

iOS Player Software

API Reference

Issue: 1.0.0

Date: 2012.11.22

www.dokodemoterebi.com

About This Document

Purpose

This document describes the related product versions, intended audience and document contents.

Related Versions

The following table lists the product version related to this document.

Product Name	Version
iOS Player Software Development Kit (SDK)	V1.0.0

Intended Audience

This document describes the reference information developed based iOS Player, and it is intended for the programmers who meet the following requirements:

- Be familiar with C/C++/Objective-C programming language
- Be familiar with xcode / iphone SDK invoking

Organization

The document first summarizes the iOS Player API function types and their connections, and then describes the using method of each function in detail.

This document is organized as follows:

Chapter	Content
1 Overview	Describes the components and development environments of iOS Player SDK.
2 API Description	Provides an overview of iOS Player SDK API, and describes all the API functions in detail.
3 Data Structures	Describes the data structures used by iOS Player.

1 Overview

This chapter describes the components of iOS Player API, mainly including four components:

1. libFurtherNet is used to connect to Box, and receive AV data and the related commands.
2. libFtUtility is mainly used to connect to P2P server and give feedback to server.
3. libHttpSvrInterface realizes Http server.
4. This component realizes the synthesis of Transport stream files.

1.1 iOS SDK components

Component	Item	Description
API interface	CFtNet.h、FtUtility.h TsMuxerInterface.h HttpSvrInterface.h	API interface should be included in the user project.
Static library	libFurtherNet.a libTsMuxerInterface.a libFtUtility.a libHttpSvrInterface.a	Static library should be linked in the user project.
Example code	CFt_Net_Manager.mm	Compiled by xcode3.2.5.

1.2 Function list

1.2.1 libFurtherNet library functions

Function name	Description
ConnectP2PServer	Connect to P2P server.
SearchLanDevice	Searches Box in LAN.
RTSPProc	Connects to the Box's RTSP port
Connect2PortProc	Connects to the Box's command and AV ports.
NetAVLivePlay	Enables the AV data receiving thread.
CloseAVThread	Disables the AV data receiving thread
GetVideoFrame	Gets a video frame.
GetAudioFrame	Gets an audio frame.
GetVideoBufSize	Gets the size of video RingBuffer.
GetAudioBufSize	Gets the size of audio RingBuffer.
GetBandWidth	Gets the current network bandwidth.
GetVideoInfor	Gets the video property.
ClearBuffer	Clears the audio and video buffers.
SetAvTaskType	Sets the task type.

NetCmdSetup	Enables the command receiving thread.
CloseCmdThread	Disables the command receiving thread.
SndLoginCmd	Login Box
SetTunnerUpDown	Switches channel up and down.
SetTunnerChn	Switches channel according to the input channel number.
GetTunnerChn	Gets the current channel number.
SndSetSource	Switches video source.
SndGetFrqTable	Gets the channel table.
SendSetAvAttr	Sets the video property.
SndIRRemoteEnter	Sends IR commands.
GetLogInState	Gets the login state.
GetCmdError	Gets the error code.
GetNetState	Gets the network state.
GetBoxInfor	Gets the information of Box.
SetPlayerType	Sets the Box's model number (A2, A5 or A7).
GetUserManagerInfor	Gets the information of the current online users.
CancelPreRecord	Cancel the schedule recording task or not when it arrives at the appointed time.
GetSourceInfor	Gets the video source information.
SndBoxRecUpdateInfoCmd	Updates the schedule recording tasks.
SndBoxRecAddTaskCmd	Adds a schedule recording task.
SndBoxRecDelTaskCmd	Deletes the schedule recording task.
SndBoxRecModifyTaskCmd	Edits the schedule recording task.
GetBoxRecInfor	Gets the information of schedule recording tasks.
HandleMsg	Callback function of the processing command.
HandleError	Callback function of the error data processing command.

