

tianocore

Getting Started Guide of EDK II HTTP Boot

TABLE OF CONTENTS

Getting Started Guide of EDKII HTTP Boot

Feature Scope

Related Protocols

Feature Scope

Quick Start Guide

Network topology

Configure DHCPv4 server

Configure DHCPv6 server

Configure DNSv4 server

Configure DNSv6 server

Configure HTTP server

Build NT32 Simulator

Run HTTP boot

Enable HTTP boot for your system

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

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

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

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

Modify PCD setting to allow HTTP connections

Figures

Figures

Figure 1 HTTP boot test-bed

Figure 2 Configure DNSv4 Service

Figure 3 Configure DNSv4 Service

Figure 4 Configure HTTP Service - Tomcat

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

Figure 6 Select "UEFI Http 2" to boot over IPv6 stack

Figure 7 Boot the downloaded UEFI Shell



GETTING STARTED GUIDE OF EDK II HTTP BOOT

DRAFT FOR REVIEW

04/30/2025 11:38:17

Revision 0.90

WHITEPAPER

Contributed by

Ye Ting, Intel Corporation

Fu Siyuan, Intel Corporation

Zhang Lubo, Intel Corporation

Acknowledgements

Redistribution and use in source (original document form) and 'compiled' forms (converted to PDF, epub, HTML and other formats) with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (original document form) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.
2. Redistributions in compiled form (transformed to other DTDs, converted to PDF, epub, HTML and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY TIANOCORE PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL TIANOCORE PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2018, Intel Corporation. All rights reserved.

Revision History

Revision	Revision History	Date
0.50	Initial release.	August 2015
0.60	Include HTTP Utilities protocol and driver	August 2015
0.70	Add Http Boot over IPv6 stack	December 2015
0.80	Corrections to code in section 2.3.	April 2016

0.90	Add PCD setting for HTTP connection. Convert to GitBook	January 2018
------	---	--------------

FEATURE SCOPE

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

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.

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 an IPv6 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`

QUICK START GUIDE

This document assumes that readers have installed EDK II 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.

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.

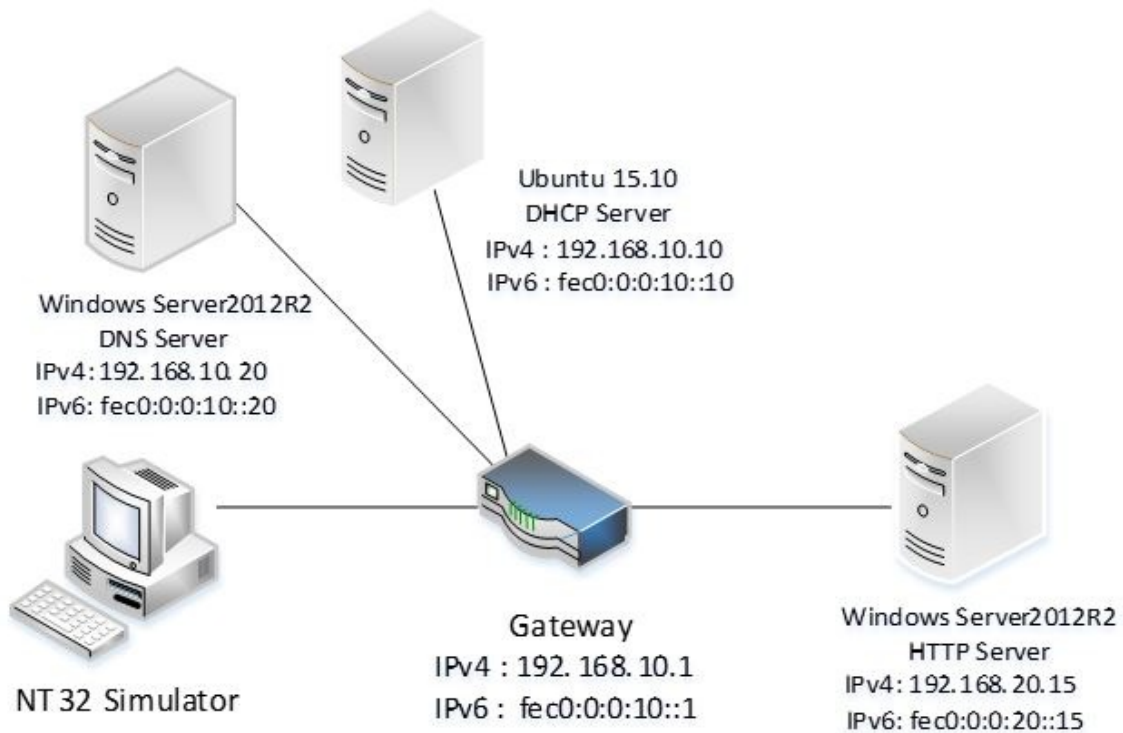


Figure 1 HTTP boot test-bed

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.

