server for Kilrogg (local include file).

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps_local.h

Date:
2006/04/21

Author:
Rey Cheng

4.16 rtsps_ubuffer.c File Reference

UBuffer reader for rtsp streaming server.

```
#include "rtsps_ubuffer.h"
```

Functions

- int **create_unix_socket** (const char ∗path)
- SCODE **GetVideoUBuffer** (TSTREAMSERVERINFO ∗pThis, int ∗piMediaTrackIndex)
- SCODE **GetAudioUBuffer** (TSTREAMSERVERINFO ∗pThis, int ∗piMediaTrackIndex)
- SCODE **initClientSocket** (int ∗piFd)
- SCODE **connectClientSocket** (int fdOut, const char ∗szSckName)
- int **writeClientSocket** (int fdOut, char ∗acUBuffer, DWORD dwWriteSize)

4.16.1 Detailed Description

UBuffer reader for rtsp streaming server.

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps_ubuffer.c

Date:
2006/05/11

Author:
Rey Cheng

4.17 rtsps_ubuffer.h File Reference

UBuffer reader for rtsp streaming server. (header file for rtsps_ubuffer.c).

```
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "rtsps_local.h"
```

Functions

- int **create_unix_socket** (const char ∗path)
- SCODE **GetVideoUBuffer** (TSTREAMSERVERINFO ∗pThis, int ∗piMediaTrackIndex)
- SCODE **GetAudioUBuffer** (TSTREAMSERVERINFO ∗pThis, int ∗piMediaTrackIndex)
- SCODE **initClientSocket** (int ∗piFd)
- int **writeClientSocket** (int fdOut, char ∗acUBuffer, DWORD dwWriteSize)
- SCODE **connectClientSocket** (int fdOut, const char ∗szSckName)

4.17.1 Detailed Description

UBuffer reader for rtsp streaming server. (header file for rtsps_ubuffer.c).

Copyright 2000-2006 Vivotek, Inc. All rights reserved.

rtsps_ubuffer.h

Date:
2006/05/11

Author:
Rey Cheng

4.18 rtspserver.h File Reference

Handle RTSP request from client site.

```
#include "osisolate.h"
#include "typedef.h"
#include "common.h"
#include "sockdef.h"
#include "rtsprtpcommon.h"
#include "streamserver.h"
```

Data Structures

- struct RTSPSERVER_CLIENTIP
- struct RTSPSERVER_PARAM
- struct RTSPSERVER_SDPREQUEST
- struct RTSPSERVER_SESSIONINFORMATION
- struct THTTPCONNINFO

Defines

- #define RTSPSERVER_MEDIATYPE_AUDIOVIDEO 1
- #define **RTSPSERVER_MEDIATYPE_AUDIOONLY** 2
- #define **RTSPSERVER_MEDIATYPE_VIDEOONLY** 3
- #define **TCP_REQUEST_CS** 1
- #define **TCP_RELEASE_CS** 2
- #define **RTSP_HTTP_ADD_SINGLE** 1
- #define **RTSP_HTTP_ADD_PAIR** 2
- #define **RTSP_HTTP_COOKIE_LEN** 32
- #define **RTSP_HTTP_MESSAGE_LEN** 512
- #define **RTSP_AUTH_DISABLE** 0
- #define **RTSP_AUTH_BASIC** 1
- #define **RTSP_AUTH_DIGEST** 2
- #define RTSPSERVER_CALLBACKFLAG_ACCESSIP_CHECK 1
- #define RTSPSERVER_CALLBACKFLAG_SDP_REQUEST 2
- #define RTSPSERVER_CALLBACKFLAG_SESSION_START 3
- #define RTSPSERVER_CALLBACKFLAG_SESSION_STOP 4
- #define RTSPSERVER_CALLBACKFLAG_SESSION_PAUSE 5
- #define RTSPSERVER_CALLBACKFLAG_SESSION_RESUME 6
- #define RTSPSERVER_CALLBACKFLAG_SESSION_RTPUPDATE 7
- #define RTSPSERVER_CALLBACKFLAG_AUTHORIZATION 8
- #define RTSPSERVER_CALLBACKFLAG_FORCE_I_FRAME 9

- #define RTSPSERVER_CALLBACKFLAG_CHECK_VIDEO_TRACK 10
- #define RTSPSERVER_CALLBACKFLAG_CHECK_STREAM_MODE 11
- #define RTSPSERVER_CALLBACKFLAG_UPDATE_SESSIONINFO 12
- #define RTSPSERVER_CALLBACKFLAG_CHECK_ACCESSNAME 13
- #define RTSPSERVER_CALLBACKFLAG_UPLOAD_AUDIODATA 14
- #define RTSPSERVER_CALLBACKFLAG_UPLOAD_AUDIODATA_TIMEOUT 15
- #define RTSPSERVER_CALLBACKFLAG_SESSION_REMOVE_BACKCHANNEL 16

Typedefs

- typedef int(* RTSPSERVERCALLBACK)(HANDLE hParentHandle, UINT uMsgFlag, void *pvParam1, void *pvParam2)

  CallBack Function of RTSP server to Control Module.

Functions

- HANDLE RTSPServer_Create (int iMaxSessionNumber, RTSPSERVER_PARAM *pstRTSPServerParameter)

  Create handle of RTSP server.

- int RTSPServer_Start (HANDLE hRTSPServerHandle)

  Start RTSP server.

- int RTSPServer_Stop (HANDLE hRTSPServerHandle)

  Stop RTSP server.

- int RTSPServer_Close (HANDLE hRTSPServerHandle)

  close RTSP server.

- int RTSPServer_SetCallback (HANDLE hRTSPServerHandle, RTSPSERVERCALLBACK fCallback, HANDLE hParentHandle)

  Set CallBack Function to RTSP server.

- int RTSPServer_SetParameters (HANDLE hRTSPServerHandle, RTSPSERVER_PARAM *pstVideoEncodingParameter)

  Set RTSP server parameter.

- int RTSPServer_RemoveSession (HANDLE hRTSPServerHandle, DWORD dwSessionID)

  Remove one sesison from RTSP server.

