AIRLITE

AIRLITE C-Library

API Reference Manual

Version 1.1

D&R Electronica B.V.
Rijnkade 15B
1382GS Weesp
The Netherlands
Phone: +31 (0)294-418014
Fax: +31 (0)294-416987
Website: http://www.d-r.nl
E-mail: info@d-r.nl

Document Version 1.1 - 2017-06-12

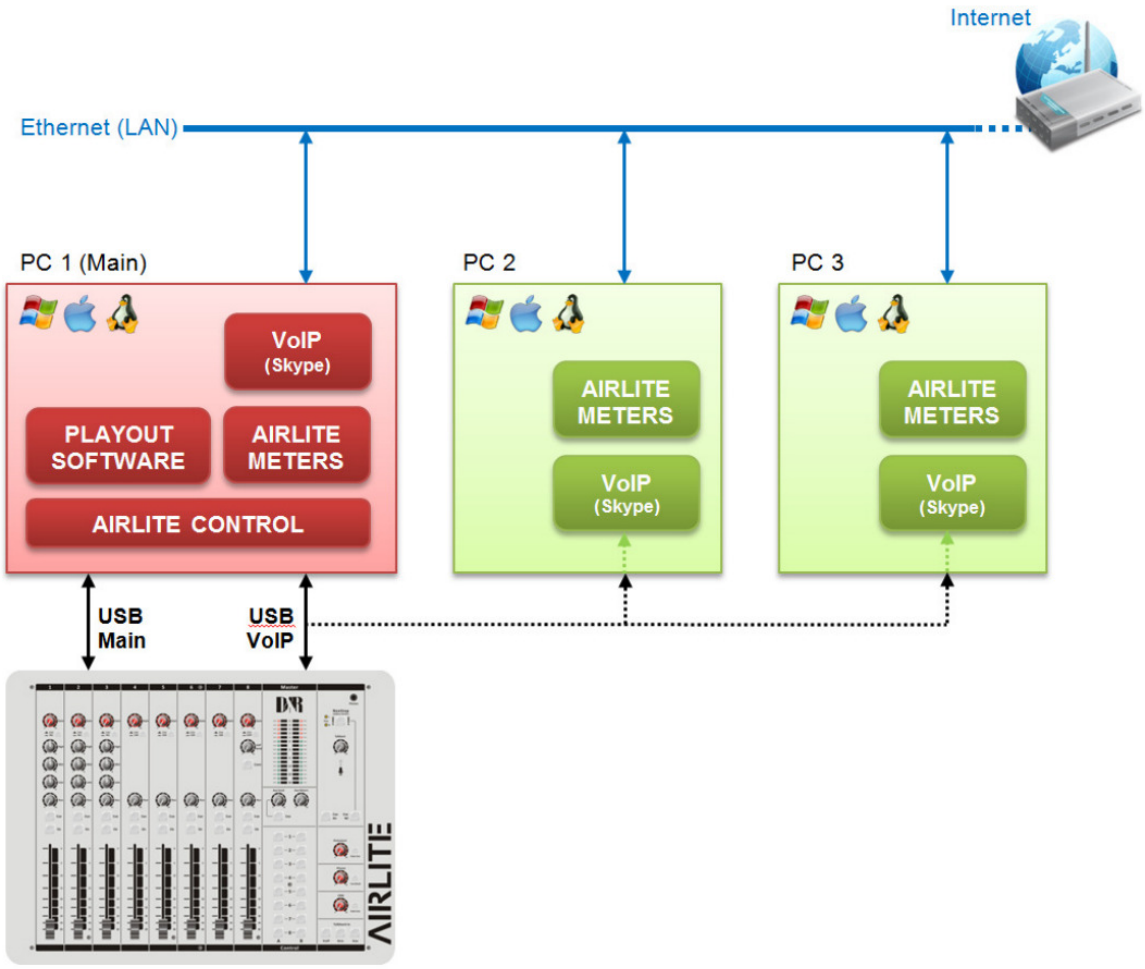
1 Table of contents

2 Introduction

The Airlite C Library provides an API to access/modify all the internal settings and/or remote control the device. When integrating the library with playout software one is also able to control this software with the switches on the desk.

The library is intended to be used with any programming language available on Windows which supports loading a DLL (Dynamic Link Library). The name of the library is *airlite.dll.*

NOTE: *Airlite Control* is required to be installed and running in order to communicate



3 Using the library

3.1 C Example

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "airlite.h"

void myCallback( Callback_st *cb, void *userdata );
AIRLITE_HANDLE handle;

int main()
{
        /* Open */
        handle = airlite_open("127.0.0.1", 19550, 19551 );
        if( handle == NULL ){
                printf("Failed to open device\n");
                return -1;
        }else printf("Succesfully opened device\n");

        /* register callback */
        airlite_register_callback( handle, myCallback, NULL );

        /* library version */
        Version_st version;
        printf("%s\n", airlite_library_version( &version ));

        /* send some request to the device */
        airlite_request(handle, STATE_ON_REQ );
        airlite_request(handle, CTRL_SWITCH_REQ );


        printf("Press a key to Exit\n");
        getch();

