Simple-Pair
User Guide

Version 1.0

About This Guide

This document introduces Simple-Pair technology developed by Espressif. The document focuses on ESP-NOW features, how to use it and the demo code.

The structure is as below:

Chapter	Title	Subject
Chapter 1	Simple-Pair Introduction	Introduction on Simple-Pair technology.
Chapter 2	Simple-Pair Usage Guide	Guidance on how to use Simple-Pair.
Chapter 3	Solution	Provision of an example solution.
Chapter 4	Demo Code	Provision of demo code.

Release Notes

Date	Version	Release notes
2016.07	V1.0	First release.

Table of Contents

1. Simple-Pair Introduction

Based on ESP IE, Simple-Pair is used for fast negotiation or to exchange Key (or other information) between two devices.

Simple-Pair enables Key exchange with only a few steps. For now, Simple-Pair supports Key exchange between Station and AP, but the Key is limited to 16 bytes. The Key will finally be sent from AP to Station. Simple-Pair contains a Key named TempKey in order to ensure the final exchange step. TempKey is stored at the Station end and should be acknowledged by the Station and the AP before Simple-Pair.

Simple Pair is designed for fast exchange of Key and for realizing pairing between two devices that have never been paired before. It is recommended to disable the paired devices' sleep mode temporarily in case of packet transmission failure. The key to be exchanged is stored at the AP end first and will then be sent from AP to Station finally, at which point Simple-Pair has been completed for the devices. However, for the application layer, pairing has finished only when the Key is verified valid after trial.

> 📖 ***Note:***
>
> *Smart phone can be used during Simple-Pair. Chapter 3 provides an example.*

2. Simple-Pair Usage Guide

There are basically only three steps for Simple-Pair: Set Peer Info, AP ANNOUNCE (STA SCAN) and START Negotiate. All statuses are indicated by the callback function.

Users need to register the status callback function to indicate a status during Simple-Pair, such as paring completed status, negotiation request, timeout, error, etc. For example, the status callback function indicates a negotiation request from the Station so that the AP can decide whether or not to accept. If the status callback function returns error, then action has to be taken according to the value (for details, please refer to SDK_V2.0 Simple Pair Demo).

The AP needs to enable Simple-Pair Announce mode and the Station needs to enable Simple-Pair Scan mode before Simple-Pair. Then, the Station will send IEEE802.11 Probe Request to AP, i.e., scan APs, so that both the Station and the AP would know which is ready for Simple-Pair.

The Station will select the AP that is ready for Simple-Pair from BSS_INFO, set MAC address and TempKey of the AP, then send STA Negotiate Start.

Upon receiving the request, the AP will decide (or decide with smart phone) whether or not to accept the request. If agreed to, the AP would set the MAC address, TempKey and the exchange Key for the Station, then send AP Negotiate Start; otherwise, the AP will send AP Negotiate Refuse.

If the Station receive AP Negotiate Start, negotiation will start; if the Station receives AP Negotiate Refuse, it will inform the application layer through status callback function and the latter will respond accordingly.

After negotiation, the Station will get the exchange Key from the AP, which means Simple-Pair has completed. The application layer needs to make sure the Key is correct through verification.

> ⚠ ***Notice:***
>
> *Before negotiation, the application layer needs to make sure no MAC address has been set to a Key by ESP-NOW (or by other function that may be developed in the future). If there is, please delete the function, otherwise, wrong Key may be used in Simple-Pair.*

3. Solution

The following is an example of exchanging ESP-NOW's with Simple-Pair.

> 📖 ***Note:***
>
> *Besides exchanging ESP-NOW's Key, Simple-Pair can also be used to exchange other function's Key.*