- int RTSPServer_AddRTPOverHTTPSock (HANDLE hRTSPServerHandle, THTTPCONNINFO *ptHTTPConnInfo, DWORD dwFlag)

  Add RTP over HTTP socket to RTSP Server.

- int RTSPServer_AddTCPMuxHandle (HANDLE hRTSPServerHandle, HANDLE hTCPMuxCS)

  Add critical section handle to RTSP Server.

- int RTSPServer_GetCurrentSessionNumber (HANDLE hRTSPServerHandle)

  Get current session number from RTSP Server.

- void RTSPServer_TeardownSessionOK (HANDLE hRTSPServerHandle, DWORD dwSessionID)

  Report Session Tear down ok from Media Channel.

- int RTSPServer_SetAuthenticationType (HANDLE hRTSPServerHandle, int iAuthType)

  Set the authentication type of RTSP Server.

4.18.1 Detailed Description

Handle RTSP request from client site.

Copyright (c) 2002 Vivotek Inc. All rights reserved.

RTSPServerAPI.h

Date:
2002/05/20

Author:
ShengFu Cheng

4.18.2 Define Documentation

#define RTSPSERVER_CALLBACKFLAG_ACCESSIP_CHECK 1

Flag of CallBack (the authentication of client IP CallBack)

#define RTSPSERVER_CALLBACKFLAG_AUTHORIZATION 8

Flag of CallBack (the callback of authorization)

#define RTSPSERVER_CALLBACKFLAG_CHECK_ACCESSNAME 13

Check if the access name is correct

#define RTSPSERVER_CALLBACKFLAG_CHECK_STREAM_MODE 11

check video only, audio only, both of the stream

#define RTSPSERVER_CALLBACKFLAG_CHECK_VIDEO_TRACK 10

check if video track or not

#define RTSPSERVER_CALLBACKFLAG_FORCE_I_FRAME 9

Flag of Callback (the callback of request I frame

#define RTSPSERVER_CALLBACKFLAG_SDP_REQUEST 2

Flag of CallBack (the request of SDP CallBack)

#define RTSPSERVER_CALLBACKFLAG_SESSION_PAUSE 5

Flag of CallBack (the notification of session pause)

#define RTSPSERVER_CALLBACKFLAG_SESSION_REMOVE_BACKCHANNEL 16

Remove multicast back channel(when more than 1 clients still exist for this stream) Added by Louis to fix multicast bug!

#define RTSPSERVER_CALLBACKFLAG_SESSION_RESUME 6

Flag of CallBack (the notification of session resume)

#define RTSPSERVER_CALLBACKFLAG_SESSION_RTPUPDATE 7

Flag of CallBack (the notification of RTP session update)

#define RTSPSERVER_CALLBACKFLAG_SESSION_START 3

Flag of CallBack (the notification of session start)

#define RTSPSERVER_CALLBACKFLAG_SESSION_STOP 4

Flag of CallBack (the notification of session stop)

#define RTSPSERVER_CALLBACKFLAG_UPDATE_SESSIONINFO 12

Receive Updated Session Info

#define RTSPSERVER_CALLBACKFLAG_UPLOAD_AUDIODATA 14

Upload audio data for SIP-2-WAY

#define RTSPSERVER_CALLBACKFLAG_UPLOAD_AUDIODATA_TIMEOUT 15

Upload audio data timeout for SIP-2-WAY

#define RTSPSERVER_MEDIATYPE_AUDIOVIDEO 1

Constant to define and setup RTSP server for aggregate or non-aggregate control

4.18.3 Typedef Documentation

typedef int(∗ RTSPSERVERCALLBACK)(HANDLE hParentHandle, UINT uMsgFlag, void ∗ pvParam1, void ∗ pvParam2)

CallBack Function of RTSP server to Control Module.

Parameters:

hParentHandle *(i)* handle of control module which created RTSP server

uMsgFlag *(i)* the flag used to identify which action should take to the callback function

pvParam1 *(i/o)* the first parameter use to callback to control module

pvParam1 *(i/o)* the second parameter use to callback to control module

Note:

in uMsgFlag case : **RTSPSERVER_CALLBACKFLAG_ACCESSIP_CHECK**

pvParam1: pointer to the RTSPSERVER_CLIENTIP structure containing client site IP and port information to check or autheticate.

pvParam2: no used.

return value: 0 means this client IP has access right, others mean access deny

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_SDP_REQUEST

pvParam1: pinter to RTSPSERVER_SDPREQUEST structure containing Describe information.

pvParam2: pointer to the buffer where SDP information is copied to (memory allocated by RTSP server)

return value: 0 or negative value means ERROR, positive value means the length of sdp

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_SESSION_START

pvParam1: pointer to RTSPSERVER_SESSIONINFORMATIO structure containing the session information

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_SESSION_STOP

pvParam1: (DWORD)session ID of the terminated session.

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_SESSION_PAUSE

pvParam1: (DWORD)session ID of the paused session.

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_SESSION_RESUME pvParam1: (DWORD)session ID of the resumed session.

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_SESSION_RTPUPDATE

pvParam1: pointer to RTSPSERVER_SESSIONINFORMATIO structure containing the session information to be updated

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_AUTHORIZATION

pvParam1: pointer to TAuthorInfo structure containing the authentication information

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_FORCE_I_FRAME

pvParam1: pointer to RTSPSERVER_SESSIONINFORMATIO structure containing the session information to be updated

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_CHECK_VIDEO_TRACK

pvParam1: index of the stream to be callbacked to.

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_CHECK_STREAM_MODE

pvParam1: index of the stream to be callbacked to.

