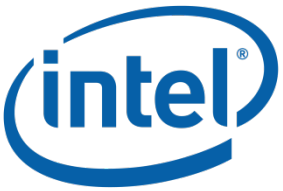

Getting Started Guide of EDKII HTTP Boot

April 2016

Revision 0.8

WHITEPAPER

Ye Ting, Fu Siyuan, Zhang Lubo

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Intel® UEFI Development Kit Debugger Tool may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel®, Intel® UEFI Development Kit Debugger Tool, Intel® UDK Debugger Tool, and the Intel® logo are trademarks or registered trademarks of Intel® Corporation or its subsidiaries in the United States and other countries.

Other names and brands may be claimed as the property of others.

Copyright© 2015 – 2-16 Intel Corporation. All rights reserved.

Contents

1 Feature Scope 1
- 1.1 Related Protocols 1
- 1.2 Feature Scope 1

2 Quick Start Guide 3
- 2.1 Network topology 3
- 2.2 Configure DHCPv4 server 3
- 2.3 Configure DHCPv6 server 4
- 2.4 Configure DNSv4 server 5
- 2.5 Configure DNSv6 server 5
- 2.6 Configure HTTP server 6
- 2.7 Build NT32 Simulator 7
- 2.8 Run HTTP boot 7

3 Enable HTTP boot for your system 10
- 3.1 Enable HTTP Boot over IPv4 stack on a platform has EDKII network 10
- 3.2 Enable HTTP Boot over IPv4 stack on a platform without EDKII network 10
- 3.3 Enable HTTP Boot over IPv6 stack on a platform has EDKII network 11
- 3.4 Enable HTTP Boot over IPv6 stack on a platform without EDKII network 12

Figures

Figure 1: HTTP boot test-bed 3
Figure 2: Configure DNSv4 Service 5
Figure 3: Configure DNSv6 Service 6
Figure 4: Configure HTTP Service - Tomcat 7
Figure 5: Select “UEFI Http” to boot over IPv4 stack 8
Figure 6: Select “UEFI Http2” to boot over IPv6 stack 8
Figure 7: Boot the downloaded UEFI Shell 9

Revision History

Revision Number	Description	Revision Date
<0.5>	Initial release.	August 2015
<0.6>	Include HTTP Utilities protocol and driver.	August 2015
<0.7>	Add Http Boot over IPv6 stack	December 2015
<0.8>	Corrections to code in section 2.3	April 2016

§

Feature Scope

1 Feature Scope

This document is a getting started guide for using the HTTP boot capability introduced in UEFI Specification, revision 2.5.

1.1 Related Protocols

UEFI 2.5 specification introduces a serial of new protocols which are related to HTTP boot in EDKII network stack:

- HTTP Service Binding Protocol – used to locate communication devices that are supported by a HTTP driver and to create and destroy instances of the HTTP child protocol.
- HTTP Protocol – used to provide HTTP service to create and transmit HTTP requests, as well as handle HTTP responses that are returned by a remote host.
- HTTP Utilities Protocol – used to build a raw HTTP message from a list of HTTP headers or to parse HTTP headers from a raw HTTP message.
- DNS4 & DNS6 Service Binding Protocol– used to locate communication devices that are supported by a DNS driver and to create and destroy instances of the DNS child protocol.
- DNS4 & DNS6 Protocol – used to get host name and address mapping from DNS server.
- IPv4 Configuration II Protocol – a new protocol which replaces IPv4 Configuration protocol, used to provide the mechanism to set and get various types of configuration for the EFI IPv4 network stack, including DNS server list.

Also, the *UEFI Specification*, revision 2.5, defines the procedure of 'HTTP boot' as one example of 'boot from URI'. The procedure uses the DHCP options to identify the name and path of the NBP (Network Boot Program), which are specified as a URI string in several formats.

1.2 Feature Scope

Current implementation adds the following new modules to the EDKII network stack:

- HTTP Boot Driver

NetworkPkg\HttpBootDxe\HttpBootDxe.inf

- HTTP Driver

NetworkPkg\HttpDxe\HttpDxe.inf

- HTTP Utilities Driver

NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf

- DNS Driver

NetworkPkg\DnsDxe\DnsDxe.inf

- HTTP Library

MdeModulePkg\Library\DxeHttpLib\DxeHttpLib.inf

Per the *UEFI Specification*, revision 2.5, the implementation can enable HTTP boot over either an IPv4 stack or an IPv6 stack, or both. The current implementation supports HTTP boot over both an IPv4 network stack and anIPv6 network stack.. The supported HTTP version is 1.1, currently in common use as of the publication of this document.

