Realtek mDNS User Guide

This document provides guideline to use mDNS module in SDK.

Table of Contents

1 Introduction

This document provides a guide to use mDNS module provided in SDK. In this section, how to start mDNS responder and manage mDNS services are explained. An mDNS example is also provided.

2 Data Types

This sub-section lists the data types used by provided mDNS API.

2.1 DNSServiceRef

```
typedef void *DNSServiceRef;
```

This data type is used to pointer to an mDNS service resource.

2.2 TXTRecordRef

```
typedef struct _TXTRecordRef_t {
        char PrivateData[16];
} TXTRecordRef;
```

This structure is used to store the context of a service text record.

3 mDNS API

This sub-section lists the provided API for mDNS operations.

3.1 mDNSResponderInit

This function triggers to start mDNS responder to serve mDNS query.

Syntax

```
int mDNSResponderInit (
        void
);
```

Parameters

None.

Return Value

If the function succeeds, the return value is 0.

Remarks

None.

3.2 mDNSResponderDeinit

This function triggers to stop mDNS responder.

Syntax

```
void mDNSResponderDeinit (
        void
);
```

Parameters

None.

Return Value

None.

Remarks

When mDNS responder is deinitailized, all registered service will also be deregistered.

3.3 mDNSRegisterService

This function triggers to register a service to mDNS.

Syntax

```
DNSServiceRef mDNSRegisterService (
        char *name,
        char *service_type,
        char *domain,
        unsigned short port,
        TXTRecordRef *txtRecord
);
```

Parameters

name

Service name to register.

service_type

Service type to register.

domain

Service domain to register.

port

Service port information for the service to register.

txtRecord

Text record information for the service to register.

Return Value

Return a pointer to the registered service resource if success, otherwise return null.

Remarks

None.

3.4 mDNSDeregisterService

This function triggers to unregister a service from mDNS.

Syntax

```
void mDNSDeregisterService (
        DNSServiceRef serviceRef
);
```

Parameters

serviceRef

Pointer to the registered service resource.

Return Value

None.

Remarks

None.

3.5 mDNSUpdateService

This function triggers to update the text record in a service resource.

Syntax

```
void mDNSUpdateService (
        DNSServiceRef serviceRef,
        TXTRecordRef *txtRecord,
        unsigned int ttl
);
```

Parameters

serviceRef

Pointer to a service resource.

txtRecord

New text record to update.

ttl

Time-to-live of this text record. Set 0 to use default TTL setting.

Return Value

None.

Remarks

None.

4 Text Record API

mDNS services may own its text record. This sub-section lists the provided text record API.

4.1 TXTRecordCreate

This function triggers to create a text record.

Syntax

```
void TXTRecordCreate (
        TXTRecordRef *txtRecord,
        uint16_t bufferLen,
        void *buffer
);
```

Parameters

txtRecord

Pointer to text record resource.

bufferLen

Default buffer length.

buffer

Default buffer assigned to save text record data.

Return Value

None.

Remarks

Default buffer is not necessary.

4.2 TXTRecordSetValue

This function triggers to add a key-value pair to text record.

Syntax

```
int TXTRecordSetValue (
        TXTRecordRef *txtRecord,
        const char *key,
        uint8_t valueSize,
        const void *value
);
```

Parameters

txtRecord

Pointer to text record resource.

key

Character string of the key for a pair

valueSize

Size of binary data value.

value

Binary data value for a pair.

Return Value

None.

Remarks

When buffer in text record resource to save text record data is not enough, it will be allocated dynamically.

4.3 TXTRecordDeallocate

This function triggers to release a text record resource.

Syntax

```
void TXTRecordDeallocate (
        TXTRecordRef *txtRecord
);
```

Parameters

txtRecord

Pointer to text record resource.

Return Value

None.

Remarks

None.

5 Platform Mandatory Implementation

This sub-section lists the mandatory functions that developers need to implement. An example is provided in mDNSPlatform.c. The mDNSPlatformHtons and mDNSPlatformInetAddr functions are used for LWIP wrapping on htons and inet_addr functions between different versions.

