Enterprise Client Application

User guide

Version 0.2

May 2016

Redpine Signals, Inc.
2107 N. First Street, #540
San Jose, CA 95131.
Tel: (408) 748-3385
Fax: (408) 705-2019
Email: info@redpinesignals.com
Website: www.redpinesignals.com

About this Document

This document describes the process of bringing up the RS9113 based module as an Enterprise client and connects with Enterprise secured AP.

Disclaimer:

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only. Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

Table of Contents

Table of Figures

Table of Tables

No table of figures entries found.

1 Introduction

This project is applicable to all the WiSeConnect variants like WiSeConnect Plus, WiSeMCU and WyzBee. The term WiSeConnect refers to its appropriate variant.

1.1 Application Overview

1.1.1 Overview

This Application demonstrates how to configure device in Enterprise client and connects with Enterprise secured AP and data traffic in Enterprise security mode.

In this application, WiSeConnect device connects to Enteroise secured AP using EAP-TLS/TTLS/PEAP/FAST method. After successful connection, Application established TCP client connection with TCP server opened on remote peer and sends TCP data on opened socket.

1.1.2 EAP overview

In wireless communications using EAP, a user requests connection to a WLAN through an AP, which then requests the identity of the user and transmits that identity to an authentication server such as RADIUS. The server asks the AP for proof of identity, which the AP gets from the user and then sends back to the server to complete the authentication.

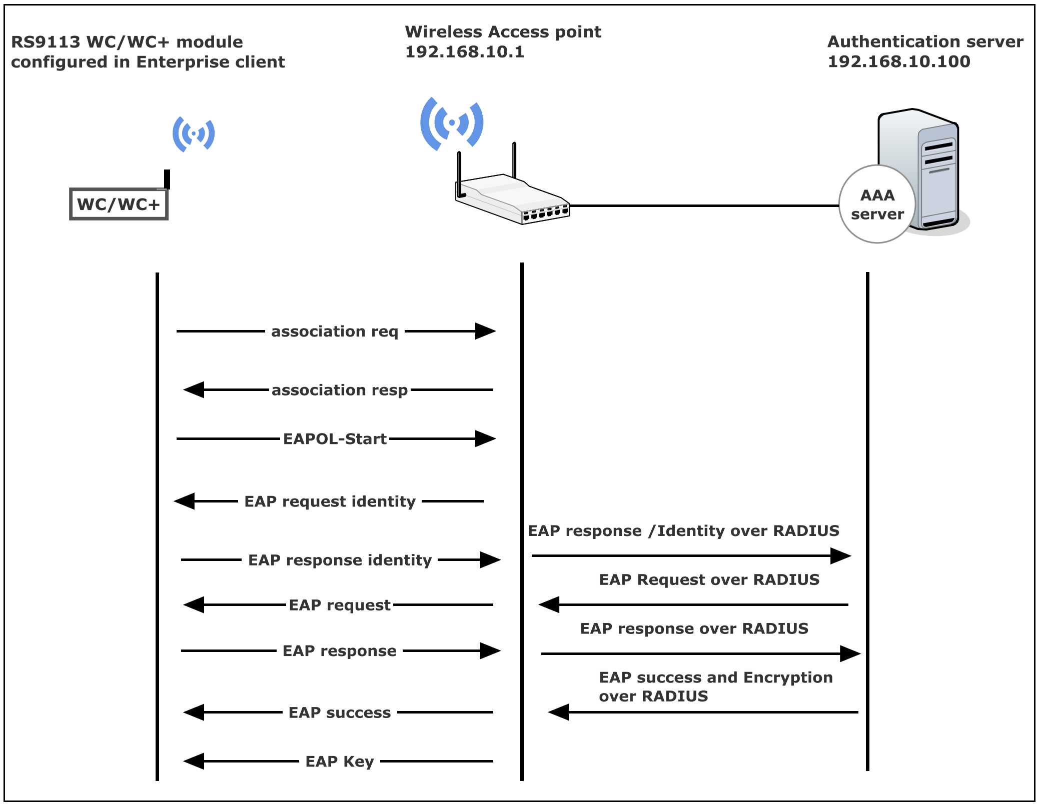


Figure 1: EAPOL-Keys exchange

1.1.3 Sequence of Events

This Application explains user how to:

- Configure device as an Enterprise client
- Connect with Enterprise secured AP using EAP-TLS/TTLS/PEAP/FAST method
- Establish TCP connection from connected WiSeConnect device to TCP server opened on remote peer.
- Send TCP data from WiSeConnect device to remote peer.

1.2 Application Setup

The WiSeConnect in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

1.2.1 SPI based Setup Requirements

- Windows PC with CooCox IDE
- Spansion (MB9BF568NBGL) micro controller

Note: If user does not have Spansion (MB9BF568NBGL) host platform, please go through the SPI-Porting guide **\sapis\docs\RS9113-WiSeConnect-SAPI-Porting-Guide-vx.x.pdf** for SAPIs porting to that particular platform.

- WiSeConnect device
- Windows/Linux PC2 with AAA Radius Server or Free Radius server
- TCP server application running on Windows/Linux PC2 (This example uses iperf for windows).

1.2.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- Windows/Linux PC2 with AAA Radius Server or Free Radius server
- TCP server application running on Windows/Linux PC2 (This example uses iperf for windows).

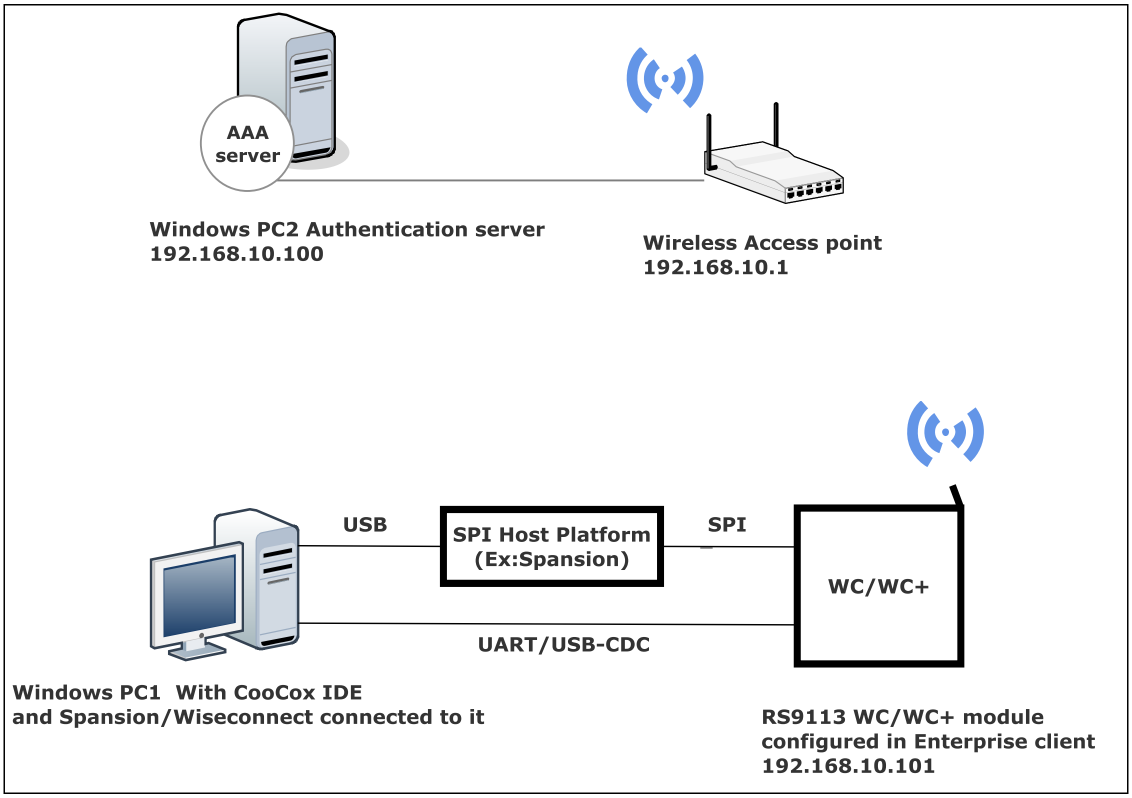


Figure 2: Setu0p Diagram

2 Configuration and Execution of the Application

The example application is available in the Release at **{Release $}/host/sapis/examples.** These examples will have to be initialized, configured and executed to test the application. The initialization varies based on the interface but configuration and execution are the common.

2.1 Initializing the Application

2.1.1 SPI Interface

If User using SPI interface, Please refer the document ***sapis/platforms/spansion_MB9BF568NBGL/RS9113-WiSeConnect_SAPIS_Spansion_Project_User_guide.pdf*** for opening the ***eap*** example in CooCox IDE.

2.1.2 UART/USB-CDC Interface

If User using UART interface, Please refer the document ***sapis/platforms/ windows_uart/RS9113-WiSeConnect_SAPIS_Windows_Project_UserGuide.pdf*** for opening the ***eap*** example in Dev-C++ IDE

2.2 Configuring the Application

1. Open ***sapis/examples/eap/rsi_eap_connectivity.c*** fileand update/modify following macros

SSID refers to the name of the Access point.

```
      #define SSID                  "REDPINE_AP"
```

SECURITY_TYPE refers to the type of security. In In this application STA supports WPA-EAP, WPA2-EAP securities.

Valid configuration is:

RSI_WPA_EAP - For WPA-EAP security mode

RSI_WPA2_EAP - For WPA2-EAP security mode

```
      #define SECURITY_TYPE         RSI_WPA2_EAP
```

To Load certificate

LOAD_CERTIFICATE refers whether certificate to load into module or not.

```
      #define  LOAD_CERTIFICATE          1
```

If **LOAD_CERTIFICATE** set to 1, application will load certificate which is included using rsi_wlan_set_certificate API.

By default, application is loading "wifiuser.pem" certificate when LOAD**_CERTIFICATE** enable. In order to load different certificate, user has to do the following steps:

- **rsi_wlan_set_certificate** API expects the certificate in the form of linear array. So, convert the pem certificate into linear array form using python script provided in the release package **"sapis/examples/utilities/certificates/certificate_script.py"**

Ex: If the certificate is wifi-user.pem .Give the command in the following way :

```
python certificate_script.py wifi-user.pem
```

Script will generate wifiuser.pem in which one linear array named wifiuser contains the certificate.

- After conversion of certificate, update ***rsi_eap_connectivity.c*** source file by including the certificate file and by providing *the* required parameters to **rsi_wlan_set_certificate** API.
  - Once certificate loads into the device, it will write into the device flash. So, user need not load certificate for every boot up unless certificate change.

    So define `LOAD_CERTIFICATE` as 0, if certificate is already present in the device.

`USER_IDENTITY` refers to user ID which is configured in the user configuration file of the radius server. In this example, user identity is “`user1`”.

```
      #define USER_IDENTITY          "\"user1\""
```

PASSWORD refers to the password which is configured in the user configuration file of the Radius Server for that User Identity.

In this example, password is “`test123`”

```
      #define PASSWORD               "\"test123\""
```

`DEVICE_PORT` port refers TCP client port number

```
      #define DEVICE_PORT       5001
```

`SERVER_PORT` port refers remote TCP server port number which is opened in Windows PC2.

```
      #define SERVER_PORT       5001
```

`SERVER_IP_ADDRESS` refers remote peer IP address to connect with TCP server socket.

IP address should be in long format and in little endian byte order.

