MediaTek Smart Connection Programming Guide-Application

Version: 2.0

Release date: 24 February 2017

Document Revision History

Revision	Date	Description
1.0	29 April 2016	Initial version.
2.0	24 February 2017	Add new version support(V5)

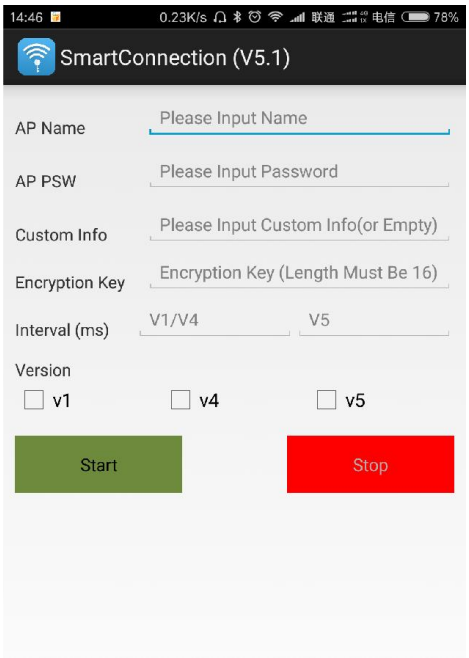
Table of Contents

1. Introduction

As the IoT devices have no user interface, there is no mechanism for users to set IP configurations and connect to an AP. To overcome this issue, MediaTek "Smart Connection" is provided. It provides a mechanism to configure the IoT device which contains the wireless network's information (SSID, password, Authmode and customer data). When power up the IoT device, try the collect the data from the air which is sent out by the Smart Connection APP. The transmissions are also encrypted with a pre-defined key to ensure security.

The SDK provides the Smart Connection library (libsmartconnection.a) for Android and iOS.

The example Smart Connection APP UI is as blow.



(Android)

(iOS)

2. APIs of Smart Connection

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

Description

Get the Smart Connection protocol version and library version.

Parameters

Attribute	Name	Description
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

New Smart Connection context and initialize its configuration.

Parameters

Attribute	Name	Description
IN	key	Pointer to Encryption key, all Smart Connection information (SSID, Password, Authmode, and customer information) should be encrypted by this key. . If the value is NULL, the keylen should be 0, and the information should be encrypted by pre-defined key . If the value is not NULL, the value length should be 16 bytes, and the keylen is 16, all the information should be encrypted by this key
IN	keylen	The encryption key length . if the key is NULL, the value should be 0
IN	target	6 bytes to indicate the MAC address which device wish to receive and handle the Smart Connection Packets.

Attribute	Name	Description
		. If the value is "0xff 0xff 0xff 0xff 0xff 0xff", the packet will be handled by all devices . If the value is NOT "0xff", the packets should ONLY be handled by the target device which MAC address is equal to target, and other devices should drop the received packets . The value is ONLY available for version V1 and V4
IN	flag	Indicate the Smart Connection Protocol version, Smart Connection APP should send the corresponding packets to the air . If the value is "ELIAN_VERSION_V1", the "key" and "keylen" will be ignored. They both will be treated as "NULL" and 0. And the Smart Connection information will be encrypted by pre-defined key. . The flag can be combined with each other e.g. : flag = ELIAN_VERSION_V1 | ELIAN_VERSION_V4 or ELIAN_VERSION_V4 | ELIAN_VERSION_V5

Note: flag defined in elian.h as below:

```
#define ELIAN_VERSION_V1            0x01
#define ELIAN_VERSION_V4            0x02
#define ELIAN_VERSION_V5            0x04
```

Return value

Success	Not NULL
Fail	NULL

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

Description

Put value into Smart Connection Context which will be sent to IoT device.