5.1 mDNSPlatformCustomInit

This mandatory function is invoked for custom initialization when mDNS initialization

Syntax

```
void mDNSPlatformCustomInit (
        void
);
```

Parameters

None.

Return Value

None.

Remarks

None.

5.2 mDNSPlatformHostname

This mandatory function is invoked to get hostname when mDNS initialization

Syntax

```
char *mDNSPlatformHostname (
        void
);
```

Parameters

None.

Return Value

Device hostname in network.

Remarks

This hostname will be used in mDNS service host information.

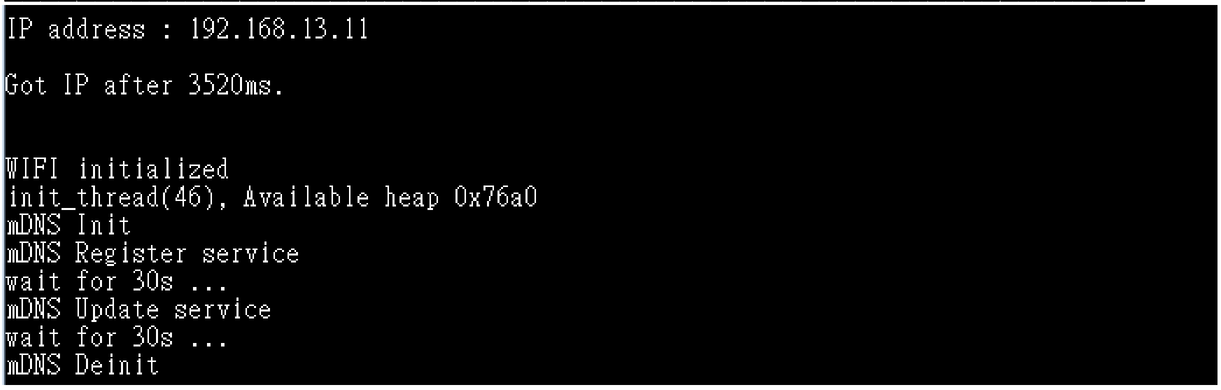
6 mDNS Example

An example to use the APIs explained in previous sections is provided in example_mdns.c. To execute this example automatically when booting, the CONFIG_EXAMPLE_MDNS in platform_opts.h must be enabled as the follows. LWIP IGMP should also be enabled to support multicast.

```
/* lwipopts.h */
#define LWIP_IGMP                   1

/* platform_opts.h *.
#define CONFIG_EXAMPLE_MDNS     1
```

In example, operations on mDNS responder and service are presented. If device Wi-Fi connection is setup by fast reconnect (Please refer to readme of fast reconnect example), the mDNS responder could be started after IP address is available. An example service is registered, updated. Finally, the mDNS responder is stopped.