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_UPDATE_SESSIONINFO

pvParam1: pointer to buffer to store the session info

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_CHECK_ACCESSNAME

pvParam1: Access name string

pvParam2: index of the stream to be callbacked to.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_UPLOAD_AUDIODATA

pvParam1: pointer to PROTOCOL_MEDIABUFFER structure containing the audio info

pvParam2: no used.

return value: 0 is OK, negative value means error in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_UPLOAD_AUDIODATA_TIMEOUT

pvParam1: Session ID of Upload Stream Audio

pvParam2: no used.

return value: 0 is OK, negative value means error

in uMsgFlag case : RTSPSERVER_CALLBACKFLAG_SESSION_REMOVE_BACKCHANNEL

pvParam1: (DWORD)session ID of the back channel that wish to be removed.

pvParam2: no used.

return value: 0 is OK, negative value means error

4.18.4 Function Documentation

int RTSPServer_AddRTPOverHTTPSock (HANDLE *hRTSPServerHandle*, THTTPCONNINFO * *ptHTTPConnInfo*, DWORD *dwFlag*)

Add RTP over HTTP socket to RTSP Server.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

ptHTTPConnInfo *(i)* pointer to THTTPCONNINFO structure which contains HTTP connection info

dwFlag *(i)* Flag to indicate RTSP_HTTP_ADD_PAIR or RTSP_HTTP_ADD_SINGLE

Return values:

0 Add RTP over HTTP success

1 Add RTP over HTTP success failed

int RTSPServer_AddTCPMuxHandle (HANDLE *hRTSPServerHandle*, HANDLE *hTCPMuxCS*)

Add critical section handle to RTSP Server.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

hTCPMuxCS *(i)* handle of critical section

Return values:

0 Add critical section success

1 Add critical section failed

int RTSPServer_Close (HANDLE *hRTSPServerHandle*)

close RTSP server.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

Return values:

- ***0*** RTSP server resource free OK
- ***1*** RTSP server resource free failed

Note:

this API will release all the resource RTSP server take

HANDLE RTSPServer_Create (int *iMaxSessionNumber*, RTSPSERVER_PARAM ∗ *pstRTSPServerParameter*)

Create handle of RTSP server.

Parameters:

- ***iMaxSessionNumber*** *(i)* the maximum number of clients that RTSP server can accept
- ***pstRTSPServerParameter*** *(i)* a pointer to **RTSPSERVER_PARAM** for setting the parameter of RTSP server

Returns:

handle of RTSP server

int RTSPServer_GetCurrentSessionNumber (HANDLE *hRTSPServerHandle*)

Get current session number from RTSP Server.

Parameters:

- ***hRTSPServerHandle*** *(i)* the handle of RTSP server

Return values:

- ***Number*** of sessions

int RTSPServer_RemoveSession (HANDLE *hRTSPServerHandle*, DWORD *dwSessionID*)

Remove one sesison from RTSP server.

Parameters:

- ***hRTSPServerHandle*** *(i)* the handle of RTSP server
- ***dwSessionID*** *(i)* the session ID would be removed

Return values:

- ***0*** the removal of the session is queued
- ***1*** the removal of session failed

int RTSPServer_SetAuthenticationType (HANDLE *hRTSPServerHandle*, int *iAuthType*)

Set the authentication type of RTSP Server.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

iAuthType *(i)* Authentication type of RTSP (Disabled, basic, or digest)

Return values:

0 Set the authentication type of RTSP Server success

1 Set the authentication type of RTSP Server failed

int RTSPServer_SetCallback (HANDLE *hRTSPServerHandle*, RTSPSERVERCALLBACK *fCallback*, HANDLE *hParentHandle*)

Set CallBack Function to RTSP server.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

fCallback *(i)* the pointer to **RTSPSERVERCALLBACK** for assigning callback function to thr RTSP server

hParentHandle *(i)* the handle of the parent boject which created the RTSP server object

Return values:

0 set callback function to RTSP server OK

1 set callback function to RTSP server failed

int RTSPServer_SetParameters (HANDLE *hRTSPServerHandle*, RTSPSERVER_PARAM ∗ *pstVideoEncodingParameter*)

Set RTSP server parameter.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

pstVideoEncodingParameter *(i)* the pointer to **RTSPSERVER_PARAM** to set into RTSP server to thr RTSP server

Return values:

0 RTSP server parameter setting OK

1 RTSP server parameter setting failed

int RTSPServer_Start (HANDLE *hRTSPServerHandle*)

Start RTSP server.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

ulThreadPriority *(i)* the priority of RTSP server task

Return values:

0 RTSP server task start OK

1 RTSP server task start failed

int RTSPServer_Stop (HANDLE *hRTSPServerHandle*)

Stop RTSP server.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

Return values:

0 RTSP server task stop OK

1 RTSP server task stop failed

Note:

this API will make RTSP server stop the server main loop

void RTSPServer_TeardownSessionOK (HANDLE *hRTSPServerHandle*, DWORD *dwSessionID*)

Report Session Tear down ok from Media Channel.

Parameters:

hRTSPServerHandle *(i)* the handle of RTSP server

dwSessionID *(i)* Session ID of the session which has been teared down

Return values:

0 Report teared down success

1 Report teared down failed

4.19 rtspstreamingserver.h File Reference

Handle the RTSP request from RTSP client and send the media stream to client.

```
#include "rtsprtpcommon.h"
#include "osisolate.h"
#include "typedef.h"
#include "common.h"
#include "sockdef.h"
#include "bitstreambufdef.h"
#include "streamserver.h"
```

Data Structures

- struct TRTSPSTREAMING_AUDENCODING_PARAM
- struct TRTSPSTREAMING_DYNAMIC_PARAM
- struct TRTSPSTREAMING_PARAM
- struct TRTSPSTREAMING_VIDENCODING_PARAM