        /* Close */
        airlite_close( handle );
        return 0;
}

void myCallback( Callback_st *cb, void *userdata )
{
        /* Your code here... */

        /* Handle incoming EVENTS and REPLY messages */
        /* Typecast the callback data to the correct data struct */
}
```

4 API Reference

4.1 Defines

```
#define b_CUE_RESET_ENABLED_BY_LINE           0x01
#define b_CUE_RESET_ENABLED_BY_MIC_USB_VOIP   0x02
#define b_CUE_RESET_BY_FADER                  0x04
#define b_CUE_RESET_BY_ON                     0x08
#define b_CUE_RESET_BY_FADER_AND_ON           0x10

#define b_ON_AIR1_ENABLED_BY_LINE             0x01
#define b_ON_AIR1_ENABLED_BY_MIC_USB_VOIP     0x02
#define b_ON_AIR2_ENABLED_BY_LINE             0x04
#define b_ON_AIR2_ENABLED_BY_MIC_USB_VOIP     0x08

#define b_STUDIO_REMOTE_COUGH_ENABLE          0x01
#define b_STUDIO_REMOTE_COMM_ENABLE           0x02
#define b_STUDIO_REMOTE_COMM_PRIVATE          0x04
```

4.2 Enumerations

4.2.1 Module_t

```
typedef enum {
        MODULE_1,
        MODULE_2,
        MODULE_3,
        MODULE_4,
        MODULE_5,
        MODULE_6,
        MODULE_7,
        MODULE_8,
        MODULE_ALL=0xFF
}Module_t;
```

4.2.2 Cue_t

```
typedef enum {
        CUE_MOD_1,
        CUE_MOD_2,
        CUE_MOD_3,
        CUE_MOD_4,
        CUE_MOD_5,
        CUE_MOD_6,
        CUE_MOD_7,
        CUE_MOD_8,
        CUE_AUX,
        CUE_NON_STOP,
        CUE_AIR,
        CUE_ALL=0xFF
}Cue_t;
```

4.2.3 Color_t

```
typedef enum {
        LED_OFF,
        LED_RED,
        LED_GREEN
}Color_t;
```

4.2.4 Ctrl_blink_t

```
typedef enum {
        CTRL_SLOW,
        CTRL_NORMAL,
        CTRL_FAST
}Ctrl_blink_t;
```

4.2.5 Ctrl_id_t

```
typedef enum {
    CTRL_1A,
    CTRL_2A,
    CTRL_3A,
    CTRL_4A,
    CTRL_5A,
    CTRL_6A,
    CTRL_7A,
    CTRL_8A,
    CTRL_1B,
    CTRL_2B,
    CTRL_3B,
    CTRL_4B,
    CTRL_5B,
    CTRL_6B,
    CTRL_7B,
    CTRL_8B,
        CTRL_ALL=0xFF
}Ctrl_id_t;
```

4.2.6 Request_t

```
typedef enum {
        FIRMWARE_REQ,
        /* Control:     0x20 - 0x3F */
        CTRL_SWITCH_REQ = 0x20,
        /* Config:      0x40 - 0x5F */
        CONFIG_AUTO_ON_REQ = 0x40,
        CONFIG_CUE_RESET_REQ,
        CONFIG_ON_AIR_REQ,
        CONFIG_CRM_MUTE_REQ,
        CONFIG_CRM_AUTO_CUE_RESET_REQ,
        CONFIG_STUDIO_REMOTE_REQ,
        CONFIG_CUE_INTERLOCK_REQ,
        CONFIG_GPO_REQ,
        CONFIG_CUE_REQ,
        CONFIG_ON_REQ,
        CONFIG_FADER_RESERVE_REQ,
        CONFIG_SILENCE_DETECTION_REQ,
        CONFIG_PHANTOM_REQ,
        CONFIG_MIC_AT_LINE_REQ,
        CONFIG_AUTO_OFF_REQ,
        /* States:      0x60 - 0x7F */
        STATE_ON_REQ = 0x60,
        STATE_CUE_REQ,
        STATE_FADER_REQ,
        STATE_AUTO_CUE_REQ,
        STATE_SRC_SEL_REQ,
        STATE_NON_STOP_REQ,
        STATE_MIC_ON_REQ,
        STATE_GPO_REQ,
        STATE_ON_AIR_REQ,
        STATE_SILENCE_DETECTION_REQ,
        STATE_MODULE_REQ,
        STATE_CRM_MUTE_REQ,
        STATE_TRACK_REQ,
        STATE_COMM_REQ,
        STATE_COUGH_REQ,
        STATE_VT_REQ
        /* Misc.:       0x80 - 0xFF */
}Request_t;
```

4.2.7 Action_t

```
typedef enum {
        DEACTIVATE,
        ACTIVATE,
        TOGGLE
}Action_t;
```

4.2.8 Config_t

```
typedef enum {
        DISABLE,
        ENABLE
}Config_t;
```

4.2.9 Callback_id_t

