MediaTek Smart Connection Programming Guide

Version: 1.0

Release date: 29 April 2016

Document Revision History

Revision	Date	Description
1.0	29 April 2016	Initial version.

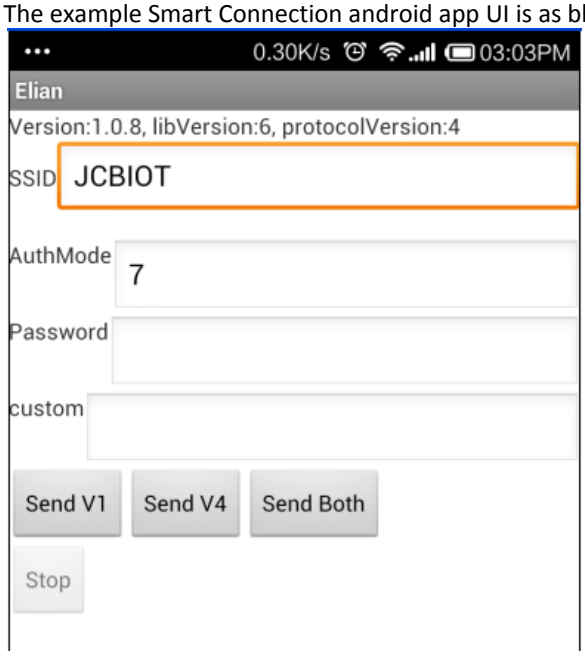
Table of Contents

1. Introduction

As the IoT devices have no user interface, there is no mechanism for users to enter IP configuration details and connect to an AP. To overcome this issue MediaTek “Smart Connection” is provided. It provides a mechanism whereby a wireless network’s information (SSID, password, Authmode, and customer data) is broadcast. When powered up the IoT device listens to detect the broadcast, and can consume it to obtain its wireless network settings. The transmissions are also encrypted with a pre-defined key to ensure security.

The SDK provides the Smart Connection library (libelian.a) for Android and iOS. There is also an example app for both included in the SDK. This section describes the APIs in that library and how to use them.

The example Smart Connection android app UI is as blow.



2. API of Smart Connection

2.1. void elianGetVersion(int *protoVersion, int *libVersion)

Description

This function is used to gets the Smart Connection protocol version and Smart Connection library version

Parameters

[OUT]： protoVersion ---The Smart Connection Protocol version
[OUT]：libVersion ---The Smart Connection Library version.

Return value

None

2.2. void *elianNew(const char *key, int keylen, const unsigned char *target, unsigned int flag)

Description

This function is used to initialize Smart Connection configuration and allocate an Smart Connection context structure

Parameters

[IN]： Key ---The pointer about Encrypt key, all Smart Connection Information include `SSID`, `Password`, `AuthMode`, CustomerTLV) shall be encrypted by this Key, developer can set this item as NULL and all Smart Connection information will be encrypted by a pre-defined key. Developer also can set this item as a 16Bytes data, and then the data shall be used to encrypt all Smart Connection information.
[IN]： keylen ---The Encrypt Key length, if the "`key`" item is set to NULL, "`keylen`" should be 0
[IN]：target --- 6 Bytes value which indicate the MAC address of target device which the Smart Connection packet should be sent to, ext: "0x00 0x0C 0x43 0x23 0xFE 0x00" . If user want to send Smart Connection packet by broadcast, the target item should be set as "0xFF:0xFF:0xFF:0xFF:0xFF:0xFF". If user wants to send Smart Connection packet to a specific device, the "target" item should be set as the MAC address of the specific device. The device will drop the received Smart Connection packet which contains the MAC address is not to me, or not a broadcast MAC address.
[IN]：Flag ---Indicate the Smart Connection protocol version, Smart Connection APP will send the Smart Connection packet with corresponding protocol to the air, there are two options declared in elian.h as blow:

```
#define ELIAN_SEND_V1       0x01
#define ELIAN_SEND_V4       0x02
```

Return value

NULL ---- Smart Connection configuration fail
Other ---- The pointer about Smart Connection Context

Note

If parameter “flag” be set as “ELIAN_SEND_V1”, the value of parameter “Key” and “keylen” will be ignore, and these parameter will be treated as “Key”=NULL, “keylen”=0 in the function of elianNew().