1.2.2 libFtUtility library functions

Function name	Description
InitUserRequest	Initializes the system member variables.
RequestDvsFromCms	Sends request to CMS server.
AckLinkDvsStoP2PSvr	Returns with the connection states to P2P server.
UpdateUsrPassword	Modify the user password to CMS server.
GetCmsAnddefP2pIp	Changes the IP address of CMS and P2P server.
FurMiniRequestP2pServer	Gets T1 information from DNS server.
FurInitMiniSystemClienInfo	Initializes the data information of client.
FurMiniGetA2DvsInfor	Gets T2 information from T1.
FurGetMiniDefSboxAddress	Gets T1 IP address.
FurChgTvtable	Gets the channel table.
FurAckUserLinkA2St	Provides feedback when login or logout or failed to connect to T2.
CalculateTwoSBoxID	Calculates the ID of main T1 and Backup T1.

1.2.3 libHttpSvrInterface library functions

Function name	Description
CreateHttpServer	Creates the Sock of Http server.
startHttpServer	Starts the Http server.
stopHttpServer	Shuts down the Http server.
SetSegmentDuration	Sets TS (Transport Stream) duration.
GetSequence	Gets the current TS number.

1.2.4 libTsMuxerInterface

Function name	Description
TSWriterInit	Initializes the TS file.
TSWriterUninit	Uninitializes the TS file.
WriteTSProgramData	Writes the TS program information.
WriteTSPacketWithPcr	Writes the reference time of TS file.
WriteTSPacketWithH264	Writes the video streaming of TS file.
WriteTSPacketWithAAC	Writes the audio streaming of TS file.

2 API Description

2.1 libFurtherNet library function

2.1.1 (int) ConnectP2PServer:(char *)inBoxSn BoxIP:(char*)outBoxIP Port:(int*)outBoxPort;

Purpose:	Connects to P2P Server and gets the Box's IP and RTSP port from P2P server.
Parameter:	Input: inBoxSn: Box's ID number
	Output: outBoxIP: Box's IP address; outBoxPort: Box's RTSP port
Return Value:	1: Gets the Box's IP and RTSP port successfully; -1: Failed to connect to P2P server; -2: Box is offline; -3: ID is wrong.

2.1.2 (BOOL) SearchLanDevice:(char *)inBoxSn BoxIP:(char*)outBoxIP Port:(int*)outBoxPort;

Purpose:	Searches Box in LAN.
Parameter:	Input: inBoxSn: Box's ID
	Output: outBoxIP: Box's IP address; outBoxPort: Box's RTSP port;
Return Value:	YES: Gets the Box's IP and RTSP port successfully; NO: Can't find any Box in LAN.

2.1.3 (BOOL) RTSPProc:(const char*)inBoxIP Port:(unsigned short)inPort CmdPort:(unsigned short*)outCmdPort AVPort:(unsigned short *)outAVPort NetType:(int)inNetType;

Purpose:	Connects to the Box's RTSP port and gets the command and AV ports.
Parameter:	Input: inBoxIP: Box's IP address; import: Box's RTSP port; inNetType: 1: WAN connection; 0: LAN connection.
	Output: outCmdPort: Box's command port; outAVPort: Box's AV port.
Return Value:	YES: Gets the Box's command and AV ports successfully; NO: Can't find any Box in LAN.

2.1.4 (BOOL) Connect2PortProc:(int*)outAVSock CmdSock:(int*)outCmdSock AVPort:(unsigned short)inAVPort CmdPort:(unsigned short)inCmdPort;

Purpose:	Connects to the Box's command and AV ports.
Parameter:	Input: inCmdPort: Box's command port; inAVPort: Box's AV port; Output: outAVSock: AV port Sock of the connected Box; outCmdSock: Command port Sock of the connected Box.
Return Value:	YES: Connects to the Box's command and AV ports successfully.

2.1.5 (BOOL) NetAVLivePlay:(int*)inAVSock;

Purpose:	Enables the AV data receiving thread, the data will be written into AV RingBuffer.
Parameter:	Input: inAVSock: AV port Sock of the connected Box. Output: NULL
Return Value:	YES: Enables the AV data receiving thread successfully; NO: Failed to enable the AV data receiving thread.

2.1.6 (BOOL) CloseAVThread;

Purpose:	Disables the AV receiving thread and stops writing data into AV RingBuffer.
Parameter:	Input: NULL Output: NULL
Return Value:	YES: Disables the AV data receiving thread successfully; NO: Failed to disable the AV data receiving thread.