```
typedef enum {
        FIRMWARE,
        /* Control:     0x20 - 0x3F */
        CTRL_LED = 0x20,
        CTRL_LED_BLINK,
        CTRL_SWITCH,
        /* Config:      0x40 - 0x5F */
        CONFIG_AUTO_ON = 0x40,
        CONFIG_CUE_RESET,
        CONFIG_ON_AIR,
        CONFIG_CRM_MUTE,
        CONFIG_CRM_AUTO_CUE_RESET,
        CONFIG_STUDIO_REMOTE,
        CONFIG_CUE_INTERLOCK,
        CONFIG_GPO,
        CONFIG_CUE,
        CONFIG_ON,
        CONFIG_FADER_RESERVE,
        CONFIG_SILENCE_DETECTION,
        CONFIG_PHANTOM,
        CONFIG_MIC_AT_LINE,
        CONFIG_AUTO_OFF,
        /* States:      0x60 - 0x7F */
        STATE_ON = 0x60,
        STATE_CUE,
        STATE_FADER,
        STATE_AUTO_CUE,
        STATE_SRC_SEL,
        STATE_NON_STOP,
        STATE_MIC_ON,
        STATE_GPO,
        STATE_ON_AIR,
        STATE_SILENCE_DETECTION,
        STATE_MODULE,
        STATE_CRM_MUTE,
        STATE_TRACK,
        STATE_COMM,
        STATE_COUGH,
        STATE_VT,
        STATE_PHONE_RINGING,
        /* Misc.:       0x80 - 0xFF */
        METERING = 0x80
}Callback_id_t;
```

4.2.10 Callback_type_t

```
typedef enum {
        REPLY,
        EVENT
}Callback_type_t;
```

4.2.11 Gpo_t

```
typedef enum {
        GPO_1,
        GPO_2
}Gpo_t;
```

4.2.12 Gpo_trigger_t

```
typedef enum {
        TR_NONE,
        TR_ON_AIR1,
        TR_ON_AIR2,
        TR_TEL_RING,
        TR_MIC_ON,
        TR_NON_STOP,
        TR_MOD1_ACTIVE,
        TR_MOD2_ACTIVE,
        TR_MOD3_ACTIVE,
        TR_MOD4_ACTIVE,
        TR_MOD5_ACTIVE,
        TR_MOD6_ACTIVE,
        TR_MOD7_ACTIVE,
        TR_MOD8_ACTIVE
}Gpo_trigger_t;
```

4.2.13 Gpo_mode_t

```
typedef enum {
        NON_INVERTED_OUTPUT,
        INVERTED_OUTPUT
}Gpo_mode_t;
```

4.2.14 Gpo_type_t

```
typedef enum {
        CONTINUES,
        PULSE_BY_ON,
        PULSE_BY_OFF,
        PULSE_BY_ON_OFF
}Gpo_type_t;
```

4.2.15 Selected_src_t

```
typedef enum {
    SRC_LINE,
    SRC_MIC_USB_VOIP
}Selected_src_t;
```

4.2.16 Time_unit_t

```
typedef enum {
        SECONDS,
        MINUTES
}Time_unit_t;
```

4.2.17 Audio_src_t

```
typedef enum {
        PROG_LEFT,
        PROG_RIGHT,
        PROG_STEREO
}Audio_src_t;
```

4.2.18 Silence_detection_t

```
typedef enum {
        IDLE,
        ALARM
}Silence_detection_t;
```

4.2.19 Track_t

```
typedef enum {
        TRACK_STOPPED,
        TRACK_PLAYING,
        TRACK_ENDING
}Track_t;
```

4.2.20 State_t

```
typedef enum {
        INACTIVE,
        ACTIVE
}State_t;
```

4.3 Data Stuctures

4.3.1 Version_st

```
typedef struct {
        int major;
        int minor;
        int build;
        int revision;
}Version_st;
```

4.3.2 Gpo_st

```
typedef struct {
        Gpo_trigger_t trigger;
        Gpo_mode_t mode;
        Gpo_type_t type;
        unsigned char time;
}Gpo_st;
```

4.3.3 Cue_reset_config_st

```
typedef struct {
        unsigned char CUE_RESET_ENABLED_BY_LINE           : 1;
        unsigned char CUE_RESET_ENABLED_BY_MIC_USB_VOIP   : 1;
        unsigned char CUE_RESET_BY_FADER                  : 1;
        unsigned char CUE_RESET_BY_ON                     : 1;
        unsigned char CUE_RESET_BY_FADER_AND_ON           : 1;
}Cue_reset_config_st;
```

4.3.4 On_air_config_st

```
typedef struct {
        unsigned char ON_AIR1_ENABLED_BY_LINE             : 1;
        unsigned char ON_AIR1_ENABLED_BY_MIC_USB_VOIP     : 1;
        unsigned char ON_AIR2_ENABLED_BY_LINE             : 1;
        unsigned char ON_AIR2_ENABLED_BY_MIC_USB_VOIP     : 1;
}On_air_config_st;
```

4.3.5 Studio_remote_config_st

```
typedef struct {
        unsigned char STUDIO_REMOTE_COUGH_ENABLE          : 1;
        unsigned char STUDIO_REMOTE_COMM_ENABLE           : 1;
        unsigned char STUDIO_REMOTE_COMM_PRIVATE          : 1;
}Studio_remote_config_st;
```

4.3.6 Cue_st