Defines

- #define RTSPSTREAMINGSERVER_VERSION MAKEFOURCC(1, 6, 0, 2)
- #define **RTSPSTREAMING_RTSPSERVER_MEDIATYPE_BASE** 0
- #define **RTSPSTREAMING_MEDIATYPE_AUDIOVIDEO** 1+RTSPSTREAMING_RTSPSERVER_MEDIATYPE_BASE
- #define **RTSPSTREAMING_MEDIATYPE_AUDIOONLY** 2+RTSPSTREAMING_RTSPSERVER_MEDIATYPE_BASE
- #define **RTSPSTREAMING_MEDIATYPE_VIDEOONLY** 3+RTSPSTREAMING_RTSPSERVER_MEDIATYPE_BASE
- #define **RTSPSTREAMING_VIDEO_PROLEVE** 0x00000001
- #define **RTSPSTREAMING_VIDEO_BITRATE** 0x00000002
- #define **RTSPSTREAMING_VIDEO_CLOCKRATE** 0x00000004
- #define **RTSPSTREAMING_VIDEO_MPEG4CI** 0x00000008
- #define **RTSPSTREAMING_VIDEO_WIDTH** 0x00000010
- #define **RTSPSTREAMING_VIDEO_HEIGHT** 0x00000020
- #define **RTSPSTREAMING_VIDEO_DECODEBUFF** 0x00000040
- #define **RTSPSTREAMING_VIDEO_SET_CI** 0x00000080
- #define **RTSPSTREAMING_AUDIO_BITRATE** 0x00000001
- #define **RTSPSTREAMING_AUDIO_CLOCKRATE** 0x00000002
- #define **RTSPSTREAMING_AUDIO_PACKETTIME** 0x00000004
- #define **RTSPSTREAMING_AUDIO_OCTECTALIGN** 0x00000008
- #define **RTSPSTREAMING_AUDIO_AMRCRC** 0x00000010
- #define **RTSPSTREAMING_AUDIO_ROBUSTSORT** 0x00000020
- #define **RTSPSTREAMING_AUDIO_CODECTYPE** 0x00000040
- #define **RTSPSTREAMING_AUDIO_SET_CI** 0x00000080
- #define **RTSPSTREAMING_ACCESSNAME_SETFLAG** 0x00000001
- #define **RTSPSTREAMING_MEDIAMODE_SETFLAG** 0x00000002
- #define **RTSPSTREAMING_RTSP_AUTHENTICATE_SETFLAG** 0x00000004
- #define **RTSPSTREAMING_TRACK_NAME_LEN** 20
- #define **RTSPSTREAMING_VIDEOENCODING_VIDEOSIGNA_BASE** 0
- #define **RTSPSTREAMING_VIDEOSIGNAL_NTSC** 0+RTSPSTREAMING_VIDEOENCODING_VIDEOSIGNA_BASE
- #define **RTSPSTREAMING_VIDEOSIGNAL_PAL** 1+RTSPSTREAMING_VIDEOENCODING_VIDEOSIGNA_BASE
- #define **RTSPSTREAMING_VIDEOENCODING_PICSIZE_BASE** 0
- #define **RTSPSTREAMING_PICSIZE_QSIF** 1+RTSPSTREAMING_VIDEOENCODING_PICSIZE_BASE
- #define **RTSPSTREAMING_PICSIZE_QCIF** 2+RTSPSTREAMING_VIDEOENCODING_PICSIZE_BASE
- #define **RTSPSTREAMING_PICSIZE_CIF** 3+RTSPSTREAMING_VIDEOENCODING_PICSIZE_BASE
- #define **RTSPSTREAMING_PICSIZE_SIF** 4+RTSPSTREAMING_VIDEOENCODING_PICSIZE_BASE

Enumerations

- enum ERTSPAudioCodecType { **ractGAMR** = 0, **ractG7221** = 1, **ractG711u** = 2, **ractAAC4** }

Functions

- int RTSPStreaming_GetVersion (BYTE ∗byMajor, BYTE ∗byMinor, BYTE ∗byBuild, BYTE ∗byRevision)

  Get the Version of RTSP Streaming server.

- HANDLE RTSPStreaming_Create (TRTSPSTREAMING_PARAM ∗pstRTSPStreamingParameter)

  Create **RTSPStreaming** *instance and initialize it.*

- int RTSPStreaming_Start (HANDLE hRTSPStreamingHandle)

  Start the **RTSPStreaming** *server.*

- int RTSPStreaming_Stop (HANDLE hRTSPStreamingHandle)

  Stop the **RTSPStreaming** *server.*

- int RTSPStreaming_Close (HANDLE ∗phRTSPStreamingHandle)

  Delete an instance of the **RTSPStreaming** *object.*

- int RTSPStreaming_SetDynamicParameters (HANDLE hRTSPStreamingHandle, TRTSPSTREAMING_DYNAMIC_PARAM ∗pstRTSPStreamingDynamicParam, DWORD dwSetFlag)

  Set the parameter of **RTSPStreaming** *server.*

- int RTSPStreaming_SetVideoParameters (HANDLE hRTSPStreamingHandle, int iSDPIndex, TRTSPSTREAMING_VIDENCODING_PARAM ∗pstVideoEncodingParameter, DWORD dwFlag)

  Set the Video parameter of **RTSPStreaming** *server.*

- int RTSPStreaming_SetAudioParameters (HANDLE hRTSPStreamingHandle, int iSDPIndex, TRTSPSTREAMING_AUDENCODING_PARAM ∗pstAudioEncodingParameter, DWORD dwFlag)

  Set the Audio parameter of **RTSPStreaming** *server.*

- SCODE RTSPStreaming_SetControlCallback (HANDLE hRTSPStreamingHandle, FControlChannel_Callback pfnCallback, HANDLE hParentHandle)

  Set the control callback function of **RTSPStreaming** *server.*

- SCODE RTSPStreaming_SetVideoCallback (HANDLE hRTSPStreamingHandle, MEDIA_CALLBACK pfnCallback, HANDLE hParentHandle)

  Set the Video callback function of **RTSPStreaming** *server.*

- SCODE RTSPStreaming_SetAudioCallback (HANDLE hRTSPStreamingHandle, MEDIA_CALLBACK pfnCallback, HANDLE hParentHandle)

  Set the Audio callback function of **RTSPStreaming** *server.*

- int RTSPStreaming_SetHostName (HANDLE hRTSPStreamingHandle, char ∗pcRTSPStreamingHostName)

  Set the host name of **RTSPStreaming** *server.*

- int RTSPStreaming_SetSDPETag (HANDLE hRTSPStreaming, char ∗pcSDPETag)

  Set the etag string in SDP of **RTSPStreaming** *server.*

- int RTSPStreaming_AddAccessList (HANDLE hRTSPStreamingHandle, unsigned long ulStartIP, unsigned long ulEndIP)

