

**Further Net Software**

# **API Reference**

Issue: 1.0.0

Date: 2012.11.22

[www.dokodemoterebi.com](http://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
FurtherNet Software Development Kit (SDK)	V1.0.0

## Intended Audience

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

- Be familiar with C/C++ programming language
- Be familiar with 32-bit Windows environment/ MFC

## Organization

The document first summarizes the FurtherNet 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 the FurtherNet SDK.
2 API Description	Provides an overview of FurtherNet SDK API, and describes all the API functions in detail.
3 Data Structures Description	Describes the enumerated types used by FurtherNet.

# 1 Overview

Further network library provides FurtherNet.dll file, and head files FG\_NET\_Manager.h and Ft\_Net\_Global.h. The library is mainly used for interactive communication with Further A series Box, including receiving audio/video data and sending commands to control Box.

## 1.1 Function List

Function name	Description
FG_NET_Manager	Constructor for interface objects
~FG_NET_Manager	Destructor for interface objects
AddUICallBack	Callback function provided for UI
NetLogin	Logout
NetPlayLive	Receives Box's audio/video data
NetTearDown	Releases network module source
Net_GetNetStat	Gets the current network type
Net_PlayLive	Starts to play data
SetResolution	Sets resolution
SetSystemCategory	Sets TV system (NTSC or PAL)
initRtspServer	Initializes RTSP server
clearRB	Clears buffers
registerGetData_callback	Registers and gets the AV data objects

## 2 API Description

### 2.1 FurtherNet library function descriptions

#### 2.1.1 FG\_NET\_Manager();

Purpose:	Creates network module object
Parameter:	NULL
Return Value:	NULL
Note:	NULL

#### 2.1.2 ~FG\_NET\_Manager();

Purpose:	Releases network module object
Parameter:	NULL
Return Value:	NULL
Note:	NULL

#### 2.1.3 int AddUICallBack(PFCALLBACK Func);

Purpose:	Callback function provided for UI
Parameter:	Input: function pointer PFCALLBACK
Return Value:	0: failed; 1: successful
Note:	After creating network object, UI has to add callback data to receive the return messages from network. The class and macro definition of return messages is in the file Ft_Net_Globle.h. Generally, there are two possible the command replies: failed or successful.
Example:	<pre>FG_NET_Manager pNet = new FG_NET_Manager(); pNet-&gt; AddUICallBack(Func);</pre>

#### 2.1.4 bool NetLogin(LOGIN\_PM loginpm);

Purpose:	Box login panel
Parameter:	Input: structure LOGIN_PM
Return Value:	false: failed; true: successful
Note:	Before invoking this interface, user has to type LOGIN_PM (Please refer to Chapter 3 for its definition).

### 2.1.5 bool NetPlayLive();

Purpose:	Receives Box's audio/video data
Parameter:	NULL
Return Value:	false: failed; true: successful
Note:	When receiving the return messages of successful login, this interface can be invoked to begin to receive AV data.

### 2.1.6 bool NetTearDown();

Purpose:	Disconnects Box and releases network module source
Parameter:	NULL
Return Value:	false: failed; true: successful
Note:	When network SOCKET interrupts, this interface can be invoked to release the network module source. It can also be invoked when user logout Box on his own initiative.

### 2.1.7 bool NetGetNetStat();

Purpose:	Gets the current network type
Parameter:	NULL
Return Value:	LAN: NETSTAT_IN_LAN; WAN: NETSTAT_IN_WAN; Offline: NETSTAT_IN_NULL(Macro definition is in the file Ft_Net_Globle.h)
Note:	NULL

### 2.1.8 UINT NetGetCurResolution();

Purpose:	Gets the current resolution
Parameter:	NULL
Return Value:	Resolution: P(0~4) N(5~9)
Note:	When modifying video resolution, the interface can be invoked to confirm if it has been changed.

### 2.1.9 void NetSetCurResolution(UINT uCurResolution);

Purpose:	Sets the current resolution
Parameter:	Resolution: P(0~4) N(5~9)
Return Value:	NULL
Note:	NULL

### 2.1.10 UINT NetGetCurVideoSourceState();

Purpose:	Gets the current video source
Parameter:	NULL
Return Value:	Video source: 0: AV 2: TV 3:CAM
Note:	NULL

### 2.1.11 void NetSetCurVideoSourceState(UINT uCurSourceState);

Purpose:	Sets the current video source
Parameter:	Video source: 0: AV 2: TV 3:CAM
Return Value:	NULL
Note:	NULL