2.1.7 (BOOL) GetVideoFrame:(char*)outData FrameLen:(unsigned int*)outLen TimeStamp:(int*)outPts NaluType:(char*)outNaluType;

Purpose:	Gets a video frame, a complete frame with H.264 raw data.
Parameter:	Input: NULL Output: outData: Video data; outLen: The length of video data; outPts: Time Mark of video frame; outNaluType: Video frame type.
Return Value:	YES: Gets a video frame successfully; NO: Failed to get a video frame.

2.1.8 (BOOL) GetAudioFrame:(char*)outData FrameLen:(unsigned int*)outLen TimeStamp:(int*)outPts;

Purpose:	Gets an audio frame, a complete frame with AAC raw data.
Parameter:	Input: NULL
	Output: outData: Audio data; outLen: The Length of audio data; outPts: Time Mark of audio data.
Return Value:	YES: Gets an audio frame successfully; NO: Failed to get an audio frame.

2.1.9 (int) GetVideoBufSize;

Purpose:	Gets the valid data size of video buffer (Byte unit).
Parameter:	Input: NULL
	Output: NULL
Return Value:	Video Buffer: The size of current valid data.

2.1.10 (int) GetAudioBufSize;

Purpose:	Gets the valid data size of audio buffer (Byte unit).
Parameter:	Input: NULL
	Output: NULL
Return Value:	Audio Buffer: The size of current valid data.

2.1.11 (int) GetBandWidth;

Purpose:	Gets the current bitrate (KBPS unit).
Parameter:	Input: NULL
	Output: NULL
Return Value:	The current bitrate

2.1.12 (int) GetVideoInfor:(int*)outChannelId PalNtsc:(int*)outPalNtsc Source:(int*)outSource;

Purpose:	Gets the video property.
Parameter:	Input: NULL
	Output: outChannelId: Video channel number; outPalNtsc: TV system; outsource: Video source (TV, AV, CAM).
Return Value:	1: Successfully; 0: Failed.

2.1.13 (void) ClearBuffer;

Purpose:	Clears the audio and video buffers.
Parameter:	Input: NULL
	Output: NULL
Return Value:	NULL

2.1.14 (void) SetAvTaskType:(AvTaskType)inType;

Purpose:	Sets the task type (Real-time watching, Schedule recording).
Parameter:	Input: inType: Task type.
	Output: NULL
Return Value:	NULL

2.1.15 (BOOL) NetCmdSetup:(int*)inCmdSock;

Purpose:	Enables the command receiving thread.
Parameter:	Input: inCmdSock: Command port Sock of the connected Box;
	Output: NULL
Return Value:	YES: Enables the command receiving thread successfully; NO: Failed to enable the command receiving thread.

2.1.16 (BOOL) CloseCmdThread;

Purpose:	Disables the command receiving thread.
Parameter:	Input: NULL
	Output: NULL
Return Value:	YES: Disables the command receiving thread successfully; NO: Failed to disable the command receiving thread.

2.1.17 (BOOL) SndLoginCmd:(struct Login_Pm *)inLoginPam;

Purpose:	Sends login command.
Parameter:	Input: inLoginPam: Login command message, please refer to structure Login_Pm.
	Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.18 (int) SetTunnerUpDown:(BOOL)inUpDown;

Purpose:	Switches channel up and down.
Parameter:	Input: inUpDown: YES: Channel up; NO: Channel down. Output: NULL
Return Value:	Channel number after switch.

2.1.19 (BOOL) SetTunnerChn:(int)inChnNum;

Purpose:	Switches channel according to the input channel number.
Parameter:	Input: inChnNum: The destination channel number. Output: NULL
Return Value:	Channel number after switch.

2.1.20 (int) GetTunnerChn:(int*)inChnNum Frq:(int*)inFrq;

Purpose:	Gets the current channel number.
Parameter:	Input: NULL Output: inChnNum: The current channel number.; inFrq: The frequency of the current channel.
Return Value:	The current channel number.