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`.

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 fec0:0:0:10:0:0: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`.

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.

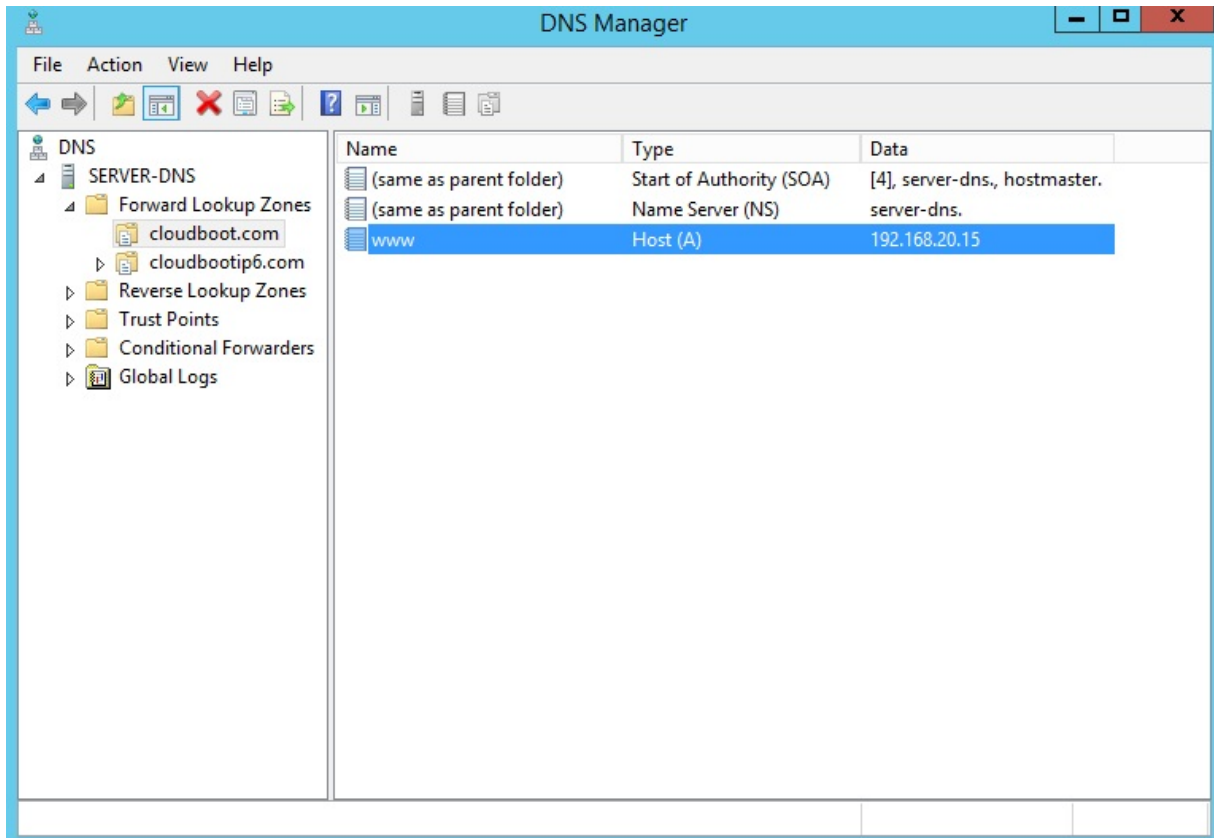


Figure 2 Configure DNSv4 Service

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.