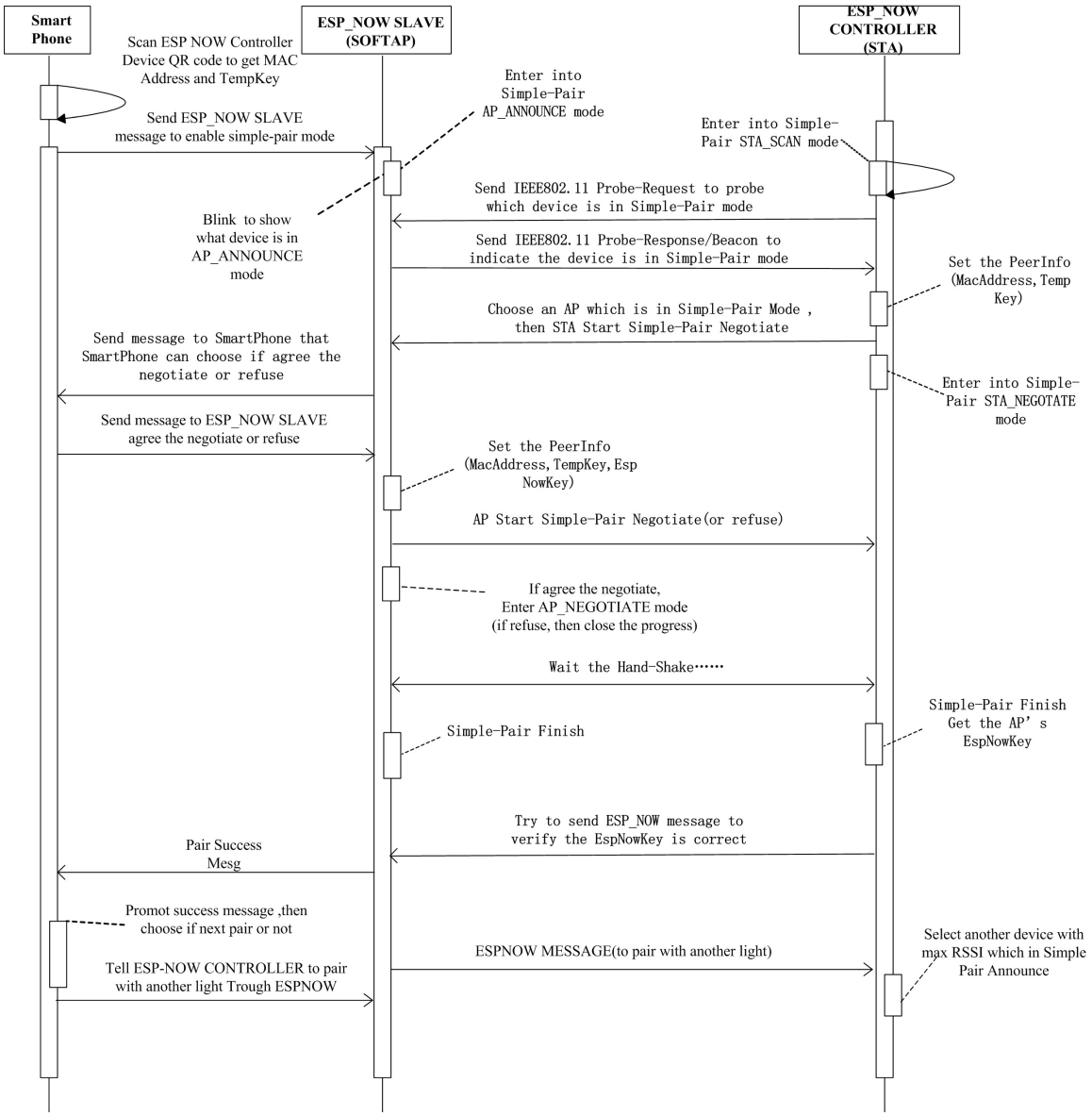

Figure 3-1. Example Solution

After Simple-Pair has finished, please verify the Key exchanged. Smart phone can control part of the process. If Simple-Pair is successful, smart phone can be used to decide if the SLAVE pairs with another CONTROLLER or not.