### 2.1.12 unsigned int NetGetTcpRBSize();

Purpose:	Get the current network buffer size
Parameter:	NULL
Return Value:	Value range: 0~5*1024*1024
Note:	NULL

### 2.1.13 bool NetSndLoginCmd(LOGIN\_PM loginPM);

Purpose:	Sends login command
Parameter:	Input: LOGIN_PM
Return Value:	false: failed; true: successful
Note:	NULL

#### 2.1.14 bool NetSndTVCmd(TV\_CMD& cmdData);

Purpose:	Sends TV commands
Parameter:	Input: TV_CMD (Please refer to Chapter 3.1 for its definition)
Return Value:	false: failed    true: successful
Note:	When sending TV command, user has to type the corresponding subcommand ID and related parameters, no need to type other irrelevant scopes. (Please refer to Chapter 3.1 for the detailed description of structure TV_CMD.)
Example	<pre>TV_CMD TVCmdData;      //Sends channel switch command TVCmdData.ID = TVCmdSubID::TVCmd_SetChn; TVCmdData.uFreq = nFreq; TVCmdData.uChnID = uChnID; pNet-&gt;NetSndTVCmd(TVCmdData);</pre>

#### 2.1.15 bool NetSndRMCmd(RM\_CMD& cmdData);

Purpose:	Sends IR learning commands
Parameter:	Input: RM_CMD (Please refer to Chapter 3.2 for its definition)
Return Value:	false: failed;    true: successful
Note:	When sending IR learning command, user has to type the corresponding subcommand ID and related parameters, no need to type other irrelevant scopes. (Please refer to Chapter 3.2 for the detailed description of structure RM_CMD.)

#### 2.1.16 bool NetSndAccountCmd(ACCOUNT\_CMD& cmdData);

Purpose:	Sends users management commands
Parameter:	Input: ACCOUNT_CMD (Please refer to Chapter 3.3 for its definition)
Return Value:	false: failed;    true: successful
Note:	When sending users management command, user has to type the corresponding subcommand ID and related parameters, no need to type other irrelevant scopes. (Please refer to Chapter 3.3 for the detailed description of structure ACCOUNT_CMD.)

### 2.1.17 bool NetSndRCCmd(RC\_CMD& cmdData);

Purpose:	Sends schedule recording commands
Parameter:	Input: RC_CMD (Please refer to Chapter 3.4 for its definition)
Return Value:	false: failed; true: successful
Note:	When sending schedule recording command, user has to type the corresponding subcommand ID and related parameters, no need to type other irrelevant scopes. (Please refer to Chapter 3.4 for the detailed description of structure RC_CMD.)

### 2.1.18 bool NetSndDLCmd(DL\_CMD& cmdData);

Purpose:	Sends recorded files download commands
Parameter:	Input: DL_CMD (Please refer to Chapter 3.5 for its definition)
Return Value:	false: failed; true: successful
Note:	When sending download command, user has to type the corresponding subcommand ID and related parameters, no need to type other irrelevant scopes. (Please refer to Chapter 3.5 for the detailed description of structure DL_CMD.)

### 2.1.19 bool NetSndPBCmd(PB\_CMD& cmdData);

Purpose:	Sends recorded files playback commands
Parameter:	Input: PB_CMD (Please refer to Chapter 3.6 for its definition)
Return Value:	false: failed; true: successful
Note:	When sending playback command, user has to type the corresponding subcommand ID and parameters, no need to type other irrelevant scopes. (Please refer to Chapter 3.6 for the detailed description of structure PB_CMD.)

### 2.1.20 bool NetSndOtherSetPortCmd(UINT uPortMode,UINT uRTSPPort,UINT uAVPort,UINT uCmdPort, UINT uUpnPVS);

Purpose:	Sends ports setting command
Parameter:	uPortMode: port type, 0: custom 1:random port; uRTSPPort: RTSP port; uAVPort: AV port; uCmdPort: CMD port; uUPnPVS: Box network mode: 1: UPNP 2: Virtual server
Return Value:	false: failed; true: successful
Note:	NULL



### 2.1.21 bool NetSndOtherUpdateVSCmd(UINT uLen);

Purpose:	Sends VS update command
Parameter:	Length of VS data
Return Value:	false: failed; true: successful
Note:	NULL

### 2.1.22 bool NetSndOtherSetSourceCmd (UINT uSource,UINT uFlag);

Purpose:	Sends video source switch command
Parameter:	Video source: 0: AV 2: TV 3:CAM uFlag: If it's failed to switch source and the failure cause is the destination source is recording, the value will be 1, else 0.
Return Value:	false: failed; true: successful
Note:	NULL