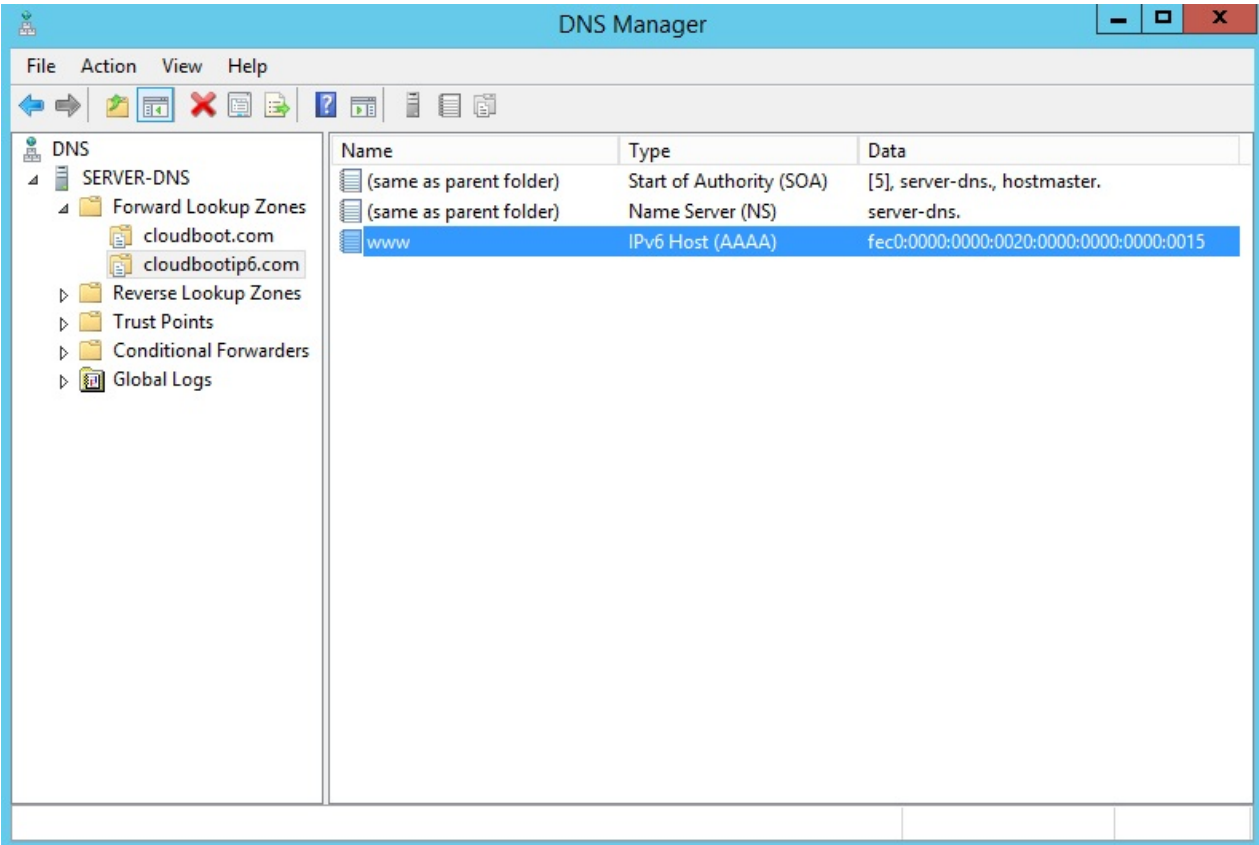


Figure 3 Configure DNSv6 Service

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.conf or dhcpd6.conf)