4. Demo Code

> 📖 ***Note:***
> *For more information on Simple-Pair APIs, please refer to ESP8266 Non-OS SDK API Guide.*

```
#include "osapi.h"
#include "user_interface.h"



#include "simple_pair.h"


/
******************************************************************
**********
 * open AS_STA to compile sta test code
 * open AS_AP to compile ap test code
 * don't open both
 *
******************************************************************
*******/
//#define AS_STA
#define AS_AP


/
******************************************************************
*********
 * FunctionName : user_rf_cal_sector_set
 * Description  : SDK just reversed 4 sectors, used for rf init data
and paramters.
 *                We add this function to force users to set rf cal
sector, since
 *                we don't know which sector is free in user's
application.
 *                sector map for last several sectors : ABCCC
 *                A : rf cal
 *                B : rf init data
 *                C : sdk parameters
```

```
 * Parameters   : none
 * Returns      : rf cal sector
*******************************************************************
**********/
uint32 ICACHE_FLASH_ATTR
user_rf_cal_sector_set(void)
{
    enum flash_size_map size_map = system_get_flash_size_map();
    uint32 rf_cal_sec = 0;

    switch (size_map) {
        case FLASH_SIZE_4M_MAP_256_256:
            rf_cal_sec = 128 - 5;
            break;

        case FLASH_SIZE_8M_MAP_512_512:
            rf_cal_sec = 256 - 5;
            break;

        case FLASH_SIZE_16M_MAP_512_512:
        case FLASH_SIZE_16M_MAP_1024_1024:
            rf_cal_sec = 512 - 5;
            break;

        case FLASH_SIZE_32M_MAP_512_512:
        case FLASH_SIZE_32M_MAP_1024_1024:
            rf_cal_sec = 1024 - 5;
            break;

        default:
            rf_cal_sec = 0;
            break;
    }

    return rf_cal_sec;
```

```
}



void ICACHE_FLASH_ATTR
user_rf_pre_init(void)
{
}
/* STA & AP use the same tmpkey to encrypt Simple Pair communication
*/
static u8 tmpkey[16] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
0x07,
                  0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f};


#ifdef AS_STA
/* since the ex_key transfer from AP to STA, so STA's ex_key don't
care */
static u8 ex_key[16] = {0x00};
#endif /* AS_STA */


#ifdef AS_AP
/* since the ex_key transfer from AP to STA, so AP's ex_key must be
set */
static u8 ex_key[16] = {0xff, 0xee, 0xdd, 0xcc, 0xbb, 0xaa, 0x99,
0x88,
                  0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11, 0x00};
#endif /* AS_AP */


void ICACHE_FLASH_ATTR
show_key(u8 *buf, u8 len)
{
      u8 i;


      for (i = 0; i < len; i++)
            os_printf("%02x,%s", buf[i], (i%16 == 15?"\n":" "));
}
```

```
#ifdef AS_STA
static void ICACHE_FLASH_ATTR
scan_done(void *arg, STATUS status)
{
    int ret;

    if (status == OK) {

        struct bss_info *bss_link = (struct bss_info *)arg;

        while (bss_link != NULL) {
          if (bss_link->simple_pair) {
                os_printf("Simple Pair: bssid %02x:%02x:%02x:%02x:
%02x:%02x Ready!\n",
                          bss_link->bssid[0], bss_link->bssid[1],
bss_link->bssid[2],
                          bss_link->bssid[3], bss_link->bssid[4],
bss_link->bssid[5]);
            simple_pair_set_peer_ref(bss_link->bssid, tmpkey, NULL);
            ret = simple_pair_sta_start_negotiate();
            if (ret)
                  os_printf("Simple Pair: STA start NEG Failed\n");
            else
                  os_printf("Simple Pair: STA start NEG OK\n");
            break;
          }
            bss_link = bss_link->next.stqe_next;
        }
    } else {
        os_printf("err, scan status %d\n", status);
    }

}
#endif
```

```
void ICACHE_FLASH_ATTR
sp_status(u8 *sa, u8 status)
{
#ifdef AS_STA
      switch (status) {
      case  SP_ST_STA_FINISH:
            simple_pair_get_peer_ref(NULL, NULL, ex_key);
            os_printf("Simple Pair: STA FINISH, Ex_key ");
            show_key(ex_key, 16);
            /* TODO: Try to use the ex-key communicate with AP, for
example use ESP-NOW */

            /* if test ok , deinit simple pair */
            simple_pair_deinit();
            break;
      case SP_ST_STA_AP_REFUSE_NEG:
            /* AP refuse , so try simple pair again  or scan other
ap*/
            os_printf("Simple Pair: Recv AP Refuse\n");
            simple_pair_state_reset();
            simple_pair_sta_enter_scan_mode();
            wifi_station_scan(NULL, scan_done);
            break;
      case SP_ST_WAIT_TIMEOUT:
            /* In negotiate, timeout , so try simple pair again */
            os_printf("Simple Pair: Neg Timeout\n");
            simple_pair_state_reset();
            simple_pair_sta_enter_scan_mode();
            wifi_station_scan(NULL, scan_done);
            break;
      case SP_ST_SEND_ERROR:
            os_printf("Simple Pair: Send Error\n");
            /* maybe the simple_pair_set_peer_ref() haven't called,
it send to a wrong mac address */
```

```
            break;
      case SP_ST_KEY_INSTALL_ERR:
            os_printf("Simple Pair: Key Install Error\n");
            /* 1. maybe something argument error.
               2. maybe the key number is full in system*/

            /* TODO: Check other modules which use lots of keys
                        Example: ESPNOW and STA/AP use lots of keys
*/
            break;