2.3. int elianPut(void *context, enum etype_id id, char *buf, int len)

Description

This function is used to put an object (SSID, Password….) into Smart Connection context structure

Parameters

[IN]：context --- The pointer about Smart Connection context structure, it should be allocated by elianNew() previously
[IN]：id ---The identifier of object, the identifiers are declared in elian.h as below

```
enum etype_id {
        TYPE_ID_BEGIN = 0x0,
        TYPE_ID_AM,             /*auth mode*/
        TYPE_ID_SSID,           /*SSID*/
        TYPE_ID_PWD,            /*Password*/
        TYPE_ID_USER,
        TYPE_ID_PMK,
        TYPE_ID_CUST = 0x7F,    /*Customer TLV*/
        TYPE_ID_MAX = 0xFF
};
```

[IN]：buf --- The buffer value of object
[IN]：len --- The buffer length of object

Return value

0 ---- success
Other ----- fail

Note

If the identifier is “TYPE_ID_AM”, the buf item should be one of the values listed in the Table 1.

Table 1 authentication mode description

Authentication mode	Value	description
OPEN	0x00	Password is null string
WEP	0x00	Password is not null string
SHARED-KEY	0x01	
AUTOSWITCH	0x02	
WPA	0x03	
WPA-PSK	0x04	
WPANONE	0x05	
WPA2	0x06	
WPA2-PSK	0x07	
WPA1WPA2	0x08	
WPA1PSK-WPA2PSK	0x09	

If the identifier is “TYPE_ID_SSID”, the buf item should be max 32 Bytes

If the identifier is “TYPE_ID_PWD”, the buf item should be max 32 Bytes
If the identifier is “TYPE_ID_CUST”, the buf item should be max 32 Bytes, and this is only available when ELIAN_SEND_V4 be selected in `elianNew()` for the “flag” parameter

2.4. int elianStart(void *context)

Description

This function is used to start send Smart Connection packet to the air

Parameters

[IN]：context---The pointer about Smart Connection context structure, it should be allocated by `elianNew()`, and the Smart Connection context content should be filled by `elianPut()` previously

Return value

0 ---- success
Other ----- fail

2.5. void elianStop(void *context)

Description

This function is used to stop send Smart Connection packet to the air

Parameters

[IN]：context ---The pointer about Smart Connection context structure

Return value

0 ---- success

2.6. Other ----- fail void elianDestory(void *context)

Description

This function is used to release Smart Connection context structure

Parameters

[IN]：context ---The pointer about Smart Connection context structure

Return value

0 ---- success
Other ----- fail

3. Sample code of android

When user taps “Send V4”,
The app runs to `Java_com_mediatek_elian_ElianNative_InitSmartConnection()` that calls `elianNew` to allocate a context structure and initialize it.

```
File:  elianjni.cpp

JNIEXPORT jint JNICALL Java_com_mediatek_elian_ElianNative_InitSmartConnection
  (JNIEnv *, jobject, jstring, jint sendV1, jint sendV4)
{
    unsigned char target[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};
    unsigned int flag = 0;

    if (context)
    {
        elianStop(context);
        elianDestroy(context);
        context = NULL;
    }

    if (sendV1)
    {
        flag |= ELIAN_SEND_V1;
    }
    if (sendV4)
    {
        flag |= ELIAN_SEND_V4;
    }
    context = elianNew(NULL, 0, target, flag);
    if (context == NULL)
    {
        return -1;
    }
    return 0;
}
```