2.1.21 (BOOL) SndSetSource:(CMD_SETSOURCE*)inSource;

Purpose:	Switches video source.
Parameter:	Input: inSource: The destination video source. Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.22 (BOOL) SndGetFrqTable;

Purpose:	Gets the channel table.
Parameter:	Input: NULL Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.24 (BOOL) SendSetAvAttr:(int) inResolution Quality:(int)inQuality;

Purpose:	Sets the video property.
Parameter:	Input: Resolution: 0,1,2; inQuality: Resolution 0-9 Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.25 (BOOL) SndIRRemoteEnter:(int)inPressdCode;

Purpose:	Sends IR commands.
Parameter:	Input: inPressdCode: IR code Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.26 (void) GetLogInState:(LogInState *)outState;

Purpose:	Gets the login state.
Parameter:	Input: NULL Output: outState: Login state
Return Value:	NULL

2.1.27 (int) GetCmdError;

Purpose:	Gets the error code.
Parameter:	Input: NULL Output: NULL
Return Value:	Error code.

2.1.28 (BOOL) GetNetState:(LogInState*)outLogInState;

Purpose:	Gets the network state.
Parameter:	Input: NULL Output: outLogInState: Network state.
Return Value:	YES: Successful; NO: Failed.

2.1.29 (BOOL) GetBoxInfor;

Purpose:	Gets the information of Box.
Parameter:	Input: NULL Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.30 (void) SetPlayerType:(PlayerType)inPlayerType;

Purpose:	Sets the Box's model number (A2, A5 or A7).
Parameter:	Input: inPlayerType: Box's model number. Output: NULL
Return Value:	NULL

2.1.31 (void) GetUserManagerInfor:(USER_MANAGER*)outUserManager;

Purpose:	Gets the information of the current online users.
Parameter:	Input: NULL Output: outUserManager: Information of the current online users.
Return Value:	NULL

2.1.32 (void) CancelPreRecord:(int)isCancel;

Purpose:	Cancel the schedule recording task or not when it arrives at the appointed time.
Parameter:	Input: isCancel: 1: Cancel; 0: Don't cancel. Output: NULL
Return Value:	NULL

2.1.33 (void) GetSourceInfor:(CMD_SETSOURCE*)outSource;

Purpose:	Gets the video source information.
Parameter:	Input: NULL Output: outsource: The video source information.
Return Value:	NULL

2.1.34 (void) SndGetIRTable;

Purpose:	Gets IR learning table.
Parameter:	Input: NULL
	Output: NULL
Return Value:	NULL

2.1.35 (BOOL) SndBoxRecUpdateInfoCmd;

Purpose:	Updates the schedule recording tasks.
Parameter:	Input: NULL
	Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.36 (BOOL) SndBoxRecAddTaskCmd:(char*) pTask Len:(unsigned int) uLen;

Purpose:	Adds a schedule recording task.
Parameter:	Input: pTask: The information of schedule recording task; uLen: The data size.
	Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.37 (BOOL) SndBoxRecDelTaskCmd:(unsigned int) uTaskId;

Purpose:	Deletes the schedule recording task.
Parameter:	Input: uTaskId: ID of the recording task to delete.
	Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.38 (BOOL) SndBoxRecModifyTaskCmd:(char*) pTask Len:(unsigned int) uLen;

Purpose:	Edits the schedule recording task.
Parameter:	Input: pTask: The information of schedule recording task; uLen: The data size.
	Output: NULL
Return Value:	YES: Successful; NO: Failed.

2.1.39 (BOOL) GetBoxRecInfor:(RECORDDATABLOCK*)outRecInfor;

Purpose:	Gets the information of schedule recording tasks.
Parameter:	Input: NULL
	Output: outRecInfor: The information of schedule recording task.
Return Value:	YES: Successful; NO: Failed.

2.1.40 (int) GetRecTaskNewId;

Purpose:	Gets the ID of the new schedule recording task.
Parameter:	Input: NULL
	Output: NULL
Return Value:	ID of the new schedule recording task.

2.1.41 (NSMutableDictionary*) GetFrqTable;

Purpose:	Gets the channel table.
Parameter:	Input: NULL
	Output: NULL
Return Value:	Channel table.