```
typedef struct {
    Color_t active;
    Color_t inactive;
}Cue_st;
```

4.3.7 On_st

```
typedef struct {
        Color_t on_active;
        Color_t on_and_fader_active;
        Color_t inactive;
}On_st;
```

4.3.8 Firmware_st

```
typedef struct {
        unsigned char major;
        unsigned char minor;
        char version[100];
}Firmware_st;
```

4.3.9 Ctrl_led_st

```
typedef struct {
    Ctrl_id_t       id;
    Color_t         color;
}Ctrl_led_st;
```

4.3.10 Ctrl_led_blink_st

```
typedef struct {
    Ctrl_id_t       id;
    Color_t         color_on;
    Color_t         color_off;
    Ctrl_blink_t    speed;
}Ctrl_led_blink_st;
```

4.3.11 Ctrl_switch_st

```
typedef struct {
    unsigned short  sw_states;
}Ctrl_switch_st;
```

4.3.12 Module_1_8_st

```
typedef struct {
    State_t module1;
    State_t module2;
    State_t module3;
    State_t module4;
    State_t module5;
    State_t module6;
    State_t module7;
    State_t module8;
}Module_1_8_st;
```

4.3.13 Module_1_3_st

```
typedef struct {
    State_t module1;
    State_t module2;
    State_t module3;
}Module_1_3_st;
```

4.3.14 State_cue_st

```
typedef struct {
        State_t module1;
    State_t module2;
    State_t module3;
    State_t module4;
    State_t module5;
    State_t module6;
    State_t module7;
    State_t module8;
        State_t aux;
        State_t nonstop;
        State_t air;
}State_cue_st;
```

4.3.15 Config_cue_reset_st

```
typedef struct {
    Cue_reset_config_st module1;
    Cue_reset_config_st module2;
    Cue_reset_config_st module3;
    Cue_reset_config_st module4;
    Cue_reset_config_st module5;
    Cue_reset_config_st module6;
    Cue_reset_config_st module7;
    Cue_reset_config_st module8;
}Config_cue_reset_st;
```

4.3.16 Config_on_air_st

```
typedef struct {
    On_air_config_st module1;
    On_air_config_st module2;
    On_air_config_st module3;
    On_air_config_st module4;
    On_air_config_st module5;
    On_air_config_st module6;
    On_air_config_st module7;
    On_air_config_st module8;
}Config_on_air_st;
```

4.3.17 Config_studio_remote_st

```
typedef struct {
    Studio_remote_config_st module1;
    Studio_remote_config_st module2;
    Studio_remote_config_st module3;
}Config_studio_remote_st;
```

4.3.18 Config_gpo_st

```
typedef struct {
    Gpo_st gpo1;
    Gpo_st gpo2;
}Config_gpo_st;
```

4.3.19 Config_cue_st

```
typedef struct {
    Cue_st module1;
    Cue_st module2;
    Cue_st module3;
    Cue_st module4;
    Cue_st module5;
    Cue_st module6;
    Cue_st module7;
    Cue_st module8;
    Cue_st aux;
    Cue_st nonstop;
    Cue_st air;
}Config_cue_st;
```

4.3.20 Config_on_st

```
typedef struct {
    On_st module1;
    On_st module2;
    On_st module3;
    On_st module4;
    On_st module5;
    On_st module6;
    On_st module7;
    On_st module8;
}Config_on_st;
```

4.3.21 State_auto_cue_st

```
typedef struct {
    State_t auto_cue_crm;
    State_t auto_cue_announcer;
}State_auto_cue_st;
```

4.3.22 State_src_sel_st

```
typedef struct {
    Selected_src_t module1;
    Selected_src_t module2;
    Selected_src_t module3;
    Selected_src_t module4;
    Selected_src_t module5;
    Selected_src_t module6;
    Selected_src_t module7;
    Selected_src_t module8;
}State_src_sel_st;
```

4.3.23 State_st

```
typedef struct {
    State_t state;
}State_st;
```

4.3.24 State_gpo_st

```
typedef struct {
    State_t gpo1;
    State_t gpo2;
}State_gpo_st;
```

4.3.25 State_on_air_st

```
typedef struct {
    State_t on_air1;
    State_t on_air2;
}State_on_air_st;
```

4.3.26 Config_silence_detection_st

```
typedef struct {
        char          threshold;
        unsigned char interval;
        Time_unit_t   interval_unit;
        Audio_src_t   source;
        Config_t      config;
}Config_silence_detection_st;
```

4.3.27 State_silence_detection_st

```
typedef struct {
        Silence_detection_t state;
}State_silence_detection_st;
```

4.3.28 State_track_st

```
typedef struct {
        Track_t module1;
        Track_t module2;
        Track_t module3;
        Track_t module4;
        Track_t module5;
        Track_t module6;
        Track_t module7;
        Track_t module8;
}State_track_st;
```

4.3.29 Metering_st

```
typedef struct {
        float prog_left_db;
        float prog_right_db;
        float phones_left_db;
        float phones_right_db;
        float crm_left_db;
        float crm_right_db;
}Metering_st;
```

4.3.30 Callback_st

