This document describes using the mDNS APIs provided by the mDNS module in: freertos\_sdk\_x.y/components/mdns. This application demonstrates using these APIs to advertise and discover the services on the local network.

For more details regarding the features and limitations of mDNS implementation and APIs, refer mdns\_apiref.pdf.

# Code Walkthrough for mDNS

## Application Flow

mDNS is a protocol that provides DNS like facility to advertise and discover services in a local area network.

Following are the steps:

1. Connects to a Wi-fi network.
2. Registers to the hostname and services.
3. Acts as a mDNS announce or discovery according to the boot arguments passed.

## Sample Code Walkthrough

A user-defined data struct is created to store the data of the mDNS:

|  |
| --- |
| #define APP\_NAME "mDNS application"  #define APP\_VERSION "2.0"  OS\_APPINFO {.stack\_size = 4096};  #define INPUT\_PARAMETER\_SSID "ssid"  #define INPUT\_PARAMETER\_PASSPHRASE "passphrase"  #define INPUT\_PARAMETER\_HOST "host"  #define INPUT\_PARAMETER\_URL "url"  #define INPUT\_PARAMETER\_PATH "path"  #define INPUT\_PARAMETER\_PORT "port"  #define INPUT\_PARAMETER\_SECURED "secured"  #define INPUT\_PARAMETER\_METHOD "method"  #define INPUT\_PARAMETER\_CA "ca\_cert"  #define INPUT\_PARAMETER\_POST\_LEN "post\_len"  #define INPUT\_PARAMETER\_CLIENT\_CERT "client\_cert"  #define INPUT\_PARAMETER\_CLIENT\_KEY "client\_key"  #define INPUT\_PARAMETER\_TEST\_ITER "test\_iterations"  #define INPUT\_PARAMETER\_USE\_CA\_BUNDLE "use\_ca\_bundle"  #define NULL\_STR ""  struct param\_t {  const char \*ssid;  const char \*passphrase;  /\* parameters for advertise \*/  const char \*servicename;  const char \*service\_type;  const char \*proto; /\* 0 - UDP, 1- TCP \*/  const char \*port;  const char \*txt\_key;  const char \*txt\_val;  const char \*hostname;  /\* Parameters for Service Discovery \*/  const char \*sd\_srvc\_type;  const char \*sd\_srvc\_proto;  const char \*action;  };  bool wcm\_connected = false;  struct param\_t param;  char default\_port[8];  char keyVal[256];  const char \*txt\_key\_val[10]; |

The following boot arguments are passed in this application:

1. SSID and passphrase to connect to the Wi-Fi network.
2. Hostname, service name, service type, proto, port, text key and value.
3. SD service type and proto, and action.

|  |
| --- |
| static int  parse\_boot\_args(void)  {  int ret = 0;  param.ssid= os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_SSID);  param.passphrase= os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_PASSPH);  param.hostname= os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_HOST\_NAME);  param.servicename= os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_SRVC\_NAME);  param.service\_type= os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_SRVC\_TYPE);  param.proto= os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_PROTO);  param.port = os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_PORT);  param.txt\_key = os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_TXT\_KEY);  param.txt\_val = os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_TXT\_VAL);  param.sd\_srvc\_type = os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_SD\_SRVC\_TYPE);  param.sd\_srvc\_proto = os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_SD\_SRVC\_PROTO);  param.action = os\_get\_boot\_arg\_str(INPUT\_PARAMETER\_ACTION); |

To connect to a Wi-Fi network, wcm\_create()API from the Wi-Fi Connection Manager is used. Initially, the Wi-Fi network interface is created using wcm\_create().

|  |
| --- |
| h = wcm\_create(NULL); |

wifi\_connect\_to\_network()API, from components library, connects to the Wi-Fi network using the AP credentials provided.

|  |
| --- |
| rval = wifi\_connect\_to\_network(&h, WCM\_CONN\_WAIT\_INFINITE, &wcm\_connect\_success);  if(rval < 0) {  os\_printf("\nError: Unable to connect to network\n");  return 0;  } |

Here, the data structure mdns\_srvc\_info\_t is used to pass information about the discovered service such as service instance, protocol, type, port and IP address. Information is passed through the callback function provided while calling the mdns\_service\_discover API from main function.

|  |
| --- |
| void app\_dns\_sd\_cb(void \*data\_ptr,  const mdns\_srvc\_info\_t \*srvinfo,  uint32\_t status)  {  os\_printf("\n------------------------------------------------------------");  os\_printf("\n[APP]%s:Discovered service info:", \_\_FUNCTION\_\_);  os\_printf("\n\tInstance = %s \n\tservice type = %s \n\tprotocol = %s\n\t"  "txt info = %s \n\tport = %d \n\tipaddr = %x\n",  srvinfo->instance, srvinfo->srvc\_type, srvinfo->protocol,  srvinfo->txtInfo, srvinfo->port, srvinfo->ipaddr);  os\_printf("\n");  os\_printf("\n------------------------------------------------------------"); |

If the action parameter is set to 0, mDNS will register and advertise the service.