2.1.42 (void) HandleMsg:(int)Msg Data:(char*)outData Len:(int)outLen;

Purpose:	Callback function of the processing command, used to give feedback to UI for different processing.
Parameter:	Input: NULL
	Output: MSG: Command type; outData: Return data of the command; outLen: The length of return data.
Return Value:	NULL

2.1.43 (void) HandleError:(int)Error Data:(char*)outData Len:(int)outLen;

Purpose:	Callback function of the error data processing command, used to give feedback to UI for different processing.
Parameter:	Input: NULL
	Output: Error: Data type; outData: Return data of the command; outLen: The length of return data.
Return Value:	NULL

2.2 libFtUtility function description

2.2.1 (int) InitUserRequest:(char *)p_CmsIpM CMSIP:(char *)p_CmsIpS UserId:(char *)p_UsrId

Password:(char *)p_UsrPsd DynSvr:(char *)p_DynSvrIpM DynSvrIpS:(char *)p_DynSvrIpS

dvsVersion:(int)n_dvsverion p2pAutolpM:(char *)p_p2pAutolpM p2pAutolpS:(char *)p_p2pAutolpS;

Purpose:	Initializes the system member variables.
Parameter:	1.p_CmsIpM: IP address of the main CMS server. 2.p_CmsIpS: IP address of the backup CMS server. 3.p_UsrId: User account. 4.p_UsrPsd: User password. 5.p_DynSvrIpM: IP address of the main top CMS server. 6.p_DynSvrIpS: IP address of the main backup CMS server. 7:n_dvsversion: System or pc version flag. System version:0; Standalone version:1 8:p_p2pAutolpM: The default IP address of main P2P server. 9:p_p2pAutolpS: The default IP address of backup P2P server.
Return Value:	1: Successful; -1: Failed.

2.2.2 (int) RequestDvsFromCms:(char*)pRemoterDvsIP

DvsRtspPort:(int*)pRemoterDvsRtspPort DvsUserPurview:(int *)pDvsUserPurview;

Purpose:	Sends request to CMS server.
Parameter:	1.p_RemoterDvsIP: The return IP of available Box. 2.p_RemoterDvsRtspPort: The return RTSP port of available Box. 3.p_DvsUserPurview: The return private or public authority of Box type.
Return Value:	1: Get the Box information from P2P server successfully. FUR_SVR_ACK_SVR_BUSY: Failed to connect to server. FUR_P2P_ACK_NULLDVS: The corresponding Box hasn't registered with server. FUR_P2P_ACK_OFFUSE: The Box is disabled. FUR_SVR_ACK_DVS_BUSY: Can't connect to Box. FUR_SVR_USRPASSWORD_ER: The password is wrong. FUR_SVR_ACK_UNREGIST: The Box ID has not been registered. FUR_SVR_ACK_NULLSVR: There's no server for distribution. FUR_SVR_ACK_NULLDVS_CMS: Can't find CMS server in the top CMS server.

2.2.3 (int) AckLinkDvsStoP2PSvr:(int) mlinkStu;

Purpose:	Returns with the connection states to P2P server.
Parameter:	1.mlinkStu: Successful or failed connection state indicator, 0: Failed to connect; 1: Connects successfully.
Return Value:	1: Gives feedback successfully; -1: Failed to give feedback.

2.2.4 (int) UpdateUsrPassword:(char *)p_UsrName OldPsd:(char *)p_OldPsd NewPsd:(char *)p_NewPsd;

Purpose:	Modify the user password to CMS server.
Parameter:	p_UsrName: User name p_OldPsd: Old password p_NewPsd: New password
Return Value:	FUR_SVR_USRPASSWORD_ER: The password is wrong. FUR_SVR_ACK_UNREGIST: The Box ID has not been registered. FUR_CMS_MIDFY_FAIL: Failed to modify password. FUR_CMS_MIDFY_OK: Modify password successfully. FUR_SVR_USER_TYPE_ER: The client ID is incorrect. FUR_SVR_ACK_SVR_BUSY: Failed to connect to server.