  Add the acceptable IP range of **RTSPStreaming** *server.*

- int RTSPStreaming_RemoveAccessList (HANDLE hRTSPStreamingHandle, unsigned long ulStartIP, unsigned long ulEndIP)

  Remove the acceptable IP range of **RTSPStreaming** *server.*

- int RTSPStreaming_AddDenyList (HANDLE hRTSPStreamingHandle, unsigned long ulStartIP, unsigned long ulEndIP)

  Add the rejective IP range of **RTSPStreaming** *server.*

- int RTSPStreaming_RemoveDenyList (HANDLE hRTSPStreamingHandle, unsigned long ulStartIP, unsigned long ulEndIP)

  Remove the rejective IP range of **RTSPStreaming** *server.*

- int **RTSPStreaming_SendLocation** (HANDLE hRTSPStreaming, char ∗pcLocation)
- int RTSPStreaming_ComposeAVSDP (HANDLE hRTSPStreaming, int iSDPIndex, unsigned long ulSDPIP, int iVivotekClient, int iMulticast, char ∗pcSDPBuffer, int iSDPBufferLen)

  compose SDP for http server.

- int **RTSPStreaming_ClearAccessList** (HANDLE hRTSPStreamingHandle)
- HANDLE **RTSPStreaming_GetIPAccessHandle** (HANDLE hRTSPStreaming)
- int **RTSPStreaming_SetMediaStreamMode** (HANDLE hRTSPStreaming, DWORD dwMediaStreamMode, int iCodecIndex)
- int **RTSPStreaming_AddRTPOverHTTPSock** (HANDLE hRTSPStreaming, TStreamServer_ConnectionSettings ∗ptConnectionSetting)
- int **RTSPStreaming_GetCurrentSessionNumber** (HANDLE hRTSPStreaming)
- int **RTSPStreaming_GetRTSPSessionInfo** (HANDLE hRTSPStreaming, char ∗pSessionInfoBuf, int iLength)

4.19.1 Detailed Description

Handle the RTSP request from RTSP client and send the media stream to client.

Copyright 2000-2002 Vivotek, Inc. All rights reserved.

RTSPStreamingAPI.h

Date:

2002/05/28

Author:

Simon Chen

4.19.2 Define Documentation

#define RTSPSTREAMINGSERVER_VERSION MAKEFOURCC(1, 6, 0, 2)

Revision history

Version 1.6.0.1, 2007/09/20, YenChun Updated in $streamingserver\rtspserver\src\parser.c

- RTSPServer_ParseURL() fix no access name will segmentation fail Updated in $streamingserver\rtspstreamingserver\src\rtspstreaming.c Updated in $streamingserver\rtspstreamingserver\inc\rtspstreamingserver.h
- RTSPStreaming_CreateMulticastSocket() modified to support Mulitcast TTL
- Update to Version 1.6.0.1

4.19.3 Enumeration Type Documentation

enum ERTSPAudioCodecType

Supported Audio Codec Type

4.19.4 Function Documentation

int RTSPStreaming_AddAccessList (HANDLE *hRTSPStreamingHandle*, unsigned long *ulStartIP*, unsigned long *ulEndIP*)

Add the acceptable IP range of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

ulStartIP *(i)* the start IP address of acceptable IP range in network order.

ulEndIP *(i)* the end IP address of acceptable IP range in network order.

Return values:

0 set acceptable IP range ok.

Others set acceptable IP range failed.

Note:

This function sets the acceptable IP range of RTSP client for RTSPStreaming server.

int RTSPStreaming_AddDenyList (HANDLE *hRTSPStreamingHandle*, unsigned long *ulStartIP*, unsigned long *ulEndIP*)

Add the rejective IP range of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

ulStartIP *(i)* the start IP address of rejective IP range in network order.

ulEndIP *(i)* the end IP address of rejective IP range in network order.

Return values:

0 set rejective IP range ok.

Others set rejective IP range failed.

Note:

This function sets the rejective IP range of RTSP client for RTSPStreaming server.

int RTSPStreaming_Close (HANDLE ∗ *phRTSPStreamingHandle*)

Delete an instance of the **RTSPStreaming** object.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

Return values:

0 delete the RSPStreaming server ok.

Others delete the RSPStreaming server failed.

Note:

After the instance is closed, the handle of this instance can’t be accessed.

See also:

RTSPStreaming_Create

int RTSPStreaming_ComposeAVSDP (HANDLE *hRTSPStreaming*, int *iSDPIndex*, unsigned long *ulSDPIP*, int *iVivotekClient*, int *iMulticast*, char ∗ *pcSDPBuffer*, int *iSDPBufferLen*)

compose SDP for http server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

ulSDPIP *(i)* the IP address of server. (if server is behind NAT and aware of public IP address, input the public IP address instead of private one

iVivotekClient *(i)* TURE means the SDP request is from vivotek client otherwise is FALSE.

iMulticast *(i)* TURE means requested SDP is for multicast, otherwise is FALSE.

ppcSDPBuffer *(i)* pointer of SDP buffer.

iSDPBufferLen *(i)* Length of SDP buffer.

Return values:

$>$ 0 the length of composed SDP .

Others compase SDP failed.

Note:

user needs to prepare the buffer to store SDP.

HANDLE RTSPStreaming_Create (TRTSPSTREAMING_PARAM * *pstRTSPStreamingParameter*)

Create **RTSPStreaming** instance and initialize it.

Parameters:

iMaxSessionNumber *(i)* Set the maximum acceptable number of RTSP client.

pstRTSPStreamingParameter *(i)* the pointer of data structure **RTSPSTREAMING_PARAM** for setting the initialization parameters of the **RTSPStreaming** instance.

Return values:

NULL Create **RTSPStreaming** object failed.

Others Create **RTSPStreaming** object ok. The return value is its handle.

Note:

This function should be called before using this instance.

See also:

RTSPStreaming_Close

int RTSPStreaming_GetVersion (BYTE * *byMajor*, BYTE * *byMinor*, BYTE * *byBuild*, BYTE * *byRevision*)

Get the Version of RTSP Streaming server.