Example: To configure “192.168.0.100” as remote IP address, update the macro **`SERVER_IP_ADDRESS`** as **`0x6400A8C0`**.

```
      #define SERVER_IP_ADDRESS      0x640AA8C0
```

`NUMEBR_OF_PACKETS` refers how many packets to receive from TCP client

```
      #define NUMBER_OF_PACKETS      1000
```

To configure IP address in STA mode

`DHCP_MODE` refers whether IP address configured through DHCP or STATIC in STA mode

```
      #define DHCP_MODE   1
```

Note: If the user wants to configure STA IP address through DHCP then skip configuring the following **`DEVICE_IP`**, **`GATEWAY`** and **`NETMASK`** macros.

(Or)

If the user wants to configure STA IP address through STATIC then set **DHCP_MODE** macro to "0" and configure following **DEVICE_IP, GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.0.10" as IP address, update the macro **DEVICE_IP** as **0x010AA8C0**.

```
      #define  DEVICE_IP          0X0A00A8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

Example: To configure "192.168.0.1" as Gateway, update the macro GATEWAY as **0x0100A8C0**

```
      #define  GATEWAY            0x0100A8C0
```

IP address of the network mask should also be in long format and in little endian byte order.

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
      #define   NETMASK            0x00FFFFFF
```

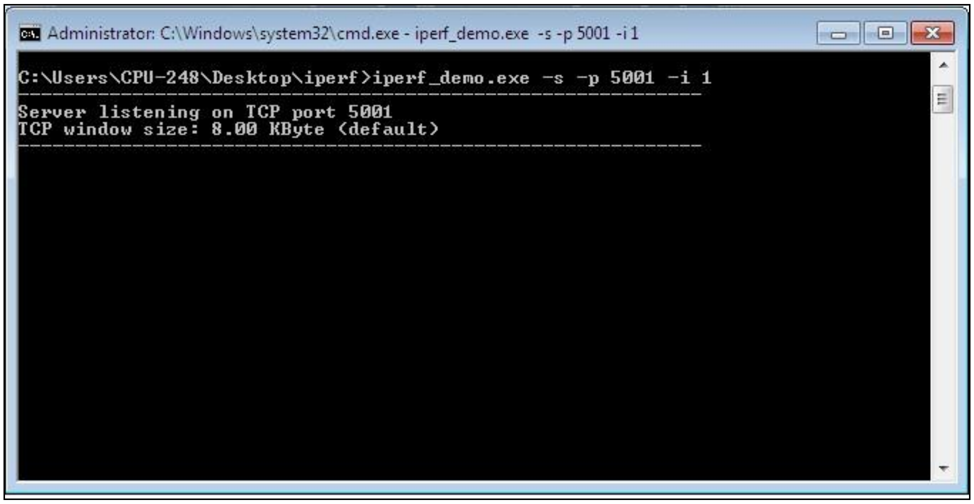
2. Open ***sapis/include/rsi_wlan_config.h*** file and update/modify following macros,

```
#define CONCURRENT_MODE                 RSI_DISABLE

#define RSI_FEATURE_BIT_MAP             FEAT_SECURITY_PSK

#define RSI_TCP_IP_BYPASS               RSI_DISABLE

#define RSI_TCP_IP_FEATURE_BIT_MAP TCP_IP_FEAT_DHCPV4_CLIENT

 #define  RSI_CUSTOM_FEATURE_BIT_MAP  0

 #define  RSI_BAND                       RSI_BAND_2P4GHZ
```