```
typedef struct {
        Callback_id_t   id;
        Callback_type_t type;
        Callback_data_t data;
}Callback_st;
```

4.4 Union

```
/*****************************************************************************/
/* UNION (used to cast the callback data to the corresponding structure) */
/*****************************************************************************/

typedef union {
    Firmware_st                     *FIRMWARE;
    Ctrl_led_st                     *CTRL_LED;
    Ctrl_led_blink_st               *CTRL_LED_BLINK;
    Ctrl_switch_st                  *CTRL_SWITCH;
    Module_1_8_st                   *CONFIG_AUTO_ON;
    Config_cue_reset_st             *CONFIG_CUE_RESET;
    Config_on_air_st                *CONFIG_ON_AIR;
    Module_1_3_st                   *CONFIG_CRM_MUTE;
    Module_1_3_st                   *CONFIG_CRM_AUTO_CUE_RESET;
    Config_studio_remote_st         *CONFIG_STUDIO_REMOTE;
    State_st                        *CONFIG_CUE_INTERLOCK;
    Config_gpo_st                   *CONFIG_GPO;
    Config_cue_st                   *CONFIG_CUE;
    Config_on_st                    *CONFIG_ON;
    Module_1_8_st                   *CONFIG_FADER_RESERVE;
    Config_silence_detection_st     *CONFIG_SILENCE_DETECTION;
    Module_1_3_st                   *CONFIG_PHANTOM;
    Module_1_3_st                   *CONFIG_MIC_AT_LINE;
    Module_1_8_st                   *CONFIG_AUTO_OFF;
    Module_1_8_st                   *STATE_ON;
    State_cue_st                    *STATE_CUE;
    Module_1_8_st                   *STATE_FADER;
    State_auto_cue_st               *STATE_AUTO_CUE;
    State_src_sel_st                *STATE_SRC_SEL;
    State_st                        *STATE_NON_STOP;
    State_st                        *STATE_MIC_ON;
    State_gpo_st                    *STATE_GPO;
    State_on_air_st                 *STATE_ON_AIR;
    State_silence_detection_st      *STATE_SILENCE_DETECTION;
    Module_1_8_st                   *STATE_MODULE;
    State_st                        *STATE_CRM_MUTE;
    State_track_st                  *STATE_TRACK;
    Module_1_3_st                   *STATE_COMM;
    Module_1_3_st                   *STATE_COUGH;
    Module_1_8_st                   *STATE_VT;
    State_st                        *STATE_PHONE_RINGING;
    Metering_st                     *METERING;
}Callback_data_t;
```

4.5 Functions

4.5.1 airlite_library_version()

Retreive the current version of this library.

```
char* airlite_library_version( Version_st* version );
```

4.5.1.1 Parameters

[OUT] version
: Pointer to a version information structure or NULL.

4.5.1.2 Return Value

Pointer to character array containing the version as a null-terminated C-string.

4.5.2 airlite_open()

Open a UDP/IP connection to communicate with *Airlite Control* application. Calling this function is required before any other function of the library can be used where an `AIRLITE_HANDLE` is present as parameter.

```
AIRLITE_HANDLE airlite_open(   char* remotehost,
                               unsigned short udp_recv_port,
                               unsigned short udp_send_port );
```

4.5.2.1 Parameters

[IN] remotehost
: IP address of the host where to connect to. (e.g. “127.0.0.1” for localhost).

[IN] udp_recv_port
: UDP receive port.

[IN] udp_send_port
: UDP send port.

4.5.2.2 Return Value

`AIRLITE_HANDLE` on succes, otherwise NULL.

4.5.3 airlite_close()

Close the connection and free allocated memory of internal resources created by the function *airlite_open()*. Calling this function terminates the communication.

```
int airlite_close( AIRLITE_HANDLE handle );
```

4.5.3.1 Parameters

[IN] handle
: Handle to an opened connection.

4.5.3.2 Return Value

0 on succes, otherwise -1.

4.5.4 airlite_register_callback()

Register a callback function *callback* to an opened connection represented by *handle*. The callback function is called on reply and event messages.

```
int airlite_register_callback( AIRLITE_HANDLE handle,
                               AIRLITE_CB callback,
                               void *userdata );
```

4.5.4.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] callback
: Address of the callback function.

[IN] userdata
: User data pointer or NULL.

4.5.4.2 Return Value

0 on succes, otherwise -1.

4.5.5 airlite_unregister_callback()

Unregister the callback function of an opened connection represented by *handle*.

```
int airlite_unregister_callback( AIRLITE_HANDLE handle );
```

4.5.5.1 Parameters

[IN] handle
: Handle to an opened connection.

4.5.5.2 Return Value

0 on succes, otherwise -1.

4.5.6 airlite_request()

Send a request message. A request can be used to obtain some information from the device. As a result, the corresponding reply message is returned by the device.

```
int airlite_request(   AIRLITE_HANDLE handle,
                       Request_t request );
```

4.5.6.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] request
: Type of request to send.

4.5.6.2 Return Value

0 on succes, otherwise -1.

4.5.7 airlite_ctrl_led()

Set the color of the sixteen control switches on the console.

