# Configure DHCP options

## Introduction

NET+OS V6.0 and V6.1 provide an ACE (Address Configuration Executive) module to configure the NET+ARM IP address. The ACE supports 6 address resolution protocols: Static ip address, DHCP, BOOTP, AUTOIP, RARP, and PING-ARP. When using DHCP protocol for getting address configuration information through the network, the user can chose what DHCP options he/she will use in the request packets or will get from the server.

This application note simply describes the DHCP protocols, packet format, options, and how to set up these DHCP options in NETOS. Another application note will be available shortly to address how to get DHCP options returned from server. In this application note, we assume the users already know how to customize and use ACE, therefore, we do not address any other issues in ACE than those related to setting up DHCP options. If you want to know ACE in detail, please refer to another application note, "The customization and usage of ACE".

## DHCP protocol and options

The Dynamic Host Configuration Protocol (DHCP) provides configuration parameters to Internet hosts in a client-server model using UDP/IP. The configuration parameters include IP address, subnet mask, next hop router, DNS server address, NTP server address, and many others. All parameters allocated by DHCP are actually leased for a finite period of time. The DHCP client sends a request packet with its required parameter list and client related information to DHCP server, then the server sends back a response that may or may not contain those requested parameters based on server's decision. For the detailed information of how DHCP works, the reader may refer to RFC 2131: Dynamic Host Configuration Protocol and 2132: DHCP Options and BOOTP Vendor Extensions.

DHCP Packet Format

0 4byte

| Operation | Hardware type | Hardware address length | Hops |
|---|---|---|---|
| Transaction ID (xid) | | | |
| Seconds | | Flags | |
| Client IP address (ciaddr) | | | |

| Assigned IP address (yidaddr) |
|---|
| Next server IP address (siaddr) |
| Relay agent IP address (giaddr) |
| Client hardware address (chaddr) |
| Next server name (sname) |
| File name |
| Options (variable length) |

Operation Message type: request (1) or reply (2)
Hardware type Client hardware address type
Hardware address length Length of the client hardware address, 6 for Ethernet.
Hops Number of relay agents that have forwarded the packet.
Transaction ID Clients use it to match requests to replies.
Seconds Time in seconds since the client began the DHCP process
Flags The Leftmost bit of the flags indicates that client requires broadcast responses, all bits must be zero.
Client IP address Used by the client to specify the IP address that it has been assigned (if it has an address).
Assigned IP address The server sets up this field to specify the IP address it assigns to the client
Next server IP address The server sets up this field to specify a configuration server (if there is one)
Relay agent IP address IP address of the relay agent (if applicable)
Client hardware address Client MAC address (typically left blank)
Next server name Name of the configuration server (if applicable, typically left blank)
File name Name of the file to get from the configuration server (if applicable, typically left blank)

Options

Option format

| Option Code | Length | Value |
|---|---|---|

BOOTP cookie

| 0x63825363 |
|---|

All options begin with an one byte option code, which uniquely identifies the option. Followed the option code is the one byte length field containing the length of the option value. Options may be fixed length or variable length. Fixed-length options without data consist of only an option code. Only options 0 and 255 are fixed length. All other options are variable-length. The value of the length does not include the two bytes specifying the option code and length. In the case of some variable-length options the length field is a constant but must still be specified. The length byte is followed by "length" bytes of data, the value field. All multi-byte quantities are in network byte-order.

The options section always begins with a four-byte “BOOTP cookie”. The users don’t need to be concerned about the “BOOTP cookie”, because NETOS DHCP client engine automatically sets it up. After the “BOOTP cookie”, there are one or more DHCP options. The readers can refer to RFCs 1533 and 2132: DHCP Options and BOOTP Vendor Extensions for their usage.