Parameters:

byMajor *(o)* Pointer to byte to store the Major version of RTSP Streaming server.

byMinor *(o)* Pointer to byte to store the Minor version of RTSP Streaming server.

byBuild *(o)* Pointer to byte to store the Build version of RTSP Streaming server.

byRevision *(o)* Pointer to byte to store the Revision version of RTSP Streaming server.

int RTSPStreaming_RemoveAccessList (HANDLE *hRTSPStreamingHandle*, unsigned long *ulStartIP*, unsigned long *ulEndIP*)

Remove the acceptable IP range of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

ulStartIP *(i)* the start IP address of acceptable IP range in network order.

ulEndIP *(i)* the end IP address of acceptable IP range in network order.

Return values:

0 remove acceptable IP range ok.

Others remove acceptable IP range failed.

Note:

This function removes the acceptable IP range of RTSP client for RTSPStreaming server.

int RTSPStreaming_RemoveDenyList (HANDLE *hRTSPStreamingHandle*, unsigned long *ulStartIP*, unsigned long *ulEndIP*)

Remove the rejective IP range of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

ulStartIP *(i)* the start IP address of rejective IP range in network order.

ulEndIP *(i)* the end IP address of rejective IP range in network order.

Return values:

0 Remove rejective IP range ok.

Others Remove rejective IP range failed.

Note:

This function removes the rejective IP range of RTSP client for RTSPStreaming server.

SCODE RTSPStreaming_SetAudioCallback (HANDLE *hRTSPStreamingHandle*, MEDIA_CALLBACK *pfnCallback*, HANDLE *hParentHandle*)

Set the Audio callback function of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

pfnCallback *(i)* Callback function

hParentHandle *(i)* callback parent handle

Return values:

0 set the parameter of RSPStreaming server ok.

Others set the parameter of RSPStreaming server failed.

int RTSPStreaming_SetAudioParameters (HANDLE *hRTSPStreamingHandle*, int *iSDPIndex*, TRTSPSTREAMING_AUDENCODING_PARAM ∗ *pstAudioEncodingParameter*, DWORD *dwFlag*)

Set the Audio parameter of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

iSDPIndex *(i)* SDP Index of the audio

pstAudioEncodingParameter *(i)* the pointer of data structure TRTSPSTREAMING_AUDENCODING_PARAM for setting the parameters of the **RTSPStreaming** server instance.

Return values:

0 set the parameter of RSPStreaming server ok.

Others set the parameter of RSPStreaming server failed.

Note:

This function can change the parameter of RTSPStreaming server.

SCODE RTSPStreaming_SetControlCallback (HANDLE *hRTSPStreamingHandle*, FControlChannel_Callback *pfnCallback*, HANDLE *hParentHandle*)

Set the control callback function of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

pfnCallback *(i)* Callback function

hParentHandle *(i)* callback parent handle

Return values:

0 set the parameter of RSPStreaming server ok.

Others set the parameter of RSPStreaming server failed.

int RTSPStreaming_SetDynamicParameters (HANDLE *hRTSPStreamingHandle*, TRTSPSTREAMING_DYNAMIC_PARAM ∗ *pstRTSPStreamingDynamicParam*, DWORD *dwSetFlag*)

Set the parameter of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

pstRTSPStreamingParameter *(i)* the pointer of data structure **RTSPSTREAMING_PARAM** for setting the parameters of the **RTSPStreaming** server instance.

Return values:

0 set the parameter of RSPStreaming server ok.

Others set the parameter of RSPStreaming server failed.

Note:

This function can change the parameter of RTSPStreaming server.

int RTSPStreaming_SetHostName (HANDLE *hRTSPStreamingHandle*, char ∗ *pcRTSPStreamingHostName*)

Set the host name of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

pcRTSPStreamingHostName *(i)* the pointer of **char** for setting the host name of the **RTSPStreaming** server.

Return values:

0 set the host name of RSPStreaming server ok.

Others set the host name of RSPStreaming server failed.

Note:

This function can change the host name of RTSPStreaming server.

int RTSPStreaming_SetSDPETag (HANDLE *hRTSPStreaming*, char ∗ *pcSDPETag*)

Set the etag string in SDP of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle (i) Handle of RTSPStreaming object.

pcSDPETag (i) the pointer of **char** for setting the etag string in SDP of the **RTSPStreaming** server.

Return values:

0 set the etag string in SDP of RSPStreaming server ok.

Others set the etag string in SDP of RSPStreaming server failed.

Note:

This function can change the etag string in SDP of RTSPStreaming server. The etag in SDP is used to distinguish the version of session description. We use frimware version combined with MAC address as the etag string. The maximun length of wtag is 60 bytes.

SCODE RTSPStreaming_SetVideoCallback (HANDLE *hRTSPStreamingHandle*, MEDIA_CALLBACK *pfnCallback*, HANDLE *hParentHandle*)

Set the Video callback function of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle (i) Handle of RTSPStreaming object.

pfnCallback (i) Callback function

hParentHandle (i) callback parent handle

Return values:

0 set the parameter of RSPStreaming server ok.

Others set the parameter of RSPStreaming server failed.

int RTSPStreaming_SetVideoParameters (HANDLE *hRTSPStreamingHandle*, int *iSDPIndex*, TRTSPSTREAMING_VIDENCODING_PARAM ∗ *pstVideoEncodingParameter*, DWORD *dwFlag*)

Set the Video parameter of **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle (i) Handle of RTSPStreaming object.

iSDPIndex (i) SDP Index of the video

pstVideoEncodingParameter (i) the pointer of data structure TRTSPSTREAMING_VIDENCODING_PARAM for setting the parameters of the **RTSPStreaming** server instance.

Return values:

0 set the parameter of RSPStreaming server ok.

Others set the parameter of RSPStreaming server failed.

Note:

This function can change the parameter of RTSPStreaming server.

int RTSPStreaming_Start (HANDLE *hRTSPStreamingHandle*)

Start the **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

Return values:

0 Start the RSPStreaming server ok.

Others Start the RSPStreaming server failed.