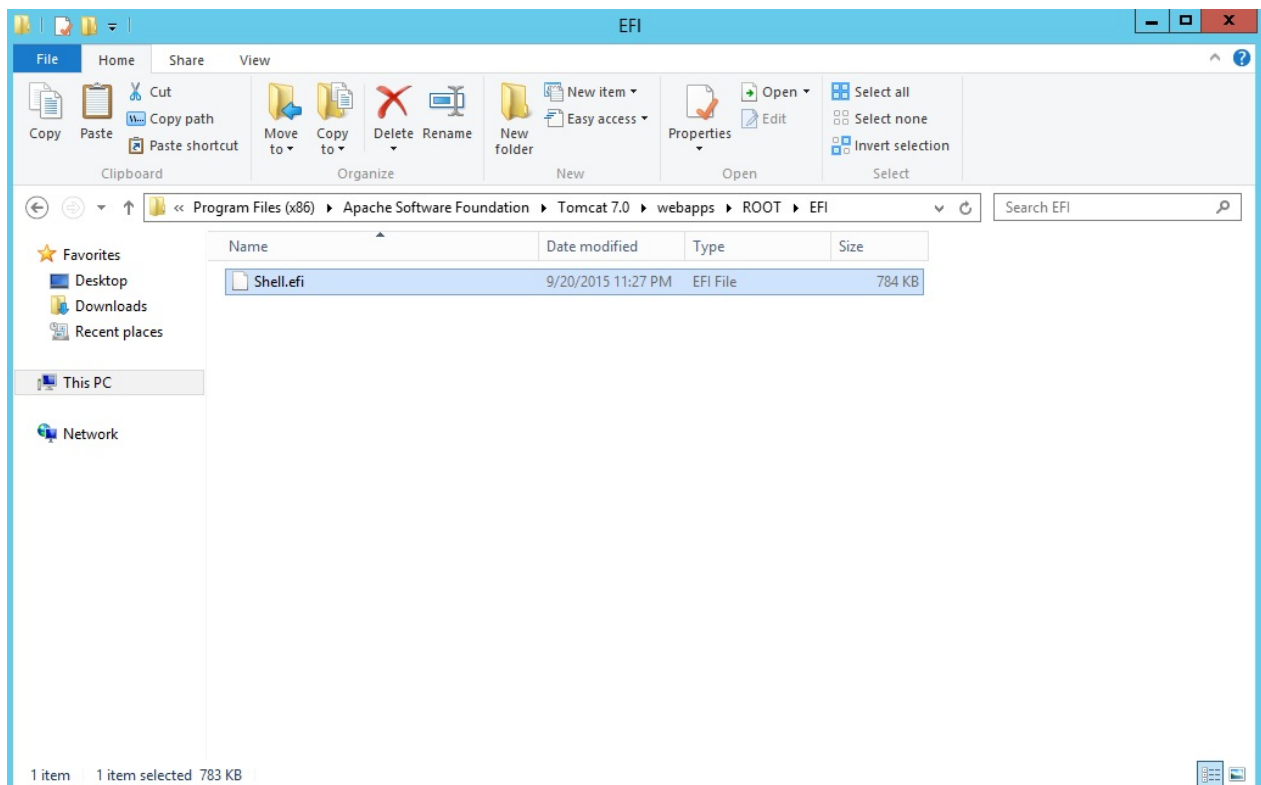


Figure 4 Configure HTTP Service - Tomcat

Build NT32 Simulator

Use the following command to build NT32 simulator:

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

The pictures shown in the next section are captured from NT32 simulator.

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).