Set up DHCP options with NETOS

NETOS DHCP client in the ACE module supports all DHCP options defined in RFC 1533. NETOS defines all the constants in netos\h\tcpip\dhcpOptions.h file for the option codes for the corresponding DHCP options. The following two DHCP options are set up by NETOS DHCP client automatically.

| Options in RFC | constants in dhcpOptions.h | option code |
|---|---|---|
| **DHCP Message Type** | DHCP_MESSAGE_TYPE | 53 |
| **Maximum DHCP Message Size** | DHCP_MAX_MSG_SIZE | 57 |

The user can set up some DHCP options through the DHCP protocol data structure “configAceDhcpInfo” in

static aceConfigInterfaceInfo NADefaultEthInterfaceConfig[]

in netos\src\bsp\platforms\××××\aceParams.c [1)] or set up all other options through “proto_specific_info” in aceProtocolInfo structure in function customizeStartAce() in netos\src\bsp\platforms\××××\aceCallbacks.c[1)].

The following two DHCP options can be set up through “configAceDhcpInfo” . If the users are only interested in these two options, this is the best way to do it, because what you need to do is only to change the values of the “suggested_ip_address” and “suggested_lease_time” elements in the “configAceDhcpInfo” structure in

static aceConfigInterfaceInfo NADefaultEthInterfaceConfig[]

in bsp\platforms\××××\aceParams.c[1)] as shown in the following example. Users can refer to netos\h\tcpip\ace_params.h for the definitions of structures "configAceDhcpInfo" and "aceConfigInterfaceInfo".

| Options in RFC | constants in dhcpOptions.h | option code |
|---|---|---|
| **Requested IP Address** | DHCP_REQUESTED_IP_ADDR | 50 |
| **IP Address Lease Time** | DHCP_IP_ADDR_LEASE_TIME | 51 |

The default setting of "configAceDhcpInfo" in NADefaultEthInterfaceConfig[] structure in netos\src\bsp\platforms\××××\aceParams.c [1)] is:

```
{                       /* config for DHCP*/
        CONFIG_IS_VALID,
        IS_ENABLED,
        ACE_UNKNOWN_IP_ADDRESS,       /* can't suggest an IP address */
        ACE_UNKNOWN_IP_ADDRESS,       /* can't suggest a server*/
        ACE_UNKNOWN_IP_ADDRESS,       /* can't suggest a gateway*/
        ACE_INFINITE,                 /* get as long a lease as possible */
        ACE_DHCP_RETRY_FOREVER,
        ACE_DONT_KNOW_START_TIME,
        ACE_RESTART_DHCP_DISCOVER,
        TRUE,                         /* need broadcast response */
        TRUE,                         /* do initial random delay */
        0,                          /* use default ARP delay timeout */
        { 0},                         /* will copy dhcp_desired_params here*/
        asizeof(dhcp_desired_params),   /* number of DHCP parameters*/
        {ACE_PROT_DHCP, 0, DEFAULT_DHCP_DELAY,
ACE_CONT_IF_GOT_ADDRESS}
      }
```

The Requested IP Address option can be used by the DHCP client to request a specific IP address. The IP Address Lease Time allows a DHCP client to request the time period during which it will "own" the offered IP address. The option of Requested IP Address is set to 0 (ACE_UNKNOWN_IP_ADDRESS ) by default and the IP Address Lease Time set to the maximum integer value (ACE_INFINITE, in seconds) by default. The user can modify the structure shown above to set up the options of Requested IP Address and IP Address Lease Time. For example, setting up the suggested IP address as "198.192.48.33" and lease time as 5 hours, you can modify the "configAceDhcpInfo" structure as below.

```
{                       /* config for DHCP*/
```

```
        CONFIG_IS_VALID,
        IS_ENABLED,
        0xc6c03021,        /* suggest an IP address as “198.192.48.33”*/
        ACE_UNKNOWN_IP_ADDRESS,        /* can't suggest a server*/
        ACE_UNKNOWN_IP_ADDRESS,        /* can't suggest a gateway*/
        18000,              /* set lease time as 5 hours */
        ACE_DHCP_RETRY_FOREVER,
        ACE_DONT_KNOW_START_TIME,
        ACE_RESTART_DHCP_DISCOVER,
        TRUE,                    /* need broadcast response */
        TRUE,                    /* do initial random delay */
        0,                     /* use default ARP deply timeout */
        { 0},                    /* will copy dhcp_desired_params here*/
        asizeof(dhcp_desired_params),   /* number of DHCP parameters*/
        {ACE_PROT_DHCP, 0, DEFAULT_DHCP_DELAY,
ACE_CONT_IF_GOT_ADDRESS}
    }
```

After this modification, the user needs to rebuild the BSP library and then the application. Thus, the DHCP client engine will include these two options in the DHCP request packets.