2.3 Executing the Application

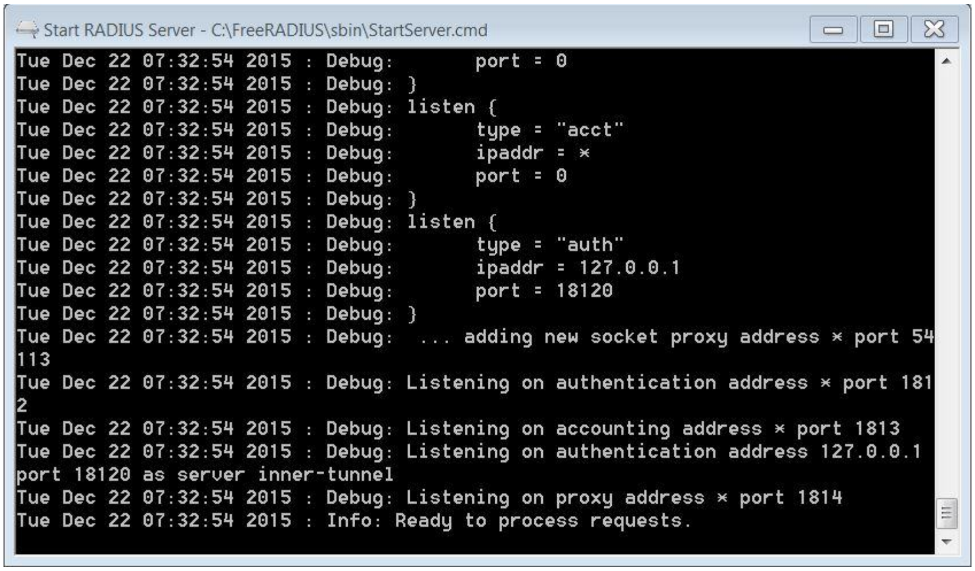
1. Configure the Access point in WPA-EAP/WPA2-EAP mode to connect WiSeConnect device in Enterprise secured mode.
2. Open TCP server application using iperf application in Windows PC2 which is connected to Access point through LAN.

```
Iperf_demo.exe -s -p <SERVER_PORT> -i 1
```


Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -p 5001 -i 1

```
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -p 5001 -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
```

3. Run Radius server in Windows/Linux PC2 which is connected to AP through LAN by providing required certificate and credintials.



```
Tue Dec 22 07:32:54 2015 : Debug:       port = 0
Tue Dec 22 07:32:54 2015 : Debug: }
Tue Dec 22 07:32:54 2015 : Debug: listen {
Tue Dec 22 07:32:54 2015 : Debug:       type = "acct"
Tue Dec 22 07:32:54 2015 : Debug:       ipaddr = *
Tue Dec 22 07:32:54 2015 : Debug:       port = 0
Tue Dec 22 07:32:54 2015 : Debug: }
Tue Dec 22 07:32:54 2015 : Debug: listen {
Tue Dec 22 07:32:54 2015 : Debug:       type = "auth"
Tue Dec 22 07:32:54 2015 : Debug:       ipaddr = 127.0.0.1
Tue Dec 22 07:32:54 2015 : Debug:       port = 18120
Tue Dec 22 07:32:54 2015 : Debug: }
Tue Dec 22 07:32:54 2015 : Debug:  ... adding new socket proxy address * port 54
113
Tue Dec 22 07:32:54 2015 : Debug: Listening on authentication address * port 181
2
Tue Dec 22 07:32:54 2015 : Debug: Listening on accounting address * port 1813
Tue Dec 22 07:32:54 2015 : Debug: Listening on authentication address 127.0.0.1
port 18120 as server inner-tunnel
Tue Dec 22 07:32:54 2015 : Debug: Listening on proxy address * port 1814
Tue Dec 22 07:32:54 2015 : Info: Ready to process requests.
```