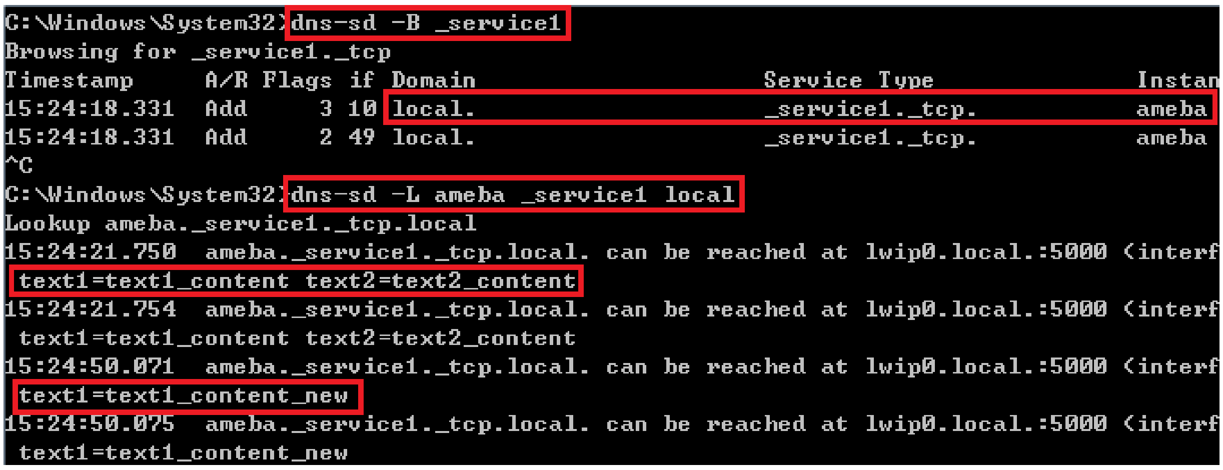


```
IP address : 192.168.13.11

Got IP after 3520ms.


WIFI initialized
init_thread(46), Available heap 0x76a0
mDNS Init
mDNS Register service
wait for 30s ...
mDNS Update service
wait for 30s ...
mDNS Deinit
```

To discover service in network, the “dns-sd” command provided in Bonjour SDK can be used. Bonjour SDK is available on Apple developer support site, and can be downloaded by Apple developer. In the following, “dns-sd -B” command is used to browse for service instance of type “_service1”, and service of “ameba._service1._tcp.local.” is found. “dns-sd -L” command is used to look up this service instance, and the content of service text record is shown. After service update operation, the original content of “text1=text1_cotent, text2=text2_content” is updated to “text1=text1_content_new”.



```
C:\Windows\System32>dns-sd -B _service1
Browsing for _service1._tcp
Timestamp     A/R Flags if Domain                    Service Type          Instan
15:24:18.331  Add      3 10 local.                   _service1._tcp.       ameba
15:24:18.331  Add      2 49 local.                   _service1._tcp.       ameba
^C
C:\Windows\System32>dns-sd -L ameba _service1 local
Lookup ameba._service1._tcp.local
15:24:21.750  ameba._service1._tcp.local. can be reached at lwip0.local.:5000 (interf
 text1=text1_content text2=text2_content
15:24:21.754  ameba._service1._tcp.local. can be reached at lwip0.local.:5000 (interf
 text1=text1_content text2=text2_content
15:24:50.071  ameba._service1._tcp.local. can be reached at lwip0.local.:5000 (interf
 text1=text1_content_new
15:24:50.075  ameba._service1._tcp.local. can be reached at lwip0.local.:5000 (interf
 text1=text1_content_new
```

7 mDNS Library Linking

The text and data of mDNS library in Ameba SDK is built in .mdns.text and .mdns.data sections. The linking of this library can be changed for requirement by modifying the image2.icf file. As the following example for image2.icf, adding .mdns.text section to .ram_image2.text block and .mdns.data section to .ram.data is for linking to SRAM. Adding .mdns.text and .mdns.data sections to SDRAM block is for linking to SDRAM. The default setting in image2.icf of Ameba SDK for mDNS libarary is linking to SDRAM.

```
/* Link mDNS to SRAM */
define block .ram_image2.text with fixed order{ section .infra.ram.start*,
                                    section .rodata*,
                                    block CPP_INIT,
                                    section .mon.ram.text*,
                                    section .hal.flash.text*,
                                    section .hal.gpio.text*,
                                    section .text*,
                                    section CODE,
                                    section .otg.rom.text,
                                    section Veneer object startup.o,
                                    section __DLIB_PERTHREAD,
                                    section .mdns.text
                                   };

define block .ram.data with fixed order{ section .data*,
                               section DATA,
                               section .ram.otg.data.a,
                               section .iar.init_table,
                               section .mdns.data
                              };
```

```
/* Link mDNS to SDRAM */
define block SDRAM with fixed order{ section .sdram.text*,
                            section .sdram.data*,
                            section .mdns.text*,
                            section .mdns.data*
                           };
```