The current implementation removes the following module from an EDKII network stack, as it was used to produce the IPv4 configuration protocol (deprecated by the UEFI 2.5 Specification).

- IP4Config Driver

MdeModulePkg\Universal\Network\Ip4ConfigDxe\Ip4ConfigDxe.inf

The current implementation updates the following module in an EDKII network stack. The updated IP4 driver produces an IPv4 configuration II protocol; also it supplies the HII configuration pages to the end user.

- IP4 Driver

MdeModulePkg\Universal\Network\Ip4Dxe\Ip4Dxe.inf

2 Quick Start Guide

This document assumes that readers have installed EDKII and can build and run the NT32 simulator. This guide gives detailed instruction to set up the HTTP boot environment over an IPv4 and an IPv6 network stack.

2.1 Network topology

Figure 1 shows a test-bed using HTTP boot. In this example, the DHCP server and DNS server are separately deployed on Ubuntu 15.10 and Windows Server 2012 R2. These two servers and an NT32 simulator are located on the same subnet: IPv4 (192.168.10.0) / IPv6 (fec0:0:0:10::/64).

In this example, the HTTP server is deployed in another Windows Server 2012 R2 and located on a different subnet: IPv4 (192.168.20.0) / IPv6 (fec0:0:0:20::/64). The two subnets are connected by a gateway.

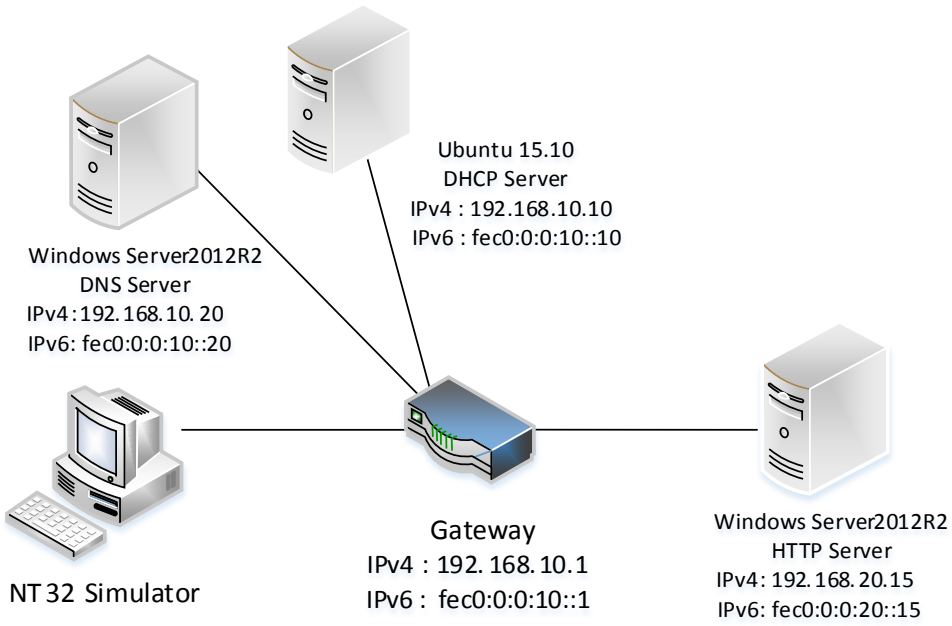

Figure 1: HTTP boot test-bed

Note: *You may use an alternative solution to establish your test-bed and configure your DHCP server, DNS server and HTTP Server. This chapter describes one such approach as a reference or guide.*

2.2 Configure DHCPv4 server

The steps to configure DHCPv4 server are as follows:

1. Install DHCPv4 server: sudo apt-get install isc-dhcp-server.

2. Edit /etc/dhcp/dhcpd.conf file as shown below.

```
default-lease-time 600;
max-lease-time 7200;
ddns-update-style none;
log-facility local7;
#option definitions common to all supported networks...
option domain-name “cloudboot.com”;
option domain-name-servers 192.168.10.20;
option routers 192.168.10.1;
option vendor-class-identifier “HTTPClient”;
option bootfile-name “http://www.cloudboot.com:8080/EFI/Shell.efi”;
#This declaration allows BOOTP clients to get dynamic address.
subnet 192.168.10.0 netmask 255.255.255.0 {
    range 192.168.10.100 192.168.10.250;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.10.255;
}
```