All other DHCP options can be set up through the pointer element “proto_specific_info” and the size element “proto_specific_info_size” in aceProtocolInfo structure in function customizeStartAce() in bsp\platforms\××××\aceCallbacks.c[1]. The options of Requested IP Address and IP Address lease time can also be set up through “proto_specific_info”.
If the user wants to set up the options of Requested IP Address and IP Address Lease Time through “proto_specific_info”, please make sure only set up here and leave the “configAceDhcpInfo” structure as default. The NETOS DHCP client engine uses the void pointer “proto_specific_info” and the size “proto_specific_info_size” to extract the DHCP options and do a memory copy when building a DHCP request packet. Please refer to netos\h\tcpip\ace.h for the definition of structure “aceProtocolInfo”.

The default code for setting up DHCP protocol information structure in function customizeStartAce() in bsp\platforms\××××\aceCallbacks.c[1] is as below.

```
if ((nif->dhcp_config.isConfigValid) && (nif->dhcp_config.isEnabled))
        {
            memcpy(&protoInfo[ifCount][protoCount].proto_start_info, &nif-
>dhcp_config.startInfo, sizeof protoInfo[ifCount][protoCount].proto_start_info);
            protoInfo[ifCount][protoCount].proto_specific_info = &nif->dhcp_config;
            protoInfo[ifCount][protoCount].proto_specific_info_size = sizeof nif-
>dhcp_config;
```

```
            protoInfo[ifCount][protoCount].proto_event_info_size    =
ace_dhcp_event_info_size;
            protoInfo[ifCount][protoCount].start_protocol_fn = ace_start_dhcp;
            protoInfo[ifCount][protoCount].stop_protocol_fn = ace_stop_dhcp;
            protoInfo[ifCount][protoCount].get_addr_fn = ace_get_dhcp_addr_info;
            protoInfo[ifCount][protoCount].release_addr_fn = ace_release_dhcp_addr;
            protoCount++;
        }
```

The actual codes for setting up DHCP options are

```
protoInfo[ifCount][protoCount].proto_specific_info = &nif->dhcp_config;
protoInfo[ifCount][protoCount].proto_specific_info_size = sizeof nif->dhcp_config;
```

Therefore, the user needs to modify these two lines to set up the DHCP options which he/she wants the DHCP client engine to include in the DHCP request packets. The DHCP client engine assumes that the pointer "proto_specific_info" points to a memory area which consists of the "configAceDhcpInfo" structure data immediately followed by the user's configured DHCP options and the size "proto_specific_info_size" is the total length of the "configAceDhcpInfo" structure and all the configured DHCP options.

The default two line codes only assign the address and size of the "configAceDhcpInfo" structure data defined in NADefaultEthInterfaceConfig[] in bsp\platforms\××××\aceParams.c[1)] to the pointer "proto_specific_info" and the size "proto_specific_info_size" elements in "aceProtocolInfo" structure, respectively. Therefore, there are no DHCP options set up at default. The user can not use the default codes ( nif->dhcp_config) to append the DHCP options at the end of the "configAceDhcpInfo" structure data defined in NADefaultEthInterfaceConfig[] in bsp\platforms\××××\aceParams.c[1)] with memory copy routine, because it will overwrite other protocol information data immediately following, (please see the NADefaultEthInterfaceConfig[] in bsp\platforms\××××\aceParams.c[1)]). The user must implement a separate buffer big enough to hold all the "configAceDhcpInfo" structure data defined in NADefaultEthInterfaceConfig[] and the configured DHCP options. The smallest size buffer must be larger than the size (324 bytes) of the structure of the "configAceDhcpInfo". The added DHCP options data must follow the DHCP option format defined in RFC and also simply addressed above.