Parameters

Attribute	Name	Description
IN	context	Pointer to Smart Connection context which has be newed previously
IN	id	Type id of the configuration to set
IN	buf	The buffer value of the object
IN	len	The buffer length of the object

Note:
enum type id definition 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 Data*/
        TYPE_ID_MAX = 0xFF
    };
```

Return value

Success	ELIAN_ERROR_CODE_OK
Fail	Others

Note

If the identifier is “TYPE_ID_AM”, the buf item should be one of the values listed in the Table 1.
The “TYPE_ID_AM” is ONLY available for version V1 and V4.

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 32/64 Bytes (V1, V4 32 bytes, V5 64 bytes)
If the identifier is “TYPE_ID_CUST”, the buf item should be max 32 Bytes, the identifier is ONLY available when “ELIAN_VERSION_V4” and “ELIAN_VERSION_V5” is set.

2.4. int elianSetInterval(void *context, unsigned int ousec, unsigned int nusec)

Description

Set the sending interval for each packet.

Parameters

Attribute	Name	Description

Attribute	Name	Description
IN	context	Pointer to Smart Connection context which has be newed previously
IN	ousec	Send packet interval for V1/V4
IN	nusec	Send packet interval for V5

Return value

Success	ELIAN_ERROR_CODE_OK
Fail	Others

2.5. int elianStart(void *context)

Description

Start to send the corresponding Smart Connection packets into air.

Parameters

Attribute	Name	Description
IN	context	Pointer to Smart Connection context which has be newed previously

Return value

Success	ELIAN_ERROR_CODE_OK
Fail	Others

2.6. void elianStop(void *context)

Description

Stop to send Smart Connection Packets.

Parameters

Attribute	Name	Description
IN	context	Pointer to Smart Connection context which has be newed previously

Return value

None

2.7. void elianDestory(void *context)

Description

Destroy the Smart Connection Context, free memory, clear flags …

Parameters

Attribute	Name	Description
IN	context	Pointer to Smart Connection context which has be newed previously

Return value

None

3. Error Codes

Value	Macro	Description
0	**ELIAN_ERROR_CODE_OK**	Succeed to execute operation
-1	**ELIAN_ERROR_CODE_NOT_INITED**	The Context not initialized
-2	**ELIAN_ERROR_CODE_WRONG_TYPE**	Wrong type to set
-3	**ELIAN_ERROR_CODE_WRONG_PARAMETER**	Wrong parameter to configure
-4	**ELIAN_ERROR_CODE_CRYPTED_FAILED**	Encrypted failed
-5	**ELIAN_ERROR_CODE_NOT_ENOUGH_MEMORY**	Not enough memory
-6	**ELIAN_ERROR_CODE_EXCEED_MAX_LENGTH**	The configuration exceeds the max length
-7	**ELIAN_ERROR_CODE_ALREADY_STARTED**	The context has been started
-8	**ELIAN_ERROR_CODE_ALREADY_STOPED**	The context has been stopped
-9	**ELIAN_ERROR_CODE_FAILED_TO_START**	Common error : failed to start the context

Please refer to elian.h.

4. Sample code of android

JniLoader.java: Class to interact with JNI library. Defined the error codes and public APIs mapping to the JNI library.

```
File: JniLoader.java

public class JniLoader {

    public static final int ERROR_CODE_OK                    = 0;
    public static final int ERROR_CODE_NOT_INITINED          = -1;
    public static final int ERROR_CODE_WRONG_TYPE            = -2;
    public static final int ERROR_CODE_WRONG_PARAMETER       = -3;
    public static final int ERROR_CODE_CRYPTED_FAILED        = -4;
    public static final int ERROR_CODE_NOT_ENOUGH_MEMORY     = -5;
    public static final int ERROR_CODE_EXCEED_MAX_LENGTH     = -6;
    public static final int ERROR_CODE_ALREADY_STARTED       = -7;
    public static final int ERROR_CODE_ALREADY_STOPED        = -8;
    public static final int ERROR_CODE_FAILED_TO_START       = -9;

    public JniLoader() {

    }

    public static boolean LoadLib() {
        try {
            System.loadLibrary("smart_connection_jni");
            return true;
        } catch (Exception ex) {
            System.err.println("WARNING: Could not load library");
            return false;
        }
    }

    public native int GetProtoVersion();
    public native int GetLibVersion();