3. Be sure that your server will listen for DHCP requests on the correct interface (Note: change the INTERFACE to match your own). Edit the /etc/default/isc-dhcp-server file:

   INTERFACE = “eth0”;

4. Restart the DHCPv4 service: sudo service isc-dhcp-server restart.

2.3 Configure DHCPv6 server

The steps to configure DHCPv6 server are as follows:

1. Install DHCPv6 server: sudo apt-get install isc-dhcp-server.

2. If there is no dhcpd6.conf file in /etc/dhcp/ directory, create and edit as shown below.

```
default-lease-time 600;
max-lease-time 7200;
log-facility local7;
#option definitions common to all supported networks...
option dhcp6.vendor-class code 16 = {integer 32, integer 16, string};
subnet6 fec0:0:0:10::/64 {
    #Range for clients
```

```
    range6 fec0:0:0:10::100 feco:0:0:0:10:0:0:ffff:ffff;
    option dhcp6.domain-search “cloudbootip6.com”;
    option dhcp6.name-servers fec0:0:0:10::20;
    option dhcp6.vendor-class 0 0  “HTTPClient”;
    option dhcp6.bootfile-url “http://www.cloudbootip6.com:8080/EFI/Shell.efi”;
}
```

3. Be sure that your server will listen for DHCP requests on the correct interface (Note: change the INTERFACE to match your own). Edit the /etc/default/isc-dhcp-server file:

   INTERFACE = “eth0”;

4. Restart the DHCPv6 service: sudo service isc-dhcp-server6 restart.

2.4 Configure DNSv4 server

The steps to configure DNSv4 server are shown as following:

1. Add DNS service in windows server manager;
2. Add a new forward lookup zone “cloudboot.com” and add new Host “www” to 192.168.20.15.

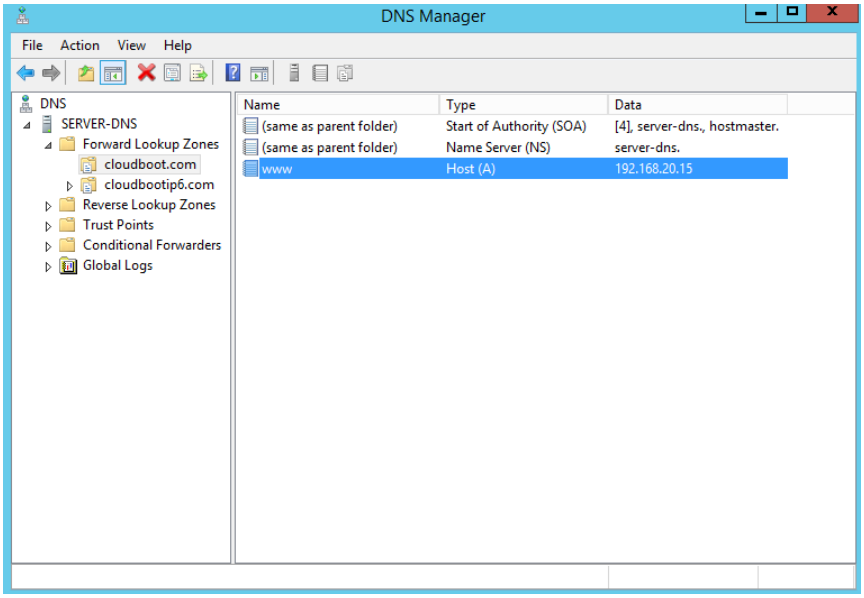

Figure 2: Configure DNSv4 Service

2.5 Configure DNSv6 server

The steps to configure DNSv6 server are shown as follows:

1. Add DNS service in windows server manager;

2. Add a new forward lookup zone “cloudbootip6.com” and add new Host “www” to fec0:0:0:20::15.

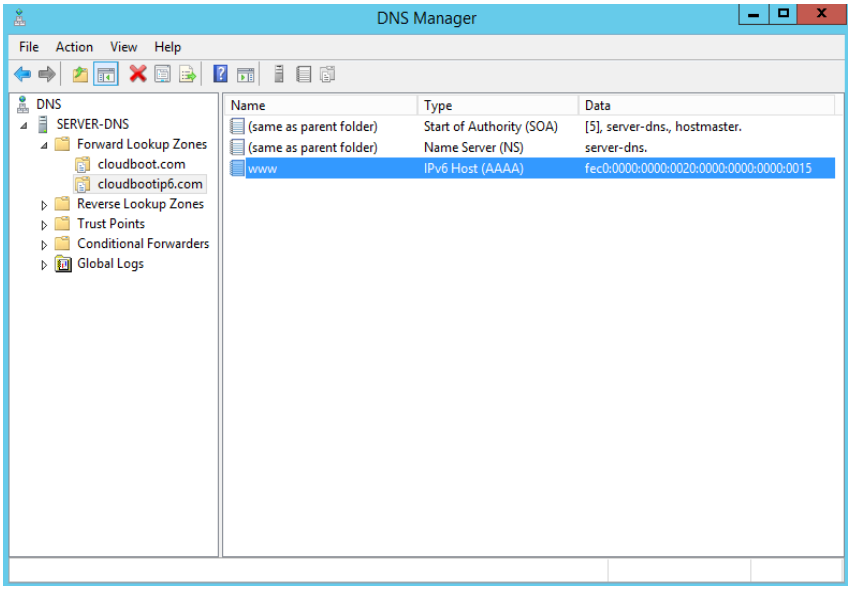

Figure 3: Configure DNSv6 Service

2.6 Configure HTTP server

The steps to configure HTTP server follow:

1. Download Tomcat 7.0 and install;
2. Start Tomcat service;
3. Create a new folder “EFI” under the root folder of Tomcat webapp:.
   - EXAMPLE: C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0\webapps\ROOT\EFI
4. Copy a shell application into the EFI folder and change the name to “Shell.efi” (same as the configuration in dhcpd.cong or dhcpd6.conf)

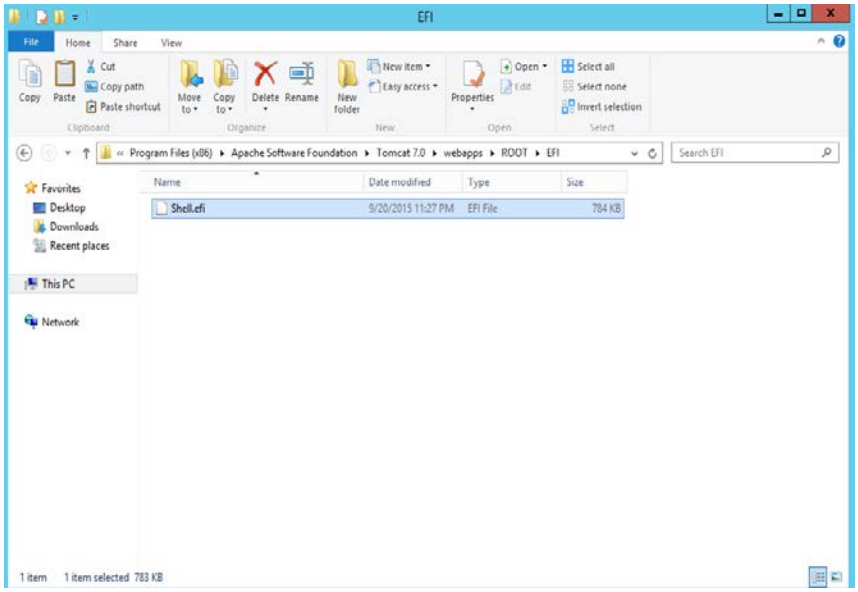

Figure 4: Configure HTTP Service - Tomcat

2.7 Build NT32 Simulator

Use the following command to build NT32 simulator:

```
build -a IA32 -t VS2013x86 -p Nt32pkg\Nt32Pkg.dsc
```

The pictures shown below are captured from NT32 simulator.

2.8 Run HTTP boot

Now, you are ready to run HTTP boot over NT32 simulator. Start NT32 simulator and enter Boot Manager, then select “UEFI Http” to boot over IPv4 stack (refer to Figure 5) or “UEFI Http 2” to boot over IPv6 stack (refer to Figure 6).