```
int airlite_ctrl_led(  AIRLITE_HANDLE handle,
                       Ctrl_id_t id,
                       Color_t color );
```

4.5.7.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] id
Control switch identifier.

[IN] color
Color to set the switch to.

4.5.7.2 Return Value

0 on succes, otherwise -1.

4.5.8 airlite_ctrl_led_blink()

Set the blinking colors of the sixteen control switches on the console and activate blinking. Use the function *airlite_ctrl_led()* to terminate blinking.

```
int airlite_ctrl_led_blink(    AIRLITE_HANDLE handle,
                               Ctrl_id_t id,
                               Color_t color_on,
                               Color_t color_off,
                               Ctrl_blink_t blink_speed );
```

4.5.8.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] id
Control switch identifier.

[IN] color_on
Color to set the switch in on-state.

[IN] color_off
Color to set the switch in off-state.

[IN] blink_speed
Alternate-speed between on and off state.

4.5.8.2 Return Value

0 on succes, otherwise -1.

4.5.9 airlite_remote_on()

Remote control the On switch of one or all modules.

```
int airlite_remote_on( AIRLITE_HANDLE handle,
                       Module_t module,
                       Action_t action );
```

4.5.9.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] module
: Module identifier.

[IN] action
: Activate, de-activate, or toggle.

4.5.9.2 Return Value

0 on succes, otherwise -1

4.5.10 airlite_remote_cue()

Remote control the Cue switch of one or all modules.

```
int airlite_remote_cue(        AIRLITE_HANDLE handle,
                               Cue_t cue,
                               Action_t action );
```

4.5.10.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] cue
: Cue identifier. (including Aux, Air, NonStop).

[IN] action
: Activate, de-activate, or toggle.

4.5.10.2 Return Value

0 on succes, otherwise -1

4.5.11 airlite_remote_cue_reset()

Remote control the Cue Reset switch.

```
int airlite_remote_cue_reset( AIRLITE_HANDLE handle );
```

4.5.11.1 Parameters

[IN] handle
: Handle to an opened connection.

4.5.11.2 Return Value

0 on succes, otherwise -1

4.5.12 airlite_remote_auto_cue_announcer()

Remote control the Auto Cue Announcer switch.

```
int airlite_remote_auto_cue_announcer(AIRLITE_HANDLE handle,
                                      Action_t action );
```

4.5.12.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] action
: Activate, de-activate, or toggle.

4.5.12.2 Return Value

0 on succes, otherwise -1

4.5.13 airlite_remote_auto_cue_crm()

Remote control the Auto Cue CRM switch.

```
int airlite_remote_auto_cue_crm(      AIRLITE_HANDLE handle,
                                      Action_t action );
```

4.5.13.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] action
: Activate, de-activate, or toggle.

4.5.13.2 Return Value

0 on succes, otherwise -1

4.5.14 airlite_remote_non_stop()

Remote control the Non Stop switch.

```
int airlite_remote_non_stop(   AIRLITE_HANDLE handle,
                               Action_t action );
```

4.5.14.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] action
: Activate, de-activate, or toggle.

4.5.14.2 Return Value

0 on succes, otherwise -1

4.5.15 airlite_remote_comm()

Remote control the Comm switch of a studio remote. Even if no studio remote is connected to the Remote jack connector at the back, this function can be seen as a virtual studio remote Comm control.

NOTE: Remote Comm function should be enabled for *module* in the device. See *airlite_config_studio_remote()* function.

```
int airlite_remote_comm(       AIRLITE_HANDLE handle,
                               Module_t module,
                               Action_t action );
```

4.5.15.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier. (Only module 1, 2, or 3).

[IN] action
Activate, de-activate, or toggle.

4.5.15.2 Return Value

0 on succes, otherwise -1

4.5.16 airlite_remote_cough()

Remote control the Cough switch of a studio remote. Even if no studio remote is connected to the Remote jack connector at the back, this function can be seen as a virtual studio remote Cough control.

NOTE: Remote Cough function should be enabled for *module* in the device. See *airlite_config_studio_remote()* function.

```
int airlite_remote_cough(      AIRLITE_HANDLE handle,
                               Module_t module,
                               Action_t action );
```

4.5.16.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier. (Only module 1, 2, or 3).

[IN] action
Activate, de-activate, or toggle.

4.5.16.2 Return Value

0 on succes, otherwise -1

4.5.17 airlite_remote_vt()

Remote control the Voicetrack mode of *module*.

```
int airlite_remote_vt( AIRLITE_HANDLE handle,
                       Module_t module,
                       Action_t action );
```

4.5.17.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] module
: Module identifier. (Only module 1, 2, 3 or 8).

[IN] action
: Activate, de-activate, or toggle.

4.5.17.2 Return Value

0 on succes, otherwise -1

4.5.18 airlite_config_save()

Save the current configuration of all settings to non-volatile (EEPROM) memory of the device.

```
int airlite_config_save( AIRLITE_HANDLE handle );
```

4.5.18.1 Parameters

[IN] handle
: Handle to an opened connection.

4.5.18.2 Return Value

0 on succes, otherwise -1

4.5.19 airlite_config_auto_on()

Configure the On switch of *module* to be automatically active when powering on the device.