      case SP_ST_KEY_OVERLAP_ERR:
            os_printf("Simple Pair: Key Overlap Error\n");
            /* 1. maybe something argument error.
               2. maybe the MAC Address is already use in ESP-NOW or
other module
                  the same MAC Address has multi key*/

            /* TODO: Check if the same MAC Address used already,
                        Example: del MAC item of ESPNOW or other
module */
            break;
      case SP_ST_OP_ERROR:
            os_printf("Simple Pair: Operation Order Error\n");
            /* 1. maybe the function call order has something wrong
*/

            /* TODO: Adjust your function call order */
            break;
      default:
            os_printf("Simple Pair: Unknown Error\n");
            break;
      }

#endif /* AS_STA */
```

```
#ifdef AS_AP
      switch (status) {
      case  SP_ST_AP_FINISH:
            simple_pair_get_peer_ref(NULL, NULL, ex_key);
            os_printf("Simple Pair: AP FINISH\n");

            /* TODO: Wait STA use the ex-key communicate with AP, for
example use ESP-NOW */

            /* if test ok , deinit simple pair */
            simple_pair_deinit();
            break;
      case SP_ST_AP_RECV_NEG:
            /* AP recv a STA's negotiate request */
            os_printf("Simple Pair: Recv STA Negotiate Request\n");

            /* set peer must be called, because the simple pair need
to know what peer mac is */
            simple_pair_set_peer_ref(sa, tmpkey, ex_key);

            /* TODO:In this phase, the AP can interaction with Smart
Phone,
                  if the Phone agree, call start_neg or refuse */
            simple_pair_ap_start_negotiate();
            //simple_pair_ap_refuse_negotiate();
            /* TODO:if refuse, maybe call simple_pair_deinit() to
ending the simple pair */

            break;
      case SP_ST_WAIT_TIMEOUT:
            /* In negotiate, timeout , so re-enter in to announce
mode*/
            os_printf("Simple Pair: Neg Timeout\n");
            simple_pair_state_reset();
            simple_pair_ap_enter_announce_mode();
            break;
```

```
        case SP_ST_SEND_ERROR:
            os_printf("Simple Pair: Send Error\n");
            /* maybe the simple_pair_set_peer_ref() haven't called,
it send to a wrong mac address */

            break;
        case SP_ST_KEY_INSTALL_ERR:
            os_printf("Simple Pair: Key Install Error\n");
            /* 1. maybe something argument error.
               2. maybe the key number is full in system*/

            /* TODO: Check other modules which use lots of keys
                        Example: ESPNOW and STA/AP use lots of keys
*/
            break;
        case SP_ST_KEY_OVERLAP_ERR:
            os_printf("Simple Pair: Key Overlap Error\n");
            /* 1. maybe something argument error.
               2. maybe the MAC Address is already use in ESP-NOW or
other module
                  the same MAC Address has multi key*/

            /* TODO: Check if the same MAC Address used already,
                        Example: del MAC item of ESPNOW or other
module */
            break;
        case SP_ST_OP_ERROR:
            os_printf("Simple Pair: Operation Order Error\n");
            /* 1. maybe the function call order has something wrong
*/

            /* TODO: Adjust your function call order */
            break;
        default:
            os_printf("Simple Pair: Unknown Error\n");
            break;
```

```
        }

#endif /* AS_AP */
}

void ICACHE_FLASH_ATTR
init_done(void)
{
        int ret;

#ifdef AS_STA
        wifi_set_opmode(STATION_MODE);

        /* init simple pair */
        ret = simple_pair_init();
        if (ret) {
                os_printf("Simple Pair: init error, %d\n", ret);
                return;
        }
        /* register simple pair status callback function */
        ret = register_simple_pair_status_cb(sp_status);
        if (ret) {
                os_printf("Simple Pair: register status cb error, %d\n",
ret);
                return;
        }

        os_printf("Simple Pair: STA Enter Scan Mode ...\n");
        ret = simple_pair_sta_enter_scan_mode();
        if (ret) {
                os_printf("Simple Pair: STA Enter Scan Mode Error, %d\n",
ret);
                return;
        }
        /* scan ap to searh which ap is ready to simple pair */
```

```
        os_printf("Simple Pair: STA Scan AP ...\n");
          wifi_station_scan(NULL,scan_done);
#endif
#ifdef AS_AP
        wifi_set_opmode(SOFTAP_MODE);

        /* init simple pair */
        ret = simple_pair_init();
        if (ret) {
              os_printf("Simple Pair: init error, %d\n", ret);
              return;
        }
        /* register simple pair status callback function */
        ret = register_simple_pair_status_cb(sp_status);
        if (ret) {
              os_printf("Simple Pair: register status cb error, %d\n",
ret);
              return;
        }

        os_printf("Simple Pair: AP Enter Announce Mode ...\n");
        /* ap must enter announce mode , so the sta can know which ap
is ready to simple pair */
        ret = simple_pair_ap_enter_announce_mode();
        if (ret) {
              os_printf("Simple Pair: AP Enter Announce Mode Error, %d
\n", ret);
              return;
        }

#endif

}
```

```
void ICACHE_FLASH_ATTR
user_init(void)
{
      system_init_done_cb(init_done);
}
```

Espressif IOT Team

www.espressif.com