Here is an example of adding **Host Name Option** (option code 12) and **Vendor class identifier** (option code 60). But the user can add as many as he/she like in the same way.

```
1. char DHCPinfo[350];
2. char hostname[]="Netsilicon";
3. char classidentifier[]="abcd";
4. u8 hostnameopcode=12,  classidentifieropcode = 60;
```

```
5.  u8 hostnamelength=sizeof hostname;
6.  u8 classidentifierlength=sizeof  classidentifier;
7.  int length=0;
8.
9.  length= sizeof (nif->dhcp_config);
10. memcpy(DHCPinfo,&nif->dhcp_config, sizeof (nif->dhcp_config));
11. memcpy(DHCPinfo+length, &hostnameopcode,1);
12. length+=1;
13. memcpy(DHCPinfo+ length ,&hostnamelength, 1);
14. length+=1;
15. memcpy(DHCPinfo+ length,&hostname, hostnamelength);
16. length+= hostnamelength;
17.
18. memcpy(DHCPinfo+length, & classidentifieropcode,1);
19. length+=1;
20. memcpy(DHCPinfo+ length ,& classidentifierlength, 1);
21. length+=1;
22. memcpy(DHCPinfo+ length,&hostname, classidentifierlength);
23. length+= classidentifierlength;
24.
25. protoInfo[ifCount][protoCount].proto_specific_info = DHCPinfo;
26. protoInfo[ifCount][protoCount].proto_specific_info_size = length;
```

Line 1 declares a 350 byte buffer DHCPinfo which will hold 324 bytes "configAceDhcpInfo" structure data and 18 bytes configured DHCP options Host Name Option and Vendor class identifer.
Lines 2 –6 declare variables (option codes, lengths, values) related to the selected DHCP options.
Line 7 declares an integer variable used to accumulate the total length of the "configAceDhcpInfo" structure data and the DHCP options.
Line 10 copies the default "configAceDhcpInfo" structure data defined in NADefaultEthInterfaceConfig[] into buffer DHCPinfo memory.
Lines 11 and 18 append the DHCP option code fields.
Lines 13 and 20 append the length fields of the DHCP option values.
Lines 15 and 22 append the fields of the DHCP option values.
Lines 9, 12, 14, 16, 19, 21, and 23 accumulate the total length of the "configAceDhcpInfo" structure data and the DHCP options.
Line 25 assigns the address of the buffer DHCPinfo which contains the "configAceDhcpInfo" structure data defined in NADefaultEthInterfaceConfig[] and the configured DHCP options to the pointer "proto_specific_info" element in the "aceProtocolInfo" structure.
Line 26 assigns the size of the total length of the "configAceDhcpInfo" structure data and the DHCP options to the size "proto_specific_info_size" element in the "aceProtocolInfo" structure.

With this example code, the DHCP client engine extracts the information of the "configAceDhcpInfo" structure and DHPC options of **Host Name Option** (option code 12) and **Vendor class identifier** (option code 60), and builds DHCP request packets with **Host Name Option** (option code 12) and **Vendor class identifier** (option code 60).

After modifying the function customizeStartAce(), please rebuild BSP library and then your application.

NETOS also allows the users to specify the DHCP options that the DHCP client will request from the server. There is an unsigned byte array called "dhcp_desired_params" defined in netos\src\bsp\platforms\××××\aceParams.c [1)] to list the desired request from the server. The constants in the list are defined in netos\h\tcpip\dhcpOptions.h file. The user can update this list to include any additional DHCP options required by your application. After modifying this list, please rebuild BSP library and then your application.

1) ×××× ------means target, such as, ns7520_a, net50_d, ns9750_a, ……