Then app will runs to `Java_com_mediatek_elian_ElianNative_StartSmartConnection()` that calls `elianPut()` to filled Elian context content and calls `elianStart()` to begins broadcasting.

```
File:  elianjni.cpp

/*
 * Class:     com_mediatek_elian_ElianNative
 * Method:    StartSmartConnection
 * Signature: (Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;B)I
 */

JNIEXPORT jint JNICALL Java_com_mediatek_elian_ElianNative_StartSmartConnection
  (JNIEnv *env, jobject, jstring SSID, jstring PASSWORD, jstring CUSTOM, jbyte
AUTHMODE)
{
```

```
    const char *ssid = NULL;
    const char *password = NULL;
    const char *custom = NULL;
    unsigned char authmode = AUTHMODE;

    if (context == NULL)
    {
        return -1;
    }
    ssid = env->GetStringUTFChars(SSID, 0);
    password = env->GetStringUTFChars(PASSWORD, 0);
    custom = env->GetStringUTFChars(CUSTOM, 0);

    elianPut(context, TYPE_ID_AM, (char *)&authmode, 1);
    elianPut(context, TYPE_ID_SSID, (char *)ssid, strlen(ssid));
    elianPut(context, TYPE_ID_PWD, (char *)password, strlen(password));
    elianPut(context, TYPE_ID_CUST, (char *)custom, strlen(custom));

    env->ReleaseStringUTFChars(SSID, ssid);
    env->ReleaseStringUTFChars(PASSWORD, password);
    env->ReleaseStringUTFChars(CUSTOM, custom);

    elianStart(context);

    return 0;
}
```

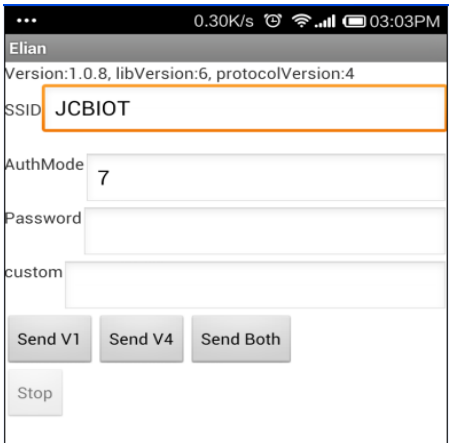
4. Sample code of iOS

When user taps “Send V4”, the app will runs (void)OnSend:(unsigned int)flag, that calls `elianNew()` to allocate Smart Connection context structure, calls `elianPut()` to filled the context content, calls `elianStart()` to begins broadcast

File: ViewController.mm

```
(void)OnSend:(unsigned int)flag {
    const char *ssid = [self.m_ssid.text cStringUsingEncoding:NSASCIIStringEncoding];
    const char *s_authmode = [self.m_authmode.text
cStringUsingEncoding:NSASCIIStringEncoding];
    int authmode = atoi(s_authmode);
    const char *password = [self.m_password.text
cStringUsingEncoding:NSASCIIStringEncoding];
    unsigned char target[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};

    NSLog(@"OnSend: ssid = %s, authmode = %d, password = %s", ssid, authmode,
password);

    if (context)
    {
        elianStop(context);
        elianDestroy(context);
        context = NULL;
    }

    context = elianNew(NULL, 0, target, flag);
    if (context == NULL)
    {
        NSLog(@"OnSend elianNew fail");
        return;
    }

    elianPut(context, TYPE_ID_AM, (char *)&authmode, 1);
    elianPut(context, TYPE_ID_SSID, (char *)ssid, strlen(ssid));
    elianPut(context, TYPE_ID_PWD, (char *)password, strlen(password));

    elianStart(context);
}
```

5. Notes



Elian.APK UI

When user press “Send V1”,”Send V4” or “Send Both” buttons, the packets of specific smart connection version will be send to the air from the device which installed this `Elian.apk`

Smart Connection V1: not support AES Key customization in `elianNew()`, and not support custom content in UI

Smart Connection V4: support AES Key customization in `elianNew()`, and support custom item in UI