### 2.1.23 bool NetSndOtherSetAVFormatCmd(UINT uFrameRate,UINT uVbitRate,UINT uAbitRate,UINT uFormat,bool bCapasChangeFlag,bool bAudioChangeFlag);

Purpose:	Sends video transmission parameter setting command
Parameter:	uFrameRate: video frame; uVbitRate: video rate; uAbitRate: audio rate; uFormat: Video resolution, P 0-4, N 5-9; bCapasChangeFlag: checks if the video resolution has been changed. 0: unchanged 1: changed bAudioChangeFlag: checks if audio has been changed. 0: unchanged 1: changed
Return Value:	false: failed; true: successful
Note:	NULL

### 2.1.24 bool NetSndOtherSetVPPParamCmd(int iBrightness,int iHue,int iSaturation,int iContrast);

Purpose:	Sends video display parameter setting command
Parameter:	iBrightness: brightness; iHue: hue; iSaturation: saturation; iContrast: contrast
Return Value:	false: failed; true: successful
Note:	NULL

### 2.1.25 bool NetSndOtherKeepLiveCmd();

Purpose:	Sends keep-alive packet setting command
Parameter:	NULL
Return Value:	false: failed; true: successful
Note:	NULL

### 2.1.26 bool NetSndOtherGetBoxInfoCmd();

Purpose:	Sends Box version information getting command
Parameter:	NULL
Return Value:	false: failed; true: successful
Note:	NULL

### 2.1.27 bool NetSndOtherBoxNetSettingCmd(char\* pData,UINT uDataLen);

Purpose:	Sends Box network mode setting command
Parameter:	pData: Pointer to network parameter structure data; uDataLen: data length
Return Value:	false: failed; true: successful
Note:	NULL

## 3 Data Structures Description

### 3.1 TV command

#### 3.1.1 Subcommands in TV command

```
typedef enum
{
    TVCmd_SetChn = 0,           // Subcommand to swith channel
    TVCmd_StartScanChn,        // Subcommand to scan channel
    TVCmd_StopScanChn,         // Subcommand to stop scanning
    TVCmd_GetFrqTable,          // Subcommand to get channel table
    TVCmd_SetCountry,           // Subcommand to set country or region
    TVCmd_UploadTVTable,        // Subcommand to upload channel table
    TVCmd_CvbsKey               // Subcommand to send CVBS remote control key value
}TVCmdSubID;
```

#### 3.1.2 Structure transmitted by UI when sending TV command

```
typedef struct tv_cmd
{
    TVCmdSubID ID;              // Subcommand ID
    UINT uFreq;                 // Channel frequency, used by channel switch subcommand
    UINT uChnID;                // Channel ID, used by channel switch subcommand
    UINT uWay;                   // Scan mode, auto or scanning according to the preset frequency range
    UINT uCountry;              // Scan country ID, used by scan subcommand
    UINT uType;                 // Scan type, cable or air, used by scan subcommand
    UINT uBeginFrq;             // Start frequency of scanning, used by scan subcommand
    UINT uEndFrq;               // End frequency of scanning, used by scan subcommand
    char* TVTableData;          // Pointer to channel table data, using by channel table subcommand
    UINT TVTableDataLen;        // Length of channel table data, using by channel table subcommand
    UINT key;                   // CVBS key value, using by CVBS key value subcommand
}TV_CMD;
```

## 3.2 IR learning command

### 3.2.3 Subcommands in IR learning command

typedef enum

```
{
    RMCmd_StartLearn = 0,      // Subcommand to start IR learning
    RMCmd_StopLearn,          //Subcommand to stop IR learning
    RMCmd_LearnEnter,         // Subcommand to confirm IR learning
    RMCmd_NormalEnter,        // Subcommand to send IR key value
    RMCmd_GetCodeTab,         // Subcommand to get IR table data
    RMCmd_DownCodeTab,        // Subcommand to upload IR table to Box
}RMCmdSubID;
```

### 3.2.2 Structure transmitted by UI when sending IR learning command

typedef struct rm\_cmd

```
{
    RMCmdSubID ID;            // Subcommand ID
    CString strDevname;        // IR learning device name, used by learning start, stop, confirming and
                                key value sending subcommamds
    char* pData;              // Pointer to uploaded IR table data, usd by IR table upload subcommand
    UINT uDataLen;            // Length of IR table data, used by IR table upload subcommand
    UINT uCode;               // IR key value, used by confirm and key value sending subcommand
}RM_CMD;
```