Figure 5: Select "UEFI Http" to boot over IPv4 stack

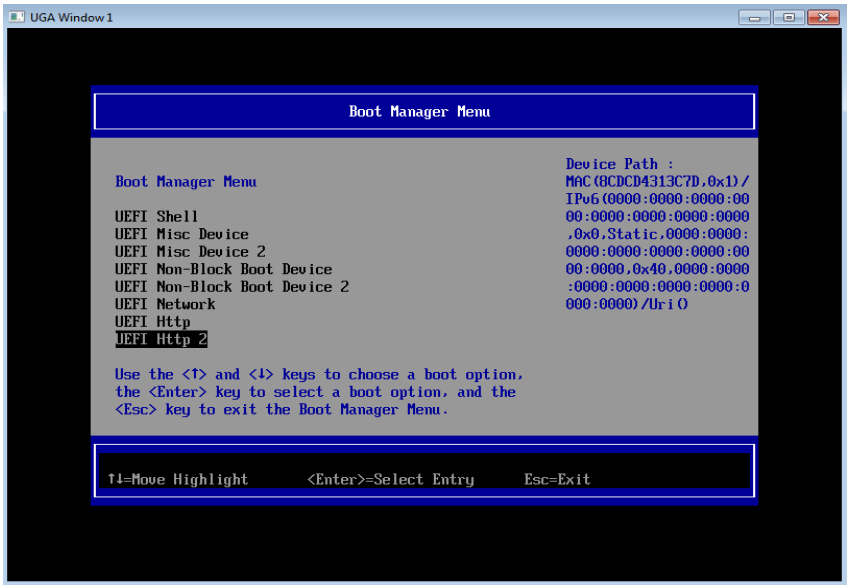

Figure 6: Select "UEFI Http2" to boot over IPv6 stack

Now the HTTP boot process is triggered over NT32 simulator. The HTTP driver stack will communicate with HTTP server, DNS server, and DHCP server to download an UEFI shell then boot to the downloaded shell. Figure 7 shows the result of this successful HTTP boot.

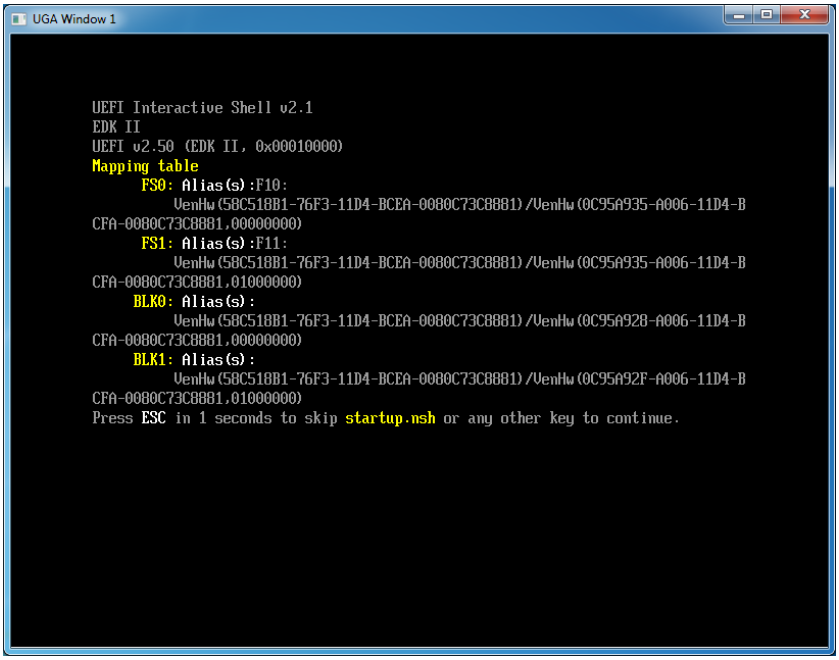

Figure 7: Boot the downloaded UEFI Shell

3 Enable HTTP boot for your system

HTTP boot is enabled in NT32 simulator by default. To enable HTTP boot over an IPv4 stack or an IPv6 stack on your system, first check whether your system has built in an EDKII network stack.

3.1 Enable HTTP Boot over IPv4 stack on a platform has EDKII network

If your system already has EDKII IPv4 network stack, you need update the network modules to latest revision.