Note:

After this function is called, RTSPStreaming server start to accept RTSP client request and send the media bitstream to client site.

See also:

RTSPStreaming_Stop

int RTSPStreaming_Stop (HANDLE *hRTSPStreamingHandle*)

Stop the **RTSPStreaming** server.

Parameters:

hRTSPStreamingHandle *(i)* Handle of RTSPStreaming object.

Return values:

0 Stop the RSPStreaming server ok.

Others Stop the RSPStreaming server failed.

Note:

After this function is called, RTSPStreaming server will be stopped.

See also:

RTSPStreaming_Start

4.20 vssdef.h File Reference

error code & definitions for Vivotek server script compiler The definitions should be consistent with farseerdef.h & netdef.h.

```
#include "netdef.h"
```

Defines

- #define **ERR_VSP_INVALID_EVENT** 0x80000001
- #define **ERR_VSP_INVALID_VERSION** 0x80000002
- #define **S_OK** 0
- #define **S_FAIL** (unsigned int)(-1)
- #define INVALID_NUM 0xFFFFFFFF

invalid ID number (for counting from zero).

- #define MAX_CAPTION_LEN 14

  max length of caption text.

- #define MAX_VERSION_LEN 39

  max length of firmware version.

- #define DATE_LEN 10

  date format length "yyyy/mm/dd".

- #define TIME_LEN 8

  time format length "hh:mm:ss".

- #define MAX_ACCOUNT_NUM 21

  max account number including root.

- #define MAX_MOTION_WIN_NUM 3

  max motion detection window numbers.

- #define MAX_PRESET_LOCATION_LEN 40

  max length of preset location.

- #define MAX_PRESET_LOCATION_NUM 20

  max preset location number.

- #define MAX_PATROL_LOCATION_NUM 40

  max patrol location number.

- #define MAX_PTZDRVNAME_LEN 40

  max length for ptz driver name.

- #define MAX_PTZDRV_NUM 20

  max external ptz drivers number.

- #define MAX_PTZDRVFILENAME_LEN 40

  max ptz driver file name len.

- #define **MAX_CAMERA_DWELLINGTIME** 255
- #define **MAX_CAMERA_PAN_SPEED** 5
- #define **MIN_CAMERA_PAN_SPEED** -5
- #define **MAX_CAMERA_TILT_SPEED** 5
- #define **MIN_CAMERA_TILT_SPEED** -5
- #define **MAX_CAMERA_AUTO_SPEED** 5
- #define **MAX_CAMERA_SPEED_LEVEL** 5
- #define **MIN_CAMERA_AUTO_SPEED** 1
- #define **MAX_CAMERA_ZOOM_SPEED** 5
- #define **MIN_CAMERA_ZOOM_SPEED** -5
- #define **MAX_CAMERA_FOCUS_SPEED** 5
- #define **MIN_CAMERA_FOCUS_SPEED** (-5)

- #define **MIN_CAMERA_PT_X** (-104)
- #define **MAX_CAMERA_PT_X** (104)
- #define **MIN_CAMERA_PT_Y** (-15)
- #define **MAX_CAMERA_PT_Y** (28)
- #define **MIN_CAMERA_Z** (-5)
- #define **MAX_CAMERA_Z** (5)
- #define **USER_DEFINED_HOME_ZOOM_POSITION** 39
- #define **ENABLE_SET_CAMERA_Z** (1<<16)
- #define **MAX_CAMERA_IRIS_LEVEL** 8
- #define **CAMERA_FOCUS_SLEEP_MSEC** 100
- #define **CAMERA_CMD_LEFT** 1
- #define **CAMERA_CMD_RIGHT** 2
- #define **CAMERA_CMD_FOCUSNEAR** 3
- #define **CAMERA_CMD_UP** 4
- #define **CAMERA_CMD_HOME** 5
- #define **CAMERA_CMD_FOCUSAUTO** 6
- #define **CAMERA_CMD_FOCUSFAR** 7
- #define **CAMERA_CMD_DOWN** 8
- #define **CAMERA_CMD_ZOOMIN** 9
- #define **CAMERA_CMD_ZOOMOUT** 10
- #define **CAMERA_CMD_STOP** 11
- #define **CAMERA_CMD_AUTOPAN** 12
- #define **CAMERA_CMD_AUTOPATROL** 13
- #define **CAMERA_CMD_GOTO** 14
- #define **CAMERA_CMD_RECALL** 15
- #define **CAMERA_CMD_CRUISE** 16
- #define **CAMERA_CMD_SEQUENCE** 17
- #define MAX_HTTP_PARAM_PAIR_NUM 256

  max parameter pair of HTTP request.

- #define MAX_MAIL_BODY_SIZE 256

  max message size in mail content.

- #define MAX_MAIL_ATTACHMENTS 4

  max attachments in mail.

- #define MAX_MSG_SIZE 256

  max message size usnig tcp.

- #define MAX_DI_NUM 1

  max number of digital input.

- #define MAX_DO_NUM 1

  max number of digital output.

- #define SVR_PAGE_DLL ”svrpage.dll”

  file name for server page dll.

- #define SVR_SCRIPT_DLL ”svrscript.dll”

  file name for server script dll.

- #define EXT_PTZDRV_DLL ”extptzdrv.dll”

  file name for external ptz driver.

- #define EXT_PTZDRV2_DLL ”extptzdrv2.dll”

  file name for second external ptz driver.

- #define EXT_PTZDRV_PATH ”/flash/extptzdrv”

  directory for place ptz driver.

- #define SETTING_PATH ”/flash/settings”

  directories for place data that will be cleared when restore system.

- #define CONFIG_INI ”config.ini”

  file name of configuration.

- #define SYSTEM_LOG ”system.log”

  file name of system log.

- #define DEFAULT_SCRIPTFILE ”script.vssx”

  file name of external script file default external file name.

- #define SNAP_PREFIX ”snap_”

  file name prefix of snapshot.

- #define SINGLE_SNAPSHOT ”video.jpg”

  file name of single snapshot.

- #define L3_PROGRAM_NAME ”flash.bin”

  file name of L3 flash program.