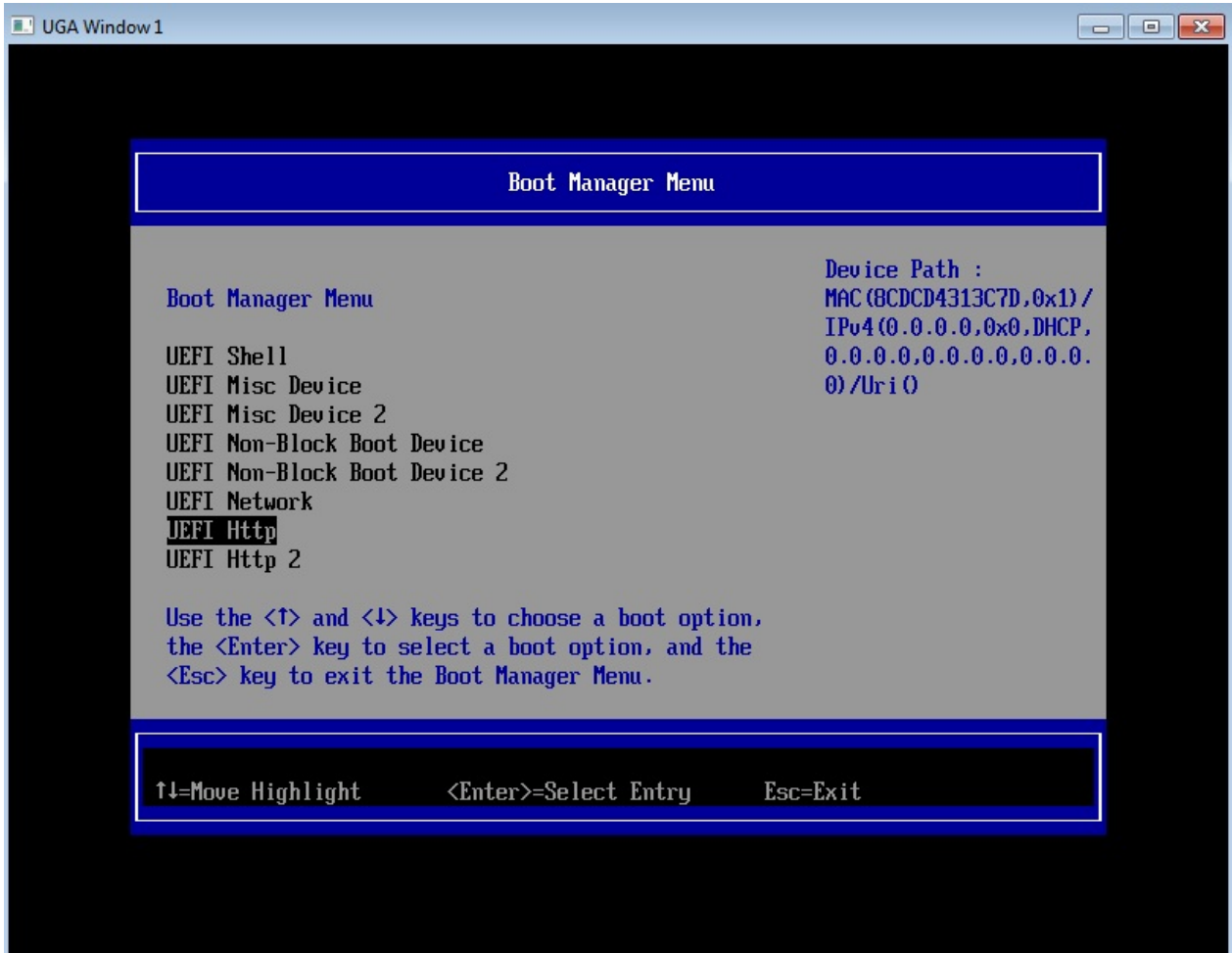
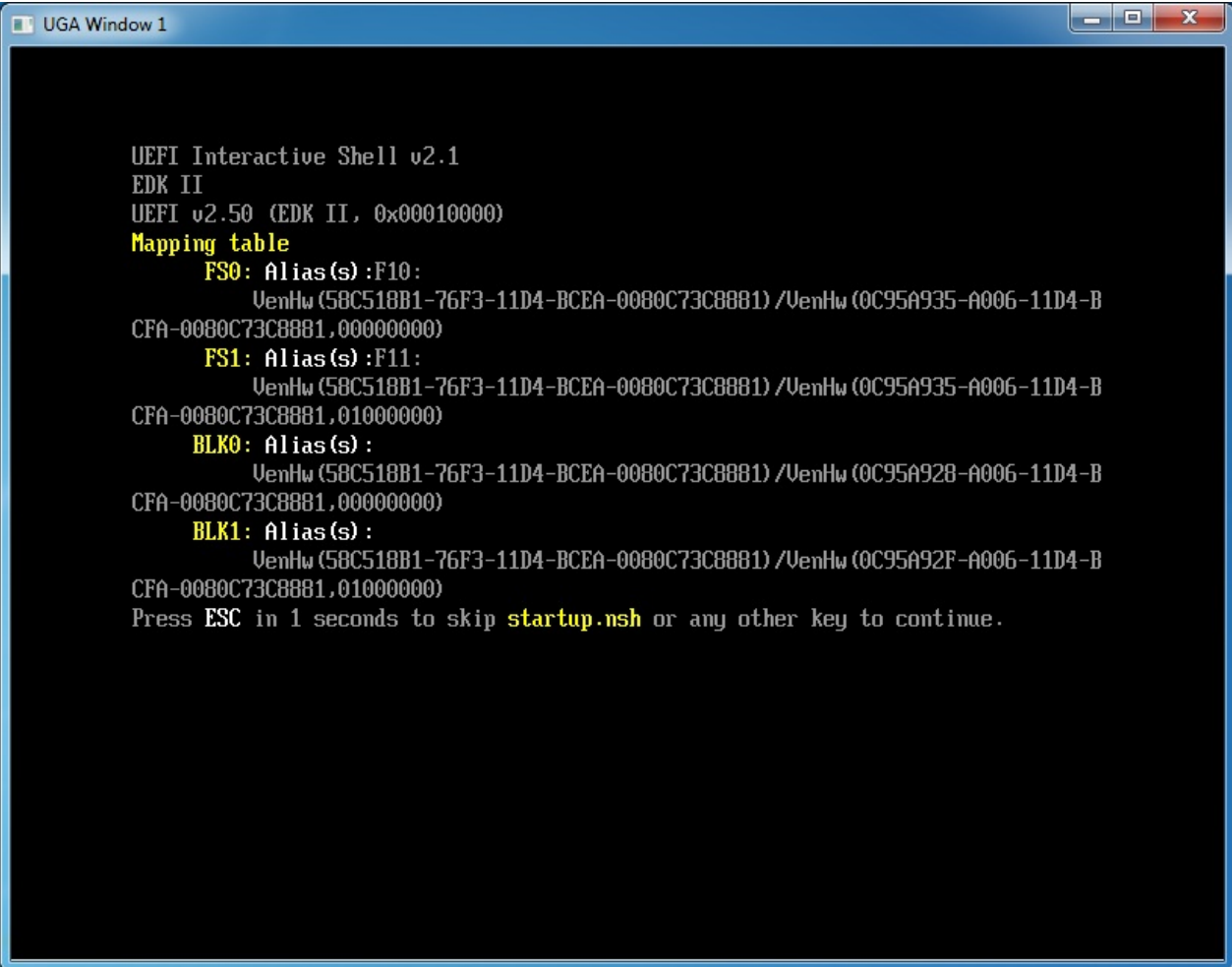