2.2.5 (int) GetCmsAnddefP2plp:(int *)p_ntype cms_m_ip:(char *)p_cms_m_ip cms_s_ip:(char *)p_cms_s_ip m_defp2p:(char *)p_m_defp2p s_defp2p:(char)p_s_defp2p;

Purpose:	Changes the IP address of CMS and P2P server.
Parameter:	p_ntype: Output parameter, 1: Changes the CMS server IP; 2: Changes the P2P server IP; 3: Changes both the CMS and P2P server IP. p_cms_m_ip: Output parameter, new IP of the main CMS server. p_cms_s_ip: Output parameter, new IP of the backup CMS server. p_m_defp2p: Output parameter, new IP of the main P2P server. p_s_defp2p: Output parameter, new IP of the backup P2P server.
Return Value:	1: Successful; 0: Failed.

2.2.6 (int) FurMiniRequestP2pServer:(char *)p_MainSvrip SecondIP:(char)p_SecdSvrip UserSN:(char *)p_UsrAttestSN OutDvsIp:(char *)p_OutDvsIP OutDvsName:(char *)p_OutDvsName;

Purpose:	Gets T1 information from DNS server.
Parameter:	1.p_MainSvrip: IP address of main DNS server. 2.p_SecdSvrip: IP address of backup DNS server.

3.p_UsrAttestSN: The client account ID.
4.p_OutDvsIP: T1 IP address.
5.p_OutDvsName: Return account ID, used to check the input ID and return ID.

Return Value: 1: Successful; 0: Failed.

2.2.7 (int) FurlInitMiniSystemClientInfo:(char *)p_MySerNum Password:(const char *)p_MyPassword SboxIp:(const char *)p_DefSboxIpM SboxIp2:(const char *)p_DefSboxIpS DnsIp:(const char *)p_DnsIpM DnsIp2:(const char *)p_DnsIpS;

Purpose: Initializes the data information of client.

Parameter: 1.p_MySerNum: The account ID.
2.p_MyPassword: Account password.
3.p_DefSboxIpM: The default IP address of main T1.
4.p_DefSboxIpS: The default IP address of backup T1.
5.p_DnsIpM: The default IP address of main DNS server.
6.p_DnsIpS: The default IP address of main DNS server.

Return Value: 1: Successful; 0: Failed.

2.2.8 (int) FurMiniGetA2DvsInfor:(char *)p_RemoterDvsIP OutRtspPort:(int *)p_RemoterDvsRtspPort OutPassword:(char *)p_DnyAtestPsd OutTunerId:(int *)p_TunerId;

Purpose: Gets T2 information from T1.

Parameter: 1.p_RemoterDvsIP: Output parameter, T2 IP address;
2.p_RemoterDvsRtspPort: Output parameter, T2 RTSP port;
3.p_DnyAtestPsd: Output parameter, T2 register dynamic password.
4.p_TunerId: Output parameter, T2 Tuner ID number,

Return Value: FUR_MINI_SBOX_ACK_SUCCESS: T1 verifies the account and returns with T2 information.
FUR_MINI_SBOX_ACK_IDERROR: T1 responds that the specified T1 ID is wrong.
FUR_MINI_SBOX_ACK_USER_IDERROR: T1 responds that the client ID is wrong.
FUR_MINI_SBOX_ACK_DISABLE: T1 responds that the client account has been disabled.
FUR_MINI_SBOX_ACK_PSDERROR: T1 responds that the client password is wrong.
FUR_MINI_SBOX_ACK_NOFREE_A2: T1 responds that there's no free T2 for distribution.
FUR_MINI_SBOX_ACK_SERVER_BUSY: T1 responds that server is busy.
FUR_MINI_SBOX_ACK_ACCOUNT_REPEAT: T1 responds that the request account is reused.
FUR_MINI_SBOX_ACK_ACCOUNT_MATURITY: T1 responds that the permission of the client account has expired.

2.2.9 (int) FurGetMiniDefSboxAddress:(char *)p_outDefSboxIpM OutSboxIp2:(char *)p_outDefSboxIpS;

Purpose:	Gets T1 IP address.
Parameter:	1.p_outDefSboxIpM: Output parameter, the default IP of main T1 after change. 2.p_outDefSboxIpS: Output parameter, the default IP of backup T1 after change.
Return Value:	1: Successful; -1: Failed.

