tianocore

Getting Started with UEFI HTTPS Boot on EDK II

TABLE OF CONTENTS

- Getting Started with UEFI HTTPS Boot on EDK II
- Introduction
  - Overview
  - Additional Protocols
  - Additional Modules
- HTTPS Authentication
  - TLS Authentication Modes
  - Self-Generated Certificate
- Start Guide
  - Configure Server and Build Client
    - Solution for IPv4
    - Solution for IPv6
  - Run HTTPS Boot
- Tables
  - Table 1 - Certificate Requirement
  - Table 2 - Key Pair
- Figures
  - Figure 1 - Authentication Mechanism
  - Figure 2 - HTTPS Boot, IPv4 Configuration
  - Figure 3 - DHCPv4 Server Scope
  - Figure 4 - DHCPv4 Server Options
  - Figure 5 - Configure New Host for IPv4
  - Figure 6 - Add MIME Type
  - Figure 7 - Add MIME Type
  - Figure 8 - Add Server Certificates
  - Figure 9 - Enroll a Certificate for the HTTPS Server
  - Figure 10 - Create a New Website for the HTTPS Server
  - Figure 11 - The UEFI Shell file, as viewed in IIS
  - Figure 12 - HTTPS boot, IPv6 Configuration
  - Figure 13 - Configure Forward Lookup Zone for IPv6
  - Figure 14 - Configure New Host for IPv6
  - Figure 15 - UEFI Client Certificate Configuration
  - Figure 16 - Select Boot Option
  - Figure 17 - Boot the Downloaded UEFI Shell Image

Getting Started with UEFI HTTPS Boot on EDK II

DRAFT FOR REVIEW

05/01/2025 12:14:05

Revision 1.30

WHITEPAPER

Contributed by

Wu Jiaxin

Fu Siyuan

Brian Richardson

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) 2017, Intel Corporation. All rights reserved.

Revision History

Revision	Revision History	Date
0.1	Initial release.	March 2016
0.2	Add UEFI Client Certificate Configuration	May 2016
0.3	Simplify the topology and configuration.	May 2016

0.4	Refine the content	May 2016
0.5	Minor content revisions	May 2016
0.6	Updated figure references & document formatting. Text cleanup for information related to IPv4 & IPv6 configuration (multiple sections).	June 2016
0.7	Modified document title. Added information on openSUSE implementation of HTTP Boot.	July 2016
0.8	Refine the contents.	Sep 2016
0.9	Update the certificate generation command for Windows OS	Sep 2016
1.0	Remove the consumed OpensslTlsLib module since the ssl building has been enabled in OpensslLib directly.	Dec 2016
1.1	Minor formatting changes. Fixed hyperlinks to tianocore.org site and whitepapers.	Feb 2017
1.2	Add the IANA approved media type link. Specify the EDKII network stack version.	Feb 2017
1.3	Replace examples of md5 signature algorithm with sha256. Added note on PXE to Overview Section with links to TianoCore wiki. Conversion to gitBook	Apr 2017

INTRODUCTION

Overview

HTTP over TLS (HTTPS) boot is a standard implementation for securely booting using the Unified Extensible Firmware Interface (UEFI) over a network device. HTTPS Boot is especially important for clients using potentially insecure networks outside of corporate infrastructure. Security for UEFI HTTPS Boot is provided by the underlying Transport Layer Security (TLS).

UEFI HTTPS Boot is designed to overcome limitations of the Preboot Execution Environment (PXE) boot method currently supported by UEFI & legacy BIOS firmware:

- PXE uses UDP as transport layer protocol. TCP is not supported.
- PXE is designed to work within a corporate network, not outside of a company firewall.
- PXE uses TFTP and does not support a secure transport method (ex: HTTPS).

This document assumes that the reader is familiar with the EDK II HTTP Boot Getting Started Guide available on the TianoCore whitepapers page. For information on configuring a HTTP Boot server, please refer to the UEFI HTTP Boot with OVMF help page available at opensuse.org.

Additional Protocols

All protocols introduced in the EDK II HTTP Boot Getting Started Guide are necessary.

The following new protocols are related to HTTPS Boot:

- EFI_TLS_SERVICE_BINDING_PROTOCOL
- EFI_TLS_PROTOCOL
- EFI_TLS_CONFIGURATION_PROTOCOL

Additional Modules

All modules introduced in the EDK II HTTP Boot Getting Started Guide are necessary. `HttpDxe` driver needs to be updated to consume the `TlsDxe` driver.

The following new TLS modules are also required by HTTPS boot:

- OpenSSL Crypto and TLS module
  `CryptoPkg/Library/OpensslLib/OpensslLib.inf`
- Base Crypto Library
  `CryptoPkg/Library/BaseCryptLib/BaseCryptLib.inf`
- TLS Library
  `CryptoPkg/Library/TlsLib/TlsLib.inf`
- TLS Driver
  `NetworkPkg/TlsDxe/TlsDxe.inf`
- TLS Authentication Config Driver
  `NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf`

HTTPS AUTHENTICATION

Figure 1 shows the regular HTTPS authentication mechanism for both the server providing the boot image, and the client booting from the image. Leveraging an asymmetric crypto system, the client and server can be authenticated by each other. The steps for mutual authentication are as follows:

1. Server and Client request the corresponding asymmetric key pair from the Certification Authority (CA). Both requested certificates can be verified by the CA.
2. The CA distributes the key pair (`servercert/serverkey`) and its own certificate (`rootcert`) to the Server. The distributed certificate (servercert) has been signed by its rootkey.
3. The CA distributes the key pair (`clientcert/clientkey`) and its own certificate (`rootcert`) to the Client. The distributed certificate (`clientcert`) has been signed by its rootkey.
4. Both Server and Client present their own certificate to each other for mutual authentication.
5. When the Server receives the Client certificate (`clientcert`) the certificate will be verified by rootcert, since it has been signed with the rootkey (and vice versa).