## 3.3 User management command

### 3.3.1 Subcommands in user management command

typedef enum

```
{
    AccountCmd_AddUser = 0,           // Subcommand to add a user
    AccountCmd_GetUserInfo,          // Subcommand to get a user's information
    AccountCmd_ModifyPAW,            // Subcommand to change user password
    AccountCmd_DelUser,              // Subcommand to delete a user
    AccountCmd_ModifyAdmin,          // Subcommand to change manager information
    AccountCmd_ModifyGuest,          // Subcommand to change the ordinary user information
    AccountCmd_GetAllInfo            // Subcommand to get all user information
}AccountCmdSubID;
```

### 3.3.2 Structure transmitted by UI when sending user management command

typedef struct account\_cmd

```
{
    AccountCmdSubID ID;               // Subcommand ID
    CString strUser;                  // User name, used by adding, modifying or deleting user, and
                                     // changing manager or ordinary user information subcommands
    CString strPAW;                   // User current password or new password
    CString strOldPAW;                // User old password
}ACCOUNT_CMD;
```

## 3.4 Schedule recording task management command

### 3.4.1 Subcommands in schedule recording task management command

```
typedef enum
{
    RCCmd_ModifyTime = 0,          // Subcommand to change Box's time
    RCCmd_UpdateInfo,              // Subcommand to update shcedule recording tasks information
    RCCmd_DeITask,                 // Subcommand to delete the schedule recording task
    RCCmd_ModifyTask,              // Subcommand to edit the schedule recording task
    RCCmd_AddTask,                 // Subcommand to add a schedule recording task
    RCCmd_ModifyTimeZone,          // Subcommand to change Box's time zone
    RCCmd_IsCancel                 // Subcommand to cancel the schedule recording task
}RCCmdSubID;
```

### 3.4.2 Structure transmitted by UI when sending schedule recording command

```
typedef struct rc_cmd
{
    RCCmdSubID ID;                 // Subcommand ID
    char* pTTZ;                    // Pointer to time zone data or recording task data, used by changing
                                   Box's time zone or time, and adding or editing recording task subcommands

    UINT uLen;                     // Data length
    UINT uTaskId;                  // Recording task ID, used by task deleting subcommand
    UINT uIndex;                   // Time zone index value, used by time zone changing subcommand
    int value;                     // Recording task cancel mark, 0: don't cancel 1: cancel, used by task
                                   cancel subcommand
}RC_CMD;
```

## 3.5 Remote recorded files download command

### 3.5.1 Subcommands in recorded files download command

typedef enum

```
{
    DLCmd_FileUpdateInfo = 0,      // Subcommand to get recorded file information
    DLCmd_FileReq,                 // Subcommand to send download request
    DLCmd_FileBegin,               // Subcommand to start to download
    DLCmd_FileCancel,              // Subcommand to cancel the download
    DLCmd_FileDel                   // Subcommand to delete remote recorded the file
}DLCmdSubID;
```

### 3.5.2 Structure transmitted by UI when sending download command

typedef struct dl\_cmd

```
{
    DLCmdSubID ID;                 // Subcommand ID
    FILE_CUT fileCut;              // Recorded file information structure
    UINT uLastSource;              // Return video source when canceling download, used by download cancel
                                   // subcommand
    UINT uFrq;                     // Return channel frequency when canceling download, used by download
                                   // cancel subcommand
    int value;                     // Record file index value, used by download request subcommand
}DL_CMD;
```

## 3.6 Remote recorded files playback command

### 3.6.1 Subcommands in remote recorded files playback command

typedef enum

```
{
    PBCmd_PlayLiveReq = 0,      // Subcommand to send playback request
    PBCmd_PlayLiveBegin,       // Subcommand to start to play remote recorded files
    PBCmd_PlayLiveCancel,      // Subcommand to cancel playing
    PBCmd_PlayLiveSeek         // Subcommand to set the playback file position
}PBCmdSubID;
```

### 3.6.2 Structure transmitted by UI when sending playback command

typedef struct pb\_cmd

```
{
    PBCmdSubID ID;              // Subcommand ID
    FILE_CUT fileCut;           // Recorded file information structure, used by playback request and
                                // playback starting subcommand
    int nRecCancel;             // Record cancel mark, used by playback position setting and playback
                                // cancel subcommand
    UINT uLastSource;           // Return source when canceling back, used by playback cancel
                                // subcommand
    UINT uFrq;                  // Return channel frequency when canceling playback, used by playback
                                // cancel subcommand
    bool back;                  // Function reservation
    UINT uPosition;             // The playback file position, used by playback setting subcommand
}PB_CMD;
```