2.2.10 (int) FurChgTvtable:(const int) m_inTvCount;

Purpose:	Gets the channel table.
Parameter:	1.m_inTvCount: Input parameter, the destination channel number of switch.
Return Value:	The destination channel number of switch..

2.2.11 (int) FurAckUserLinkA2St:(int) n_inlinkStue;

Purpose:	Provides feedback when login or logout or failed to connect to T2.
Parameter:	1.m_LinkSt: Input parameter, 2: Quits the connection with T2; 1: Successful to connect to T2; -1: Failed to connect to T2.
Return Value:	1: Successful; -1: Failed.

2.2.12 (int) CalculateTwoSBoxID:(char*) plnAccID OutID:(char*) pOutSboxID1 OutID2:(char*) pOutSboxID2;

Purpose:	Calculates the ID of main T1 and Backup T1.
Parameter:	1. plnAccID: Input parameter, ID number with 10 characters. 2. pOutSboxID1: Output parameter, ID number with 10 characters. 3. pOutSboxID2: Output parameter, ID number with 10 characters.
Return Value:	1: Successful; -1: Failed.

2.3 libHttpSvrInterface function description

2.3.1 (int) CreateHttpServer:(unsigned short) inPort Socket:(int*)outSocket;

Purpose:	Creates the Sock of Http server.
Parameter:	Input: import: Http server port
	Output: outSocket: Sock of the Http server
Return Value:	1: Successful; 0: Failed.

2.3.2 (void) startHttpServer: (char *)strRoot Port: (int)nPort Socket:(int)inSocket SegmentDuration:(int)inSegmentDuration Seq:(int)inSeq irstIndex:(int)inFirsIndex;

Purpose:	Starts the Http server.
Parameter:	Input: strRoot: Root directory of the Http server; nPort: Server port; inSocket: Sock of the Http server; inSegmentDuration: TS duration (second unit); inSeq: TS file number; inFirsIndex: 0: Starts from the first one; 1: Starts from the third one.
	Output: NULL
Return Value:	1: Successful; 0: Failed.

2.3.3 (void) stopHttpServer;

Purpose:	Starts the Http server.
Parameter:	Input: NULL
	Output: NULL
Return Value:	NULL

2.3.4 (void) SetSegmentDuration:(int) inSegmentDuration;

Purpose:	Sets TS duration.
Parameter:	Input: inSegmentDuration: TS Duration
	Output: NULL
Return Value:	NULL

2.3.5 (int) GetSequence;

Purpose:	Gets the current TS number.
Parameter:	Input: NULL
	Output: NULL
Return Value:	The current TS number.

2.4 libTsMuxerInterface function description

2.4.1 (int) TSWriterInit:(TS_writer_p *)ts_writer_p fileName:(char *)filename;

Purpose:	Initializes the TS file.
Parameter:	Input: TS_writer_p: Pointer to TS file; filename: TS file path. Output: NULL
Return Value:	1: Successful; 0: Failed.

2.4.2 (int) TSWriterUninit:(TS_writer_p)ts_writer_p;

Purpose:	Uninitializes the TS file.
Parameter:	Input: TS_writer_p: Pointer to TS file. Output: NULL
Return Value:	1: Successful; 0: Failed.

2.4.3 (int) WriteTSProgramData:(TS_writer_p) ts_writer_p StreamId:(uint32_t)transport_stream_id ProgramNumber:(uint32_t)program_number PmtPid:(uint32_t)pmt_pid PcrPid:(uint32_t)pcr_pid NumProgs:(int)num_progs ProgPids:(uint32_t*)prog_pids ProgType:(byte*)prog_type StreamType:(int)inStreamType;

Purpose:	Writes the TS program information.
Parameter:	Input: ts_writer_p: Pointer to TS file; transport_stream_id: TS ID; program_number:Program number; pmt_pid PmtID; pcr_pid pcrID; num_progs numProg; prog_pids prog_type: Please refer to Demo. Output: NULL
Return Value:	1: Successful; 0: Failed. /* * TS program information * StreamType: 0:audio and video; 1:only audio; 2:only video */