    /**
     * Init SmartConnection
     */
    public native int InitSmartConnection(String key, String Target, int
sendV1, int sendV4, int sendV5);

    /**
     * Set Send interval
     * @param oldInterval
     * @param newInterval
     * @return
     */
    public native int SetSendInterval(float oldInterval, float newInterval);

    /**
     * Start SmartConnection
     *
     * @SSID : SSID of Home AP
     * @Password : Password of Home AP
     */
```

```
    public native int StartSmartConnection(String SSID, String Password,
String Custom);

    /**
     * Stop SmartConnection
     *
     */
    public native int StopSmartConnection();
}
```

smart_connection_jni_loader.c: Implement the JNI APIs from JniLoader.java, call SmartConnection library to implement features.

Default send the data to all devices, the target value is {0xff, 0xff, 0xff, 0xff, 0xff, 0xff}. The target is only available for V1, V4 or V1|V4.

```
File:  smart_connection_jni_loader.c

/*
 * Class:     com_mediatek_demo_smartconnection_JniLoader
 * Method:    InitSmartConnection
 * Signature: (Ljava/lang/String;III)I
 */
JNIEXPORT jint JNICALL
Java_com_mediatek_demo_smartconnection_JniLoader_InitSmartConnection
  (JNIEnv *env, jobject object, jstring key, jstring tag, jint sendV1, jint
sendV4, jint sendV5)
{
    unsigned char   target[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};
    unsigned int    flag = 0;
    const char      *encryp_key = NULL;

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

    if (sendV1)
    {
        flag |= SEND_V1;
    }
    if (sendV4)
    {
        flag |= SEND_V4;
    }
    if (sendV5) {
        flag |= SEND_V5;
    }
    encryp_key = (*env)->GetStringUTFChars(env, key, 0);
    if (encryp_key != NULL && strlen(encryp_key) == 16) {
        context = elianNew(encryp_key, strlen(encryp_key), target, flag);
    } else {
        context = elianNew(NULL, 0, target, flag);
    }
```

```
    (*env)->ReleaseStringUTFChars(env, key, encryp_key);
    if (context == NULL)
    {
        return ELIAN_ERROR_CODE_NOT_INITED;
    }
    //elianSetInterval(context, 20*1000);
    return ELIAN_ERROR_CODE_OK;
}

/*
 * Class:     com_mediatek_demo_smartconnection_JniLoader
 * Method:    StartSmartConnection
 * Signature: (Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I
 */
JNIEXPORT jint JNICALL
Java_com_mediatek_demo_smartconnection_JniLoader_StartSmartConnection
  (JNIEnv *env, jobject object, jstring SSID, jstring PASSWORD, jstring
CUSTOM)
{
    const char *ssid = NULL;
    const char *password = NULL;
    const char *custom = NULL;
    int retValue = ELIAN_ERROR_CODE_OK;

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

//    elianPut(mcContext, TYPE_ID_AM, (char *)&authmode, 1);
    retValue = elianPut(context, TYPE_ID_SSID, (char *)ssid, strlen(ssid));
    if (retValue != ELIAN_ERROR_CODE_OK) {
        goto error;
    }
    retValue = elianPut(context, TYPE_ID_PWD, (char *)password,
strlen(password));
    if (retValue != ELIAN_ERROR_CODE_OK) {
        goto error;
    }
    retValue = elianPut(context, TYPE_ID_CUST, (char *)custom,
strlen(custom));
    if (retValue != ELIAN_ERROR_CODE_OK) {
        goto error;
    }

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

    retValue = elianStart(context);

    return retValue;
}
```

5. Sample code of iOS

Because the library used the STL features, so please modify the file suffix from m to mm which one used the library. Otherwise will build fail.