```
int airlite_config_auto_on(    AIRLITE_HANDLE handle,
                               Module_t module,
                               Config_t config );
```

4.5.19.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] module
: Module identifier.

[IN] config
: Enable or disable.

4.5.19.2 Return Value

0 on succes, otherwise -1

4.5.20 airlite_config_cue_reset()

Configure the Cue Reset feature. Cue Reset can be enabled for each input source separately and activated by Faderstart, On switch or a combination of these two. Configuration is done by bit *flags*.

```
int airlite_config_cue_reset(  AIRLITE_HANDLE handle,
                               Module_t module,
                               unsigned char flags );
```

4.5.20.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier.

[IN] flags
Configuration octet defined as bit flags:

```
#define b_CUE_RESET_ENABLED_BY_LINE           0x01
#define b_CUE_RESET_ENABLED_BY_MIC_USB_VOIP   0x02
#define b_CUE_RESET_BY_FADER                  0x04
#define b_CUE_RESET_BY_ON                     0x08
#define b_CUE_RESET_BY_FADER_AND_ON           0x10
```

4.5.20.2 Return Value

0 on succes, otherwise -1

4.5.21 airlite_config_on_air()

Configure the two logical On Air busses to be enabled for each input source separately. Configuration is done by bit *flags*.

```
int airlite_config_on_air(     AIRLITE_HANDLE handle,
                               Module_t module,
                               unsigned char flags );
```

4.5.21.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier.

[IN] flags
Configuration octet defined as bit flags:

```
#define b_ON_AIR1_ENABLED_BY_LINE             0x01
#define b_ON_AIR1_ENABLED_BY_MIC_USB_VOIP     0x02
#define b_ON_AIR2_ENABLED_BY_LINE             0x04
#define b_ON_AIR2_ENABLED_BY_MIC_USB_VOIP     0x08
```

4.5.21.2 Return Value

0 on succes, otherwise -1

4.5.22 airlite_config_crm_mute()

Configure the Control Room Monitor (CRM) mute. The CRM output will be muted if *module* is active (On + Fader up) and Mic is selected as input source.

```
int airlite_config_crm_mute(   AIRLITE_HANDLE handle,
                               Module_t module,
                               Config_t config );
```

4.5.22.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier.

[IN] config
Enable or disable.

4.5.22.2 Return Value

0 on succes, otherwise -1

4.5.23 airlite_config_crm_auto_cue_reset()

Configure the CRM Auto Cue Reset. The CRM Auto Cue function is temporarely disabled(or reset) if Mic is selected as input source and Cue is active on *module*.

```
int airlite_config_crm_auto_cue_reset(        AIRLITE_HANDLE handle,
                                              Module_t module,
                                              Config_t config );
```

4.5.23.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier.

[IN] config
Enable or disable.

4.5.23.2 Return Value

0 on succes, otherwise -1

4.5.24 airlite_config_studio_remote()

Configure Studio Remote functionality. Cough and Comm can be enabled separately for *module*. Configuration is done by bit *flags*. Private Comm results in only *module* to be active on the cue buss.

```
int airlite_config_studio_remote(     AIRLITE_HANDLE handle,
                                      Module_t module,
                                      unsigned char flags );
```

4.5.24.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier.

[IN] flags
Configuration octet defined as bit flags:

```
#define b_STUDIO_REMOTE_COUGH_ENABLE          0x01
#define b_STUDIO_REMOTE_COMM_ENABLE           0x02
#define b_STUDIO_REMOTE_COMM_PRIVATE          0x04
```

4.5.24.2 Return Value

0 on succes, otherwise -1

4.5.25 airlite_config_phantom()

Configure Phantom power (+48V) for condensor microphones.

```
int airlite_config_phantom(    AIRLITE_HANDLE handle,
                               Module_t module,
                               Config_t config );
```

4.5.25.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier. (Only module 1, 2, or 3).

[IN] config
Enable or disable.

4.5.25.2 Return Value

0 on succes, otherwise -1

4.5.26 airlite_config_mic_at_line()

Configure the Mic at Line feature. Use this if an external mic pre-amp is connected to the line input of *module*.

```
int airlite_config_mic_at_line(        AIRLITE_HANDLE handle,
                                       Module_t module,
                                       Config_t config );
```

4.5.26.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier. (Only module 1, 2, or 3).

[IN] config
Enable or disable.

4.5.26.2 Return Value

0 on succes, otherwise -1

4.5.27 airlite_config_auto_off()

Configure the On switch of *module* to be automatically in-active when fader gets down.

```
int airlite_config_auto_off(   AIRLITE_HANDLE handle,
                               Module_t module,
                               Config_t config );
```

4.5.27.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier.

[IN] config
Enable or disable.

4.5.27.2 Return Value

0 on succes, otherwise -1

4.5.28 airlite_config_cue_interlock()

Configure Cue Interlock. Allows one module to be active on the cue buss at once.

```
int airlite_config_cue_interlock(      AIRLITE_HANDLE handle,
                                       Config_t config );
```

4.5.28.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] config
Enable or disable.

4.5.28.2 Return Value

0 on succes, otherwise -1

4.5.29 airlite_config_gpo()

Configure the two General Purpose Outputs (GPO) of the device.