4. SPI Interface

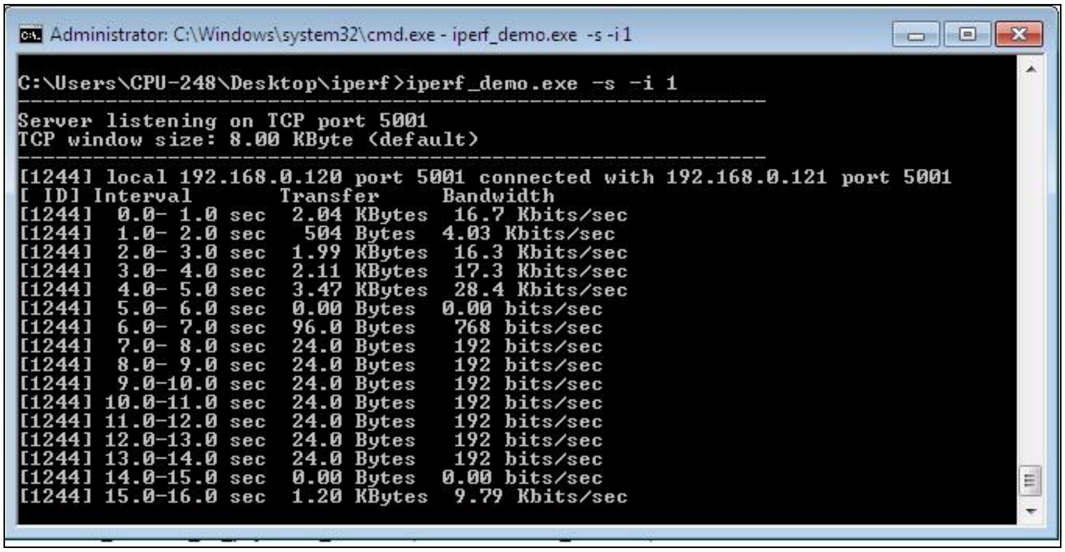
If User using SPI interface, Please refer the document ***sapis/platforms/spansion_MB9BF568NBGL/RS9113-WiSeConnect_SAPIS_Spansion_Project_User_guide.pdf*** for executing the ***eap*** example in CooCox IDE.

5. **UART/USB-CDC Interface**

If User using UART interface, Please refer the document ***sapis/platforms/ windows_uart/RS9113-WiSeConnect_SAPIS_Windows_Project_UserGuide.pdf*** for executing the ***eap*** example in Dev-C++ IDE

6. After the program gets executed, WiSeConnect Device would be connected to Access point which is in enterprise security having the configuration same that of in the application and get IP.

7. After successful connection, WiSeConenct STA connects to TCP server socket opened on Windows/Linux PC2 using TCP client socket and sends configured **NUMBER_OF_PACKETS** to remote TCP server. Please refer the given below image for reception of TCP data on TCP server.


Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1

```
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1244] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval       Transfer     Bandwidth
[1244]  0.0- 1.0 sec  2.04 KBytes  16.7 Kbits/sec
[1244]  1.0- 2.0 sec   504 Bytes  4.03 Kbits/sec
[1244]  2.0- 3.0 sec  1.99 KBytes  16.3 Kbits/sec
[1244]  3.0- 4.0 sec  2.11 KBytes  17.3 Kbits/sec
[1244]  4.0- 5.0 sec  3.47 KBytes  28.4 Kbits/sec
[1244]  5.0- 6.0 sec  0.00 Bytes  0.00 bits/sec
[1244]  6.0- 7.0 sec  96.0 Bytes   768 bits/sec
[1244]  7.0- 8.0 sec  24.0 Bytes   192 bits/sec
[1244]  8.0- 9.0 sec  24.0 Bytes   192 bits/sec
[1244]  9.0-10.0 sec  24.0 Bytes   192 bits/sec
[1244] 10.0-11.0 sec  24.0 Bytes   192 bits/sec
[1244] 11.0-12.0 sec  24.0 Bytes   192 bits/sec
[1244] 12.0-13.0 sec  24.0 Bytes   192 bits/sec
[1244] 13.0-14.0 sec  24.0 Bytes   192 bits/sec
[1244] 14.0-15.0 sec  0.00 Bytes  0.00 bits/sec
[1244] 15.0-16.0 sec  1.20 KBytes  9.79 Kbits/sec
```