- #define ROOT_NAME ”root”

  root name.

- #define USE_DEFAULT_NTP_SERVER ”skip to invoke default server”

  default NTP server string.

- #define NTP_UPDATE_ONE_HOUR 1

  definitions for configuration settings.

- #define **NTP_UPDATE_ONE_DAY** 2
- #define **NTP_UPDATE_ONE_WEEK** 3
- #define **NTP_UPDATE_ONE_MONTH** 4
- #define **VIDEO_CODEC_MPEG4** 0
- #define **VIDEO_CODEC_MJPEG** 1
- #define **MAX_VIDEO_CODEC** 1
- #define **VIDEO_SIZE_HALF** 1
- #define **VIDEO_SIZE_HALFx2** 2
- #define **VIDEO_SIZE_NORMAL** 3
- #define **VIDEO_SIZE_NORMALx2** 4

- #define **VIDEO_SIZE_DOUBLE** 5
- #define **MAX_VIDEO_SIZE** 5
- #define **MIN_VIDEO_SIZE** 1
- #define **VIDEO_COLOR_MONO** 0
- #define **VIDEO_COLOR_COLOR** 1
- #define **MAX_VIDEO_COLOR** 1
- #define **VIDEO_QUALITY_FIX_BITRATE** 0
- #define **VIDEO_QUALITY_FIX_QUANT** 1
- #define **MAX_VIDEO_QUALITY** 1
- #define **MIN_VIDEO_QUANT** 1
- #define **MAX_VIDEO_QUANT** 5
- #define **MIN_VIDEO_BITRATE** 32000
- #define **MAX_VIDEO_BITRATE** 3000000
- #define **VIDEO_WB_AUTOWB** 0
- #define **VIDEO_WB_FIXED_INDOOR** 1
- #define **VIDEO_WB_FIXED_FLUORESCENT** 2
- #define **VIDEO_WB_FIXED_OUTDOOR** 3
- #define **MAX_VIDEO_WB** 3
- #define **VIDEO_MODULATION_NTSC** 0
- #define **VIDEO_MODULATION_PAL** 1
- #define **VIDEO_MODULATION_AUTO** 2
- #define **MAX_VIDEO_MODULATION** 2
- #define **AUDIO_TRANSMODE_UNKNOWN** -1
- #define **AUDIO_TRANSMODE_FULLDUPLEX** 0
- #define **AUDIO_TRANSMODE_HALFDUPLEX** 1
- #define **AUDIO_TRANSMODE_TALK** 2
- #define **AUDIO_TRANSMODE_LISTEN** 3
- #define **AUDIO_TRANSMODE_NONE** 4
- #define **MAX_AUDIO_TRANSMODE** 4
- #define **AUDIO_SOURCE_INTERNAL** 1
- #define **AUDIO_SOURCE_EXTERNAL** 0
- #define **MAX_AUDIO_SOURCE** 1
- #define **AUDIO_OPMODE_TOGGLE** 0
- #define **AUDIO_OPMODE_PRESS** 1
- #define **MAX_AUDIO_OPMODE** 1
- #define **WIRELESS_MODE_INFRASTRUCT** 0
- #define **WIRELESS_MODE_ADHOC** 1
- #define **MAX_WIRELESS_MODE** 1
- #define **MIN_WIRELESS_CHANNEL** 1
- #define **MAX_WIRELESS_CHANNEL** 11
- #define **WIRELESS_TXRATE_1M** 1
- #define **WIRELESS_TXRATE_2M** 2
- #define **WIRELESS_TXRATE_5p5M** 3
- #define **WIRELESS_TXRATE_11M** 4
- #define **WIRELESS_TXRATE_22M** 5
- #define **WIRELESS_PREAMBLE_LONG** 0
- #define **WIRELESS_PREAMBLE_SHORT** 1
- #define **MAX_WIRELESS_PREAMBLE** 1
- #define **WIRELESS_AUTHMODE_OPEN** 0
- #define **WIRELESS_AUTHMODE_SHARED** 1

- #define **WIRELESS_AUTHMODE_AUTO** 2
- #define **MAX_WIRELESS_AUTHMODE** 2
- #define **WIRELESS_KEYLENGTH_64** 1
- #define **WIRELESS_KEYLENGTH_128** 2
- #define **WIRELESS_KEYLENGTH_256** 3
- #define **WIRELESS_KEYFORMAT_HEX** 0
- #define **WIRELESS_KEYFORMAT_ASCII** 1
- #define **MAX_WIRELESS_KEYFORMAT** 1
- #define **MAX_WIRELESS_KEY_SELECT** 4
- #define **PTZ_ENABLED** 2
- #define **EXTPTZDRV_NONE** 128
- #define **EXTPTZDRV_CUSTOM_CAMERA** 129
- #define **MAX_ACCESSLIST_NUM** 10
- #define **WHITEBALANCE_AUTOTRACKING** 0
- #define **WHITEBALANCE_MANUALSET** 1
- #define **MAX_SNMP_COMMUN_LEN** 14
- #define **MAX_SNMP_LEN** 30
- #define **SNMP_VERSIONS_V1V2V3** 1
- #define **SNMP_VERSIONS_V1V2** 2
- #define **SNMP_VERSIONS_V3** 3
- #define **SNMP_AUTH_NONE** 0
- #define **SNMP_AUTH_MD5** 1
- #define **SNMP_AUTH_SHA** 2
- #define **VSSEVENT_PARAMTYPE_NETWORK** 1
- #define **VSSEVENT_PARAMTYPE_POWER** 2
- #define **NETWORK_NONE** 0
- #define **NETWORK_10M** 1
- #define **NETWORK_100M** 2
- #define **POWER_PWR1** 0x01
- #define **POWER_PWR2** 0x02
- #define MAX_CONNECT_NUM 10

  max allowed connections for streaming.

4.20.1 Detailed Description

error code & definitions for Vivotek server script compiler The definitions should be consistent with farseerdef.h & netdef.h.

Copyright 2000-2003 Vivotek, Inc. All rights reserved.

vssdef.h

Date:
2003/05/30

Author:
Joe Wu

Index