The data structure mdns\_hnreg\_param\_t is used to pass parameters while registering the Host Name. Set a unique host name for a device by using mac address. Add domain name as MDNS\_LOCAL\_DOMAIN and IP address in little endian format.

|  |
| --- |
| mdns\_hnreg\_param\_t hn\_reg\_prm = {0};  hn\_reg\_prm.hostname = (!param.hostname)?  (uint8\_t \*)"InnophaseDev" : (uint8\_t \*)param.hostname;  hn\_reg\_prm.domain = (uint8\_t \*)MDNS\_LOCAL\_DOMAIN;  hn\_reg\_prm.ipaddr = ntohl(ipaddr); |

The function mdns\_hostname\_reg is used for registering the host name of the interface.

|  |
| --- |
| mdns\_hostname\_reg(mc, &hn\_reg\_prm);  os\_printf("\n[APP]Host name Reg done"); |

The data structure mdns\_srvreg\_param\_t is used to pass the parameters while registering a service.

|  |
| --- |
| mdns\_srvreg\_param\_t srv\_reg = {0};  srv\_reg.srvc\_name = (uint8\_t \*)param.servicename;  srv\_reg.srvc\_type = (uint8\_t \*)param.service\_type;  srv\_reg.srvc\_proto = (!param.proto) ?  (uint8\_t \*)"\_tcp": (uint8\_t \*)param.proto;  srv\_reg.port = (!param.port)? 80 : atoi(param.port);  srv\_reg.domain = (uint8\_t \*)MDNS\_LOCAL\_DOMAIN; |

The function mdns\_service\_register is used for registering the service. Once registered, response for any matching query will be given internally.

|  |
| --- |
| mdns\_service\_register(mc, &srv\_reg);  os\_printf("\n[APP]Service Reg done"); |

The following section provides information on how the discovery of the service happens when the action parameter is not set to 0.

The data structure mdns\_srvcdisc\_param\_t is used for passing the parameters while discovering for service of specific type.

|  |
| --- |
| os\_printf("\n[APP]Discovering Service");  mdns\_srvcdisc\_param\_t sd\_param = {0};  sd\_param.srvc\_sub\_type = NULL;  sd\_param.srvc\_type = (!param.sd\_srvc\_type)?  (uint8\_t \*)"\_http": (uint8\_t \*)param.sd\_srvc\_type;  sd\_param.srvc\_proto =(!param.sd\_srvc\_proto)?  (uint8\_t \*)"\_tcp": (uint8\_t \*)param.sd\_srvc\_proto;  sd\_param.domain = (uint8\_t \*)MDNS\_LOCAL\_DOMAIN;  sd\_param.scope = MDNS\_SCOPE\_IPV4\_LOCAL;  sd\_param.cb = app\_dns\_sd\_cb;  mdns\_service\_discover(mc, &sd\_param); |

# Running the Application

## Programming Talaria TWO board with ELF

Program mdns.elf *(freertos\_sdk\_x.y/examples/mdns/bin)* using the Download tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down.
   2. ELF Input: Load the mdns.elf by clicking on Select ELF File.
   3. AP Options: Pass the SSID and Passphrase to connect to an Access Point.
   4. Boot Arguments: Pass the appropriate boot arguments.
   5. Programming: Prog RAM or Prog Flash as per requirement.

# Using the Application

Following are the list of boot arguments:

1. ssid: SSID of the Wi-Fi network to connect to.
2. passphrase: Passphrase of the network.
3. hostname: Hostname of the device. For example: “InnoDev”.
4. service\_name: Service name. For example: “Prov”.
5. service\_type: Type of service. For example: "\_http".
6. proto: Protocol type. For example: "\_tcp" / "\_udp".
7. port: Get/Post.
8. txt\_key: “key” part of the one key-val pair of txt data.
9. txt\_val: “val” part of the one key-val pair of txt data. Only one Key-val pair can be set.
10. sd\_srvc\_type: Type of service to discover.
11. sd\_srvc\_proto: Type of protocol to discover.
12. action:
    1. 0 – Register and Advertise a service.
    2. 1 – Discover a service. The combination of sd\_srvc\_type and sd\_srvc\_protoare used for discovering the service. Providing both the parameters is a must for discovering a service.