2.4.4 (int) WriteTSPacketWithPcr:(TS_writer_p) ts_writer_p Pid:(uint32_t) pid StreamId:(byte) stream_id

Purpose:	Writes the reference time of TS file.
Parameter:	Input: TS_writer_p: Pointer to TS file. Output: NULL
Return Value:	1: Successful; 0: Failed. GotPcr:(int) got_pcr PcrBase:(uint64_t) pcr_base PcrExtn:(uint32_t) pcr_extn;

2.4.5 (int) WriteTSPacketWithH264:(char *)inData Len:(uint32_t) inlen Isldr:(int) isldr TsWriter:(TS_writer_p) ts_writer_p VideoPid:(uint32_t) videoPid Pts:(uint64_t) pts;

Purpose:	Writes the video streaming of TS file.
Parameter:	Input: TS_writer_p: Pointer to TS file. Output: NULL
Return Value:	1: Successful; 0: Failed.

2.4.6 (int) WriteTSPacketWithAAC:(char *)inData Len:(uint32_t) inlen TsWriter:(TS_writer_p) ts_writer_p audioPid:(uint32_t) audioPid Pts:(uint64_t) pts;

Purpose:	Writes the audio streaming of TS file.
Parameter:	Input: TS_writer_p: Pointer to TS file. Output: NULL
Return Value:	1: Successful; 0: Failed.

3 Data Structures

3.1 User information structure

```
typedef struct usermanager
{
    char user_id;           // User ID
    char user_level;        // User grade, the default is 0.
    char user_name[33];     // User name
    char password[33];      // User password
    char net_type;          // Network type: intranet: 0; internet: 1
    char video_source;      // Video source: CV: 0; SV: 1; TV: 2
    char client_type;       // 0: Web; 1: AP; 2: phone
    char video_size;        // P: 0 – 4; N: 5- 9
    unsigned int country;   // Country
    int tv_channel;         // Channel number
    int tv_fre;             // Frequency
    char cur_user_num;      // Online users number
    char audio_br;          // Audio bitrate
    char rand_port;         // 0: not rand port; 1: rand port
    char cancel_rec;        // 0: first; 1: stop rec and login in
    int upnp_vs;            // 1: UPnP; 2: vs
    int rerserver0;         // Function reservation, user_mask
    int rerserver1;
    int rerserver2;
    int rerserver3;         // Audio encoding type: 0: AAC; 1: AMR
    int rerserver4;
}USER_MANAGER;
```

3.2 Login structure

```
typedef struct Login_Pm
{
    unsigned int uActive;           // Active state  0: Not activated;  1: Activated
    unsigned int uLastSource;       // Last video source: 0: CV;  1: SV;  2: TV;  3: CV
    unsigned int uLastChnID;        // Last channel number
    unsigned int uLastFreq;         // Frequency of last channel
    unsigned int uFlag;             // 0: QCIF;  1: CIF, fixed bitrate;  2: CIF: Auto bitrate
    char SN[16];                   // User SN
    char PW[32];                   // User password
    char CmsDnsMainName[64];        // CMS dns main name
    char CmsDnsSecondName[64];     // CMS dns second name
    char p2pMainIp[32];             // P2P main IP
    char p2pSecondIp[32];          // P2P second IP
    int uNetType;                  // 0: Disables the search in LAN;  1: Enables the search in LAN.
    char cancelRecord;             // 1: Stop the Box's recording
    char PlayerType[32];           // PlayerType:"8960Player" or "8960A1Player" or "8960A2Player"
    char dir[1024];                // The current directory
    char isDirect;                 // 0: Disables the direct connection;  1: Enables the direct connection,
                                   // and connect to Box according to the specific IP directly.

    char userIp[32];               // Static ip of user set
    short int userPort;            // The rtsp port of user set
    char lastIp[32];               // IP of last success login
    short int lastPort;            // Post of last success login
    short int linktype;            // 0: userIp;  1: lastIp;  2: p2pIp
    int tunnerId;                  // Tunerid for T2 login
}LOGIN_PM;
```

