DHCP Option-81 (FQDN) Application

User guide

Version 0.2

Oct 2016

Redpine Signals, Inc.
2107 N. First Street, #680
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 a WiFi station and sends DNS update requests to the desired DNS server.

Disclaimer:

Table of Contents

1 Introduction 4
1.1 DHCP Option-81 Overview 4
1.2 Application Overview 4
1.2.1 Overview 4
1.2.2 Sequence of Events 4
1.3 Application Setup 4
1.3.1 SPI based Setup Requirements 4
1.3.2 UART/USB-CDC based Setup Requirements 5
2 Configuration and Execution of the Application 6
2.1 Initializing the Application 6
2.1.1 SPI Interface 6
2.1.2 UART/USB-CDC Interface 6
2.2 Configuring the Application 6
2.3 Executing the Application 8

Table of Figures

No table of figures entries found.

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 DHCP Option-81 Overview

DHCP Option – 81 is is used to update the devices DNS hostname in the configured DNS server. Option-81 can be used to exchange information about a DHCPv4 client’s fully qualified domain name and about responsibility for updating the DNS RR related to the client’s address assignment .

DNS maintains (among other things) the information about the mapping between hosts' Fully Qualified Domain Names (FQDNs) [11] and IP addresses assigned to the hosts. The information is maintained in two types of Resource Records (RRs): A and PTR. The DNS update specification ([4]) describes a mechanism that enables DNS information to be updated over a network. The Dynamic Host Configuration Protocol for IPv4 (DHCPv4 or just DHCP in this document) [5] provides a mechanism by which a host (a DHCP client) can acquire certain configuration information, along with its address. This document specifies a DHCP option, the Client FQDN option, which can be used by DHCP clients and servers to exchange information about the client's fully qualified domain name for an address and who has the responsibility for updating the DNS with the associated A and PTR RRs.

1.2 Application Overview

1.2.1 Overview

The application demonstrates how to configure WiSeConnect device in client mode to send DNS update requests to the configured DNS server.

1.2.2 Sequence of Events

This Application explains user how to:

- Connect to Access Point in station mode
- Send DNS update request to the configured DNS server

1.3 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:

1.3.1 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
- Wireless Access Point

1.3.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- Wireless Access Point

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 ***dhcp_dns_fqdn*** 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 ***dhcp_dns_fqdn*** example in Dev-C++ IDE

2.2 Configuring the Application

1. Open sapis/examples/wlan/dhcp_dns_fqdn/***rsi_dhcp_dns_fqdn.c*** file and update/modify following macros,

SSID refers to the name of the Access point.

```
#define SSID                    "<ap name>"
```

CHANNEL_NO refers to the channel in which device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO          0
```

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

Valid configuration is:

RSI_OPEN - For OPEN security mode

RSI_WPA - For WPA security mode

RSI_WPA2 - For WPA2 security mode

```
#define SECURITY_TYPE          RSI_OPEN
```

PSK refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK                      "<psk>"
```

To configure IP address

DHCP_MODE refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE    1
```

> **Note:** If user wants to configure STA IP address through DHCP then set **`DHCP_MODE`** to 1 and skip configuring the following **`DEVICE_IP, GATEWAY`** and **`NETMASK`** macros.
>
> (Or)
>
> If 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 "192.168.10.10" as IP address, update the macro **`DEVICE_IP`** as **`0x0A0AA8C0`**.

```
        #define   DEVICE_IP            0X0A0AA8C0
```

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

Example: To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **`0x010AA8C0`**

```
        #define   GATEWAY              0x010AA8C0
```

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
```

Configure following macors to send DNS update to the configured DNS server

IP address of the DNS server.

Example: To configure "192.168.10.1" as **`RSI_DNS_SERVER_IP`**, update the macro **`RSI_DNS_SERVER_IP`** as **`0x0A0AA8C0`**.

```
        #define    RSI_DNS_SERVER_IP    0x6500A8C0
```

`RSI_DNS_TTL` refers the time to live value of the hostname.

```
        #define    RSI_DNS_TTL          53
```

`RSI_DNS_ZONE_NAME` refers zone name of the configured DNS server.

```
        #define    RSI_DNS_ZONE_NAME    "rps"
```

`RSI_DNS_HOST_NAME` refers host name of the configured DNS server.

```
        #define    RSI_DNS_HOST_NAME     "redpine"
```

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_OPEN

#define RSI_TCP_IP_BYPASS            RSI_DISABLE

#define RSI_TCP_IP_FEATURE_BIT_MAP  (TCP_IP_FEAT_DHCPV4_CLIENT
                                     | TCP_IP_FEAT_DNS_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 OPEN/WPA-PSK/WPA2-PSK mode to connect WiSeConnect device in STA mode.
2. 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 ***dhcp_dns_fqdn*** example in CooCox IDE.
3. 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 ***dhcp_dns_fqdn*** example in Dev-C++ IDE
4. After the program gets executed, WiSeConnect module configured as client and connects to AP and gets IP.
5. After successful connection with the Access Point, Device starts sending DNS update request to the given **`RSI_DNS_SERVER_IP`** with configured **`RSI_DNS_ZONE_NAME`** `and` **`RSI_DNS_HOST_NAME`** to update the hostname of the Device.
6. In rsi_dhcp_dns_fqdn.c file, **`rsi_dns_update`** API returns success status, which means that the DNS update packet is successfully sent in to the medium. When actual response comes from the DNS server, it is known from the status parameter of the callback function (**rsi_dns_response_handler**) registered in the rsi_dns_update API.