## Example Bootargs for mDNS Service Register and Announce

|  |
| --- |
| ssid=<ssid>,passphrase=<passphrase>,host=Innodev, service\_name =Prov, service\_type=\_http,proto=\_tcp,port=80,txt\_key=path,txt\_val=/data,action=0 |

Console output:

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PWWWWWWAE  Build $Id: git-e52d93e $  Flash detected. flash.hw.uuid: 39483937-3207-0080-0055-ffffffffffff  Bootargs: ssid=innotest\_AP passphrase=inno@1234 sd\_srvc\_type=\_http, sd\_srvc\_proto=\_tcp, action=1  $App:git-8b301e9  SDK Ver: FREERTOS\_SDK\_1.0  MDNS Demo App  Application Information:  ------------------------  Name : MDNS application  Version : 1.0  Build Date : Aug 24 2023  Build Time : 15:26:32  Heap Available: 311 KB (318968 Bytes)  [APP]Bootparams:  hostname=<null>  port = <null>  servicename = <null>  service\_type = <null>  proto = <null>  key = <null>  val = <null>  sd\_service\_type = \_http,  sd\_service\_proto = \_tcp,  [APP]Bootparams check done....ret = 0  addr e0:69:3a:00:08:38  network profile created for ssid: innotest\_AP  Connecting to added network : innotest\_AP  [1.330,481] DEAUTHENTICATED: reason 1  [1.740,583] CONNECT:0e:70:6c:d6:3a:62 Channel:6 rssi:-49 dBm  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_LINK\_UP  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_ADDRESS  [1.786,814] MYIP 192.168.99.195  [1.786,978] IPv6 [fe80::e269:3aff:fe00:838]-link  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_CONNECTED  Connected to added network : innotest\_AP  [APP]Starting Mdns  [APP]Mdns init done  [APP]Discovering Service |

Once mDNS Service Register and Announce is started, start any of the discovery apps and scan for the services that are announced by Talaria TWO. Following are some of the options for the user for the service discovery:

**Option 1:** Mobile application: mDNS Discovery

1. Install and launch the mDNS Discovery app on the mobile.

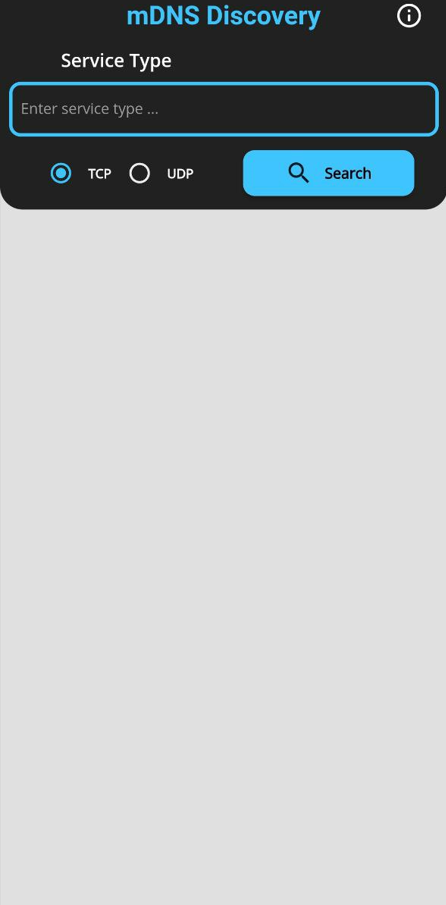


Figure : mDNS Discovery app

1. Add the service type i.e., HTTP in the search option, enable TCP which is the proto and click on Search.

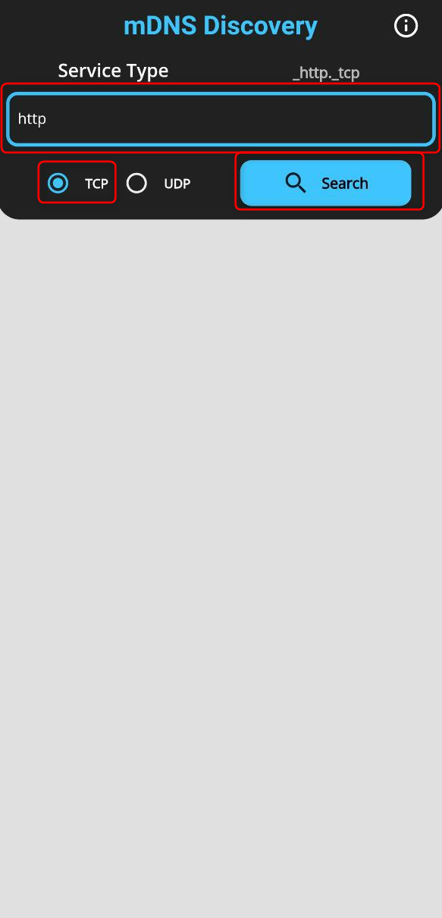


Figure : Add service type

1. Now the announced service from Talaria TWO can be found as shown in Figure 3.



Figure : Discovered service

**Option 2**: Command line on Windows OS

Prerequisite: Install the Bonjour Browser from the following link to run the command line on Windows OS: <https://hobbyistsoftware.com/bonjourbrowser>.

1. Service discovery can be done from a Windows command line, using the dns-sd command to browse for services that are being broadcast on the local network by Talaria TWO.

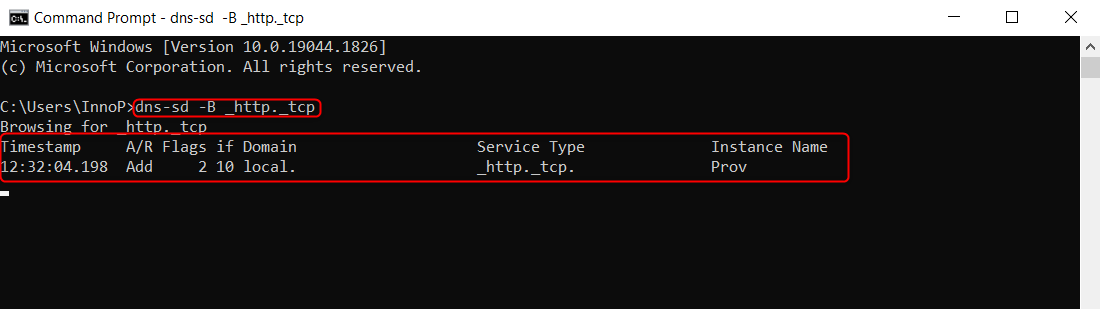


Figure : Service discovery from Windows command line

**Option 3**: User can also use two Talaria TWO modules: One for service register and announce from section 6.1 and one more for service discovery from section 6.2 to load the application. On the service discovery console, the announced service can be observed.

## Example Bootargs for mDNS Service Discovery

|  |
| --- |
| ssid=<ssid>,passphrase=<passphrase>, sd\_srvc\_type=\_http, sd\_srvc\_proto=\_tcp, action=1 |

Console output:

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PWWWWWWAE  Build $Id: git-e52d93e $  Flash detected. flash.hw.uuid: 39483937-3207-0080-0055-ffffffffffff  Bootargs: ssid=innotest\_AP passphrase=inno@1234 host=Innodev service\_name=Prov service\_type=\_http proto=\_tcp port=80 txt\_key=path txt\_val=/data action=0  $App:git-8b301e9  SDK Ver: FREERTOS\_SDK\_1.0  MDNS Demo App  Application Information:  ------------------------  Name : MDNS application  Version : 1.0  Build Date : Aug 24 2023  Build Time : 15:26:32  Heap Available: 311 KB (318968 Bytes)  [APP]Bootparams:  hostname=<null>  port = 80  servicename = Prov  service\_type = \_http  proto = \_tcp  key = path  val = /data  sd\_service\_type = <null>  sd\_service\_proto = <null>  [APP]Bootparams check done....ret = 0  addr e0:69:3a:00:08:38  network profile created for ssid: innotest\_AP  Connecting to added network : innotest\_AP  [0.746,137] CONNECT:0e:70:6c:d6:3a:62 Channel:6 rssi:-34 dBm  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_LINK\_UP  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_ADDRESS  [0.823,237] MYIP 192.168.99.195  [0.823,400] IPv6 [fe80::e269:3aff:fe00:838]-link  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_CONNECTED  Connected to added network : innotest\_AP  [APP]Starting Mdns  [APP]Mdns init done  [APP]Advertising Service  [APP]WCM interface ip addr = c363a8c0  [APP]Host name Reg done  [APP]Service Reg done  [APP]Hostname and Service Announce done  [APP]Service is Now discoverable by other devices in the N/w |