1. Specifically, you need to update the MdePkg to get the new protocol definitions for UEFI 2.5.

2. After that, update following modules:

- IP4 Driver

```
MdeModulePkg\Universal\Network\Ip4Dxe\Ip4Dxe.inf
```

- NetLib Library

```
MdeModulePkg\Library\DxeNetLib\DxeNetLib.inf
```

3. Update your platform DSC file:

- Remove IP4Config Driver

```
MdeModulePkg\Universal\Network\Ip4ConfigDxe\Ip4ConfigDxe.inf
```

- Add HTTP and DNS Drivers

```
NetworkPkg\HttpDxe\HttpDxe.inf
NetworkPkg\HttpBootDxe\HttpBootDxe.inf
NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
NetworkPkg\DnsDxe\DnsDxe.inf
```

- Add HTTP Library to [LibraryClasses] section

```
HttpLib|MdeModulePkg\Library\DxeHttpLib\DxeHttpLib.inf
```

4. Finally, update your platform FDF file:

- Remove IP4Config Driver

```
INF MdeModulePkg\Universal\Network\Ip4ConfigDxe\Ip4ConfigDxe.inf
```

- Add HTTP and DNS Drivers

```
INF NetworkPkg\HttpDxe\HttpDxe.inf
INF NetworkPkg\HttpBootDxe\HttpBootDxe.inf
INF NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
INF NetworkPkg\DnsDxe\DnsDxe.inf
```

3.2 Enable HTTP Boot over an IPv4 stack on a platform without EDKII network

If you want to enable HTTP boot on a system without EDKII IPv4 network stack, you need add corresponding network modules to your platform files.

First, you need update the network modules to latest revision. The most convenient solution is to update MdePkg, MdeModulePkg, and NetworkPkg to latest revisions from an EDKII repository.

Then, update your platform files as follows:

1. Update your platform DSC file to add following drivers:

```
MdeModulePkg\Universal\Network\DpcDxe\DpcDxe.inf
MdeModulePkg\Universal\Network\SnpDxe\SnpDxe.inf
MdeModulePkg\Universal\Network\MnpDxe\MnpDxe.inf
MdeModulePkg\Universal\Network\ArpDxe\ArpDxe.inf
MdeModulePkg\Universal\Network\Ip4Dxe\Ip4Dxe.inf
MdeModulePkg\Universal\Network\Tcp4Dxe\Tcp4Dxe.inf
MdeModulePkg\Universal\Network\Udp4Dxe\Udp4Dxe.inf
MdeModulePkg\Universal\Network\Dhcp4Dxe\Dhcp4Dxe.inf
NetworkPkg\HttpDxe\HttpDxe.inf
NetworkPkg\HttpBootDxe\HttpBootDxe.inf
NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
NetworkPkg\DnsDxe\DnsDxe.inf
```

2. Update your platform DSC file to add following libraries to [LibraryClasses] section:

```
DpcLib|MdeModulePkg\Library\DxeDpcLib\DxeDpcLib.inf
NetLib|MdeModulePkg\Library\DxeNetLib\DxeNetLib.inf
IpIoLib|MdeModulePkg\Library\DxeIpIoLib\DxeIpIoLib.inf
UdpIoLib|MdeModulePkg\Library\DxeUdpIoLib\DxeUdpIoLib.inf
TcpIoLib|MdeModulePkg\Library\DxeTcpIoLib\DxeTcpIoLib.inf
HttpLib|MdeModulePkg\Library\DxeHttpLib\DxeHttpLib.inf
```

3. Update your platform FDF file to add following drivers:

```
INF MdeModulePkg\Universal\Network\DpcDxe\DpcDxe.inf
INF MdeModulePkg\Universal\Network\SnpDxe\SnpDxe.inf
INF MdeModulePkg\Universal\Network\MnpDxe\MnpDxe.inf
INF MdeModulePkg\Universal\Network\ArpDxe\ArpDxe.inf
INF MdeModulePkg\Universal\Network\Ip4Dxe\Ip4Dxe.inf
INF MdeModulePkg\Universal\Network\Tcp4Dxe\Tcp4Dxe.inf
INF MdeModulePkg\Universal\Network\Udp4Dxe\Udp4Dxe.inf
INF MdeModulePkg\Universal\Network\Dhcp4Dxe\Dhcp4Dxe.inf
INF NetworkPkg\HttpDxe\HttpDxe.inf
INF NetworkPkg\HttpBootDxe\HttpBootDxe.inf
INF NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
INF NetworkPkg\DnsDxe\DnsDxe.inf
```

Note: *You also need add your own UNDI driver to your platform files.*

3.3 Enable HTTP Boot over an IPv6 stack on a platform with EDKII network

If your system already has EDKII IPv6 network stack, you need update the network modules to latest revision.