```
int airlite_config_gpo(         AIRLITE_HANDLE handle,
                                Gpo_t gpo,
                                Gpo_trigger_t trigger,
                                Gpo_mode_t mode,
                                Gpo_type_t type,
                                unsigned char time );
```

4.5.29.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] gpo
: GPO identifier.

[IN] trigger
: The source to trigger *gpo*.

[IN] mode
: Non-inverted or inverted output.

[IN] type
: Continuous or pulse.

[IN] time
: GPO time when *type* is configured as pulse. {10...255} [ms].

4.5.29.2 Return Value

0 on succes, otherwise -1

4.5.30 airlite_config_cue()

Configure the colors of the Cue switches depending their state.

```
int airlite_config_cue(         AIRLITE_HANDLE handle,
                                Cue_t cue,
                                Color_t active,
                                Color_t inactive );
```

4.5.30.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] cue
: Cue identifier.

[IN] active
: Cue active color.

[IN] inactive
: Cue inactive color.

4.5.30.2 Return Value

0 on succes, otherwise -1.

4.5.31 airlite_config_on()

Configure the colors of the On switches depending their state.

```
int airlite_config_on( AIRLITE_HANDLE handle,
                       Module_t module,
                       Color_t active,
                       Color_t active_and_fader,
                       Color_t inactive );
```

4.5.31.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] module
: Module identifier.

[IN] active
: On active color.

[IN] active_and_fader
: On active and fader up color.

[IN] inactive
: On inactive color.

4.5.31.2 Return Value

0 on succes, otherwise -1.

4.5.32 airlite_config_fader_reserve()

Configure the dynamic range of the faders on the desk. 0dB or +10dB on top. If fader reserve is enabled +10dB is on top.

```
int airlite_config_fader_reserve(     AIRLITE_HANDLE handle,
                                      Module_t module,
                                      Config_t config );
```

4.5.32.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] module
: Module identifier.

[IN] config
: Enable or disable.

4.5.32.2 Return Value

0 on succes, otherwise -1

4.5.33 airlite_config_silence_detection()

Configure the Silence Detector.

```
int airlite_config_silence_detection( AIRLITE_HANDLE handle,
                                      char threshold,
                                      unsigned char interval,
                                      Time_unit_t interval_unit,
                                      Audio_src_t source,
                                      Config_t config );
```

4.5.33.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] threshold
: The detector is active below this level {0..-40} [dB] .

[IN] interval
: Amount of time after which the detector enters alarm state. {2..255}

[IN] interval_unit
: Seconds or minutes.

[IN] source
: The input source for the detector to listen on.

[IN] config
: Enable or disable.

4.5.33.2 Return Value

0 on succes, otherwise -1

4.5.34 airlite_authenticate()

The Airlite device contains internal authorization keys for third-party software activation. AES128 encryption is used to encrypt the provided *data*. A dedicated key can be requested from D&R. This function is blocking during the encryption process.

```
int airlite_authenticate(      AIRLITE_HANDLE handle,
                               int key,
                               void *data,
                               int len );
```

4.5.34.1 Parameters

[IN] handle
: Handle to an opened connection.

[IN] key
: Key identifier. Value of the internal key is 16 bytes (128 bits).

[IN][OUT] data
: Buffer to read input data from and write encrypted data back to.

[IN] len
: Length of *data* buffer. 16 bit aligned.

4.5.34.2 Return Value

0 on succes, otherwise -1

4.5.35 airlite_track_state()

Visualize the state of a (usb)track in a playout application to the On switch of *module*. Several states are represented using different colors and blinking behavior. In such a way the user behind the desk is notified about a track is playing, stopped or ending.

```
int airlite_track_state(       AIRLITE_HANDLE handle,
                               Module_t module,
                               Track_t state );
```

4.5.35.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] module
Module identifier.

[IN] state
State of the track in a playout application.

4.5.35.2 Return Value

0 on succes, otherwise -1

4.5.36 airlite_get_last_error()

Retreive the last error occurred in this library.

```
char* airlite_get_last_error( AIRLITE_HANDLE handle );
```

4.5.36.1 Parameters

[IN] handle
Handle to an opened connection.

4.5.36.2 Return Value

Pointer to character array containing the last error as a null-terminated C-string.

4.5.37 airlite_logging()

Log all internal activities of the library (like events, errors) to a local text (.txt) file.

```
void airlite_logging(  AIRLITE_HANDLE handle,
                       bool state );
```

4.5.37.1 Parameters

[IN] handle
Handle to an opened connection.

[IN] state
Enable or disable.

4.5.37.2 Return Value

Void.

4.6 Callback

Only one callback function is required to receive all the event and reply messages. Register and unregister a callback function can be done with *airlite_register_callback()* and *airlite_unregister_callback()* respectively.

4.6.1 Prototype

```
typedef void (*AIRLITE_CB)  ( Callback_st *cb, void *userdata );
```

4.6.1.1 Parameters

[IN] cb

Callback data structure.

[IN] userdata

User data pointer or NULL.

4.6.1.2 Return Value

Void.