Figure 5 Select “UEFI Http” to boot over IPv4 stack



Figure 6 Select “UEFI Http 2” 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.



```
UGA Window 1

UEFI Interactive Shell v2.1
EDK II
UEFI v2.50 (EDK II, 0x00010000)
Mapping table
  FS0: Alias(s) :F10:
        VenHw (58C518B1-76F3-11D4-BCEA-0080C73C8881) /VenHw (0C95A935-A006-11D4-B
CFA-0080C73C8881,00000000)
  FS1: Alias(s) :F11:
        VenHw (58C518B1-76F3-11D4-BCEA-0080C73C8881) /VenHw (0C95A935-A006-11D4-B
CFA-0080C73C8881,01000000)
  BLK0: Alias(s) :
        VenHw (58C518B1-76F3-11D4-BCEA-0080C73C8881) /VenHw (0C95A928-A006-11D4-B
CFA-0080C73C8881,00000000)
  BLK1: Alias(s) :
        VenHw (58C518B1-76F3-11D4-BCEA-0080C73C8881) /VenHw (0C95A92F-A006-11D4-B
CFA-0080C73C8881,01000000)
Press ESC in 1 seconds to skip startup.nsh or any other key to continue.
```

Figure 7 Boot the downloaded UEFI Shell

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.

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
```

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.

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
```

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

If you want to enable HTTP boot on a system without EDK II 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 EDK II repository.

Then, update your platform files as follows:

1. (do steps 1-4 from the previous section) 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
```

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 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.

Modify PCD setting to allow HTTP connections

HTTP communication is forbidden by default due to security consideration (only HTTPS is allowed). You need to override the default PCD setting in your platform DSC file to allow HTTP connections.

1. Update your platform DSC file to add following line to [PcdsFixedAtBuild] section:

```
gEfiNetworkPkgTokenSpaceGuid.PcdAllowHttpConnections|TRUE
```


Figures

- [Figure 1 HTTP boot test-bed](#)
- [Figure 2 Configure DNSv4 Service](#)
- [Figure 3 Configure DNSv4 Service](#)
- [Figure 4 Configure HTTP Service - Tomcat](#)
- [Figure 5 Select “UEFI Http” to boot over IPv4 stack](#)
- [Figure 6 Select “UEFI Http 2” to boot over IPv6 stack](#)
- [Figure 7 Boot the downloaded UEFI Shell](#)
- [Figure 1 HTTP boot test-bed](#)
- [Figure 2 Configure DNSv4 Service](#)
- [Figure 3 Configure DNSv4 Service](#)
- [Figure 4 Configure HTTP Service - Tomcat](#)
- [Figure 5 Select “UEFI Http” to boot over IPv4 stack](#)
- [Figure 6 Select “UEFI Http 2” to boot over IPv6 stack](#)
- [Figure 7 Boot the downloaded UEFI Shell](#)