1. Specifically, you need to update the MdePkg to get the new protocol definitions for UEFI 2.5.

2. After that, update following modules:

- NetLib Library

```
MdeModulePkg\Library\DxeNetLib\DxeNetLib.inf
```

3. Update your platform DSC file:

- Add HTTP and DNS Drivers

```
NetworkPkg\HttpDxe\HttpDxe.inf
NetworkPkg\HttpBootDxe\HttpBootDxe.inf
NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
NetworkPkg\DnsDxe\DnsDxe.inf
```

- Add HTTP Library to [LibraryClasses] section

```
HttpLib|MdeModulePkg\Library\DxeHttpLib\DxeHttpLib.inf
```

4. Finally, update your platform FDF file:

- Add HTTP and DNS Drivers

```
INF NetworkPkg\HttpDxe\HttpDxe.inf
INF NetworkPkg\HttpBootDxe\HttpBootDxe.inf
INF NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
INF NetworkPkg\DnsDxe\DnsDxe.inf
```

3.4 Enable HTTP Boot over IPv6 stack on a platform without EDKII network

If you want to enable HTTP boot on a system without EDKII IPv6 network stack, you need add corresponding network modules to your platform files.

First, you need update the network modules to latest revision. The most convenient solution is to update MdePkg, MdeModulePkg, and NetworkPkg to latest revisions from an EDKII repository.

Then, update your platform files as follows:

5. Update your platform DSC file to add following drivers:

```
MdeModulePkg\Universal\Network\DpcDxe\DpcDxe.inf
MdeModulePkg\Universal\Network\SnpDxe\SnpDxe.inf
MdeModulePkg\Universal\Network\MnpDxe\MnpDxe.inf

NetworkPkg\Ip6Dxe\Ip6Dxe.inf
NetworkPkg\TcpDxe\TcpDxe.inf
NetworkPkg\Udp6Dxe\Udp6Dxe.inf
NetworkPkg\Dhcp6Dxe\Dhcp6Dxe.inf
NetworkPkg\HttpDxe\HttpDxe.inf
NetworkPkg\HttpBootDxe\HttpBootDxe.inf
NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
NetworkPkg\DnsDxe\DnsDxe.inf
```

6. Update your platform DSC file to add following libraries to [LibraryClasses] section:

```
DpcLib|MdeModulePkg\Library\DxeDpcLib\DxeDpcLib.inf
NetLib|MdeModulePkg\Library\DxeNetLib\DxeNetLib.inf
IpIoLib|MdeModulePkg\Library\DxeIpIoLib\DxeIpIoLib.inf
UdpIoLib|MdeModulePkg\Library\DxeUdpIoLib\DxeUdpIoLib.inf
TcpIoLib|MdeModulePkg\Library\DxeTcpIoLib\DxeTcpIoLib.inf
HttpLib|MdeModulePkg\Library\DxeHttpLib\DxeHttpLib.inf
```

7. Update your platform FDF file to add following drivers:

```
INF MdeModulePkg\Universal\Network\DpcDxe\DpcDxe.inf
INF MdeModulePkg\Universal\Network\SnpDxe\SnpDxe.inf
INF MdeModulePkg\Universal\Network\MnpDxe\MnpDxe.inf

INF NetworkPkg\Ip6Dxe\Ip6Dxe.inf
INF NetworkPkg\TcpDxe\TcpDxe.inf
INF NetworkPkg\Udp6Dxe\Udp6Dxe.inf
INF NetworkPkg\Dhcp6Dxe\Dhcp6Dxe.inf
INF NetworkPkg\HttpDxe\HttpDxe.inf
INF NetworkPkg\HttpBootDxe\HttpBootDxe.inf
INF NetworkPkg\HttpUtilitiesDxe\HttpUtilitiesDxe.inf
INF NetworkPkg\DnsDxe\DnsDxe.inf
```

Note: *You also need to add your own UNDI driver to your platform files.*