```
File:  SmartConnectionTableViewController.mm

#pragma mark - start button handle action
- (IBAction)startButtonAction:(id)sender {

    if ([_versionV5 isOn] == NO
        && [_versionV1 isOn] == NO
        && [_versionV4 isOn] == NO) {
        [self showWarningAlertDialog:@"Please select at least 1 version to
send"];
        return;
    }

    const char *ssid = NULL;
    const char *pwd = NULL;
    const char *cinfo = NULL;
    float oi = 0.0f;
    float ni = 0.0f;
    int retValue = ELIAN_ERROR_CODE_OK;
    int flag = 0;

    NSLog(@"SSID                    : %@", [_wifiApName text]);
    NSLog(@"Password                : %@", [_wifiApPassword text]);

    if ([_versionV1 isOn] == YES) {
        flag |= ELIAN_VERSION_V1;
    }
    if ([_versionV4 isOn] == YES) {
        flag |= ELIAN_VERSION_V4;
    }
    if ([_versionV5 isOn] == YES) {
        flag |= ELIAN_VERSION_V5;
    }
    unsigned char target[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};
    context = elianNew(NULL, 0, target, flag);
    if (context == NULL) {
        [self showWarningAlertDialog:@"Failed to init"];
        return;
    }
    if ([[_wifiApName text]isEqualToString:@""] == NO) {
        ssid = [[_wifiApName text] UTF8String];
        retValue = elianPut(context, TYPE_ID_SSID, (char *)ssid,
(int)strlen(ssid));
    }
    if (retValue != ELIAN_ERROR_CODE_OK) {
        [self showWarningAlertDialog:@"Failed to set ssid"];
        elianDestroy(context);
        return;
    }
    if ([[_wifiApPassword text]isEqualToString:@""] == NO) {
```

```
        pwd = [[_wifiApPassword text] UTF8String];
        retValue = elianPut(context, TYPE_ID_PWD, (char *)pwd,
(int)strlen(pwd));
    }
    if (retValue != ELIAN_ERROR_CODE_OK) {
        [self showWarningAlertDialog:@"Failed to set password"];
        elianDestroy(context);
        return;
    }
    if ([[_customInfo text]isEqualToString:@""] == NO) {
        cinfo = [[_customInfo text] UTF8String];
        retValue = elianPut(context, TYPE_ID_CUST, (char *)cinfo,
(int)strlen(cinfo));
    }
    if (retValue != ELIAN_ERROR_CODE_OK) {
        [self showWarningAlertDialog:@"Failed to set customer info"];
        elianDestroy(context);
        return;
    }

    if ([[_oldversioninterval text] isEqualToString:@""] == NO) {
        oi = [[_oldversioninterval text] floatValue];
    }
    if ([[_nversionInterval text] isEqualToString:@""] == NO) {
        ni = [[_nversionInterval text] floatValue];
    }
    if (oi != 0.0 || ni != 0.0) {
        retValue = elianSetInterval(context, oi * 1000, ni * 1000);
    }

    if (retValue != ELIAN_ERROR_CODE_OK) {
        [self showWarningAlertDialog:@"Failed to set interval"];
        elianDestroy(context);
        return;
    }

    retValue = elianStart(context);
    if (retValue != ELIAN_ERROR_CODE_OK) {
        [self showWarningAlertDialog:@"Failed to set interval"];
        return;
    }

    [_startButton setEnabled:NO];
    [_stopButton setEnabled:YES];
    [_runningIndicator startAnimating];
    [_wifiApName setEnabled:NO];
    [_wifiApPassword setEnabled:NO];
    [_customInfo setEnabled:NO];
    [_versionV4 setEnabled:NO];
    [_versionV1 setEnabled:NO];
    [_versionV5 setEnabled:NO];
    [_nversionInterval setEnabled:NO];
    [_oldversioninterval setEnabled:NO];

}
```