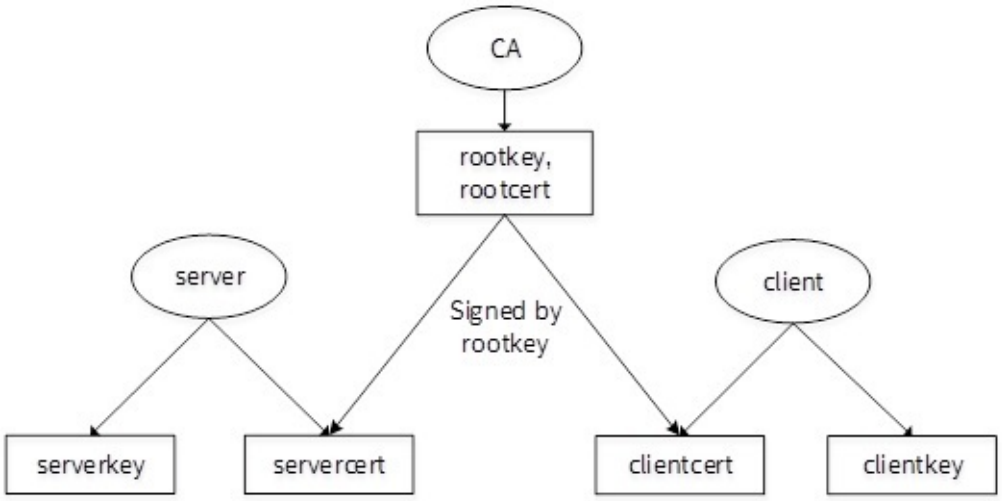

Figure 1 Authentication Mechanism

TLS Authentication Modes

TLS supports three authentication modes:

1. Two-way authentication: authentication of both parties. In this mode, both server and client will be authenticated.
2. One-way authentication: server authentication with an unauthenticated client. That means only the server is authenticated by the client, and the client won't be authenticated by the server.
3. Total anonymity: the server and client won't authenticate each other.

Table 1 shows the certificate requirement in each authentication mode for the HTTPS client and HTTPS server.

Part → --- Mode ↓	Authentication of both parties	Server authentication with an unauthenticated client	Total anonymity
HTTPS Client	rootcert, clientcert, clientkey	Rootcert	NULL
HTTPS Server	rootcert, servercert, serverkey	servercert, serverkey	servercert, serverkey

Table 1 Certificate Requirement

Self-Generated Certificate

This example shows how vendors can generate custom certificates for HTTPS Boot:

(1.) Install OpenSSL.
Windows:
Download and install an OpenSSL binary distribution. This document uses Win32 OpenSSL as an example.
Linux (Ubuntu as example):
`sudo apt-get install openssl`

(2.) Create a self-signed CA Certificate:
Note: (Use `type` command instead of `cat` in Windows) in the following examples

```
openssl req -new -sha256 -keyout rootkey.pem -out rootreq.pem -days 3650

openssl x509 -req -in rootreq.pem -sha256 -signkey rootkey.pem -out rootcert.pem -days 3650

cat rootcert.pem rootkey.pem > root.pem
```

(3.) Create a server certificate signed by the CA certificate:

```
openssl req -new -sha256 -keyout serverkey.pem -out serverreq.pem -days 3650

openssl x509 -req -in serverreq.pem -sha256 -CA root.pem -CAkey root.pem -CAcreateserial -out servercert.pem -days 3650

cat servercert.pem serverkey.pem root.pem > server.pem


openssl pkcs12 -export -in server.pem -out server.pfx
```

Note: The .pem file is encoded as BASE64, but only PKCS12 format key can be used when booting to a Microsoft Windows server. This requires the last step in process above, converting `server.pem` to `server.pfx`.

(4.) Create a client certificate signed by the CA certificate:

```
openssl req -new -sha256 -keyout clientkey.pem -out clientreq.pem -days 3650

openssl x509 -req -in clientreq.pem -sha256 -CA root.pem -CAkey root.pem -CAcreateserial -out clientcert.pem -days 3650

cat clientcert.pem clientkey.pem root.pem > client.pem
```

Using the steps above, the required key pairs are generated as shown in Table 2:

CA	**`rootkey.pem, rootcert.pem, root.pem`**
Server	`serverkey.pem, servercert.pem, server.pem, server.pfx`
Client	`clientkey.pem, clientcert.pem, client.pem`

Table 2 Key Pair

The next section demonstrates how to use ‘`rootcert.pem`’ and ‘`server.pfx`’ to enable server authentication with an unauthenticated client (one-way authentication).

START GUIDE

This guide gives instructions on how to set up a UEFI HTTPS Boot environment for both IPv4 and IPv6 network environments. This section assumes the reader has installed EDK II, and can build and run the NT32 simulator. The NT32 simulator serves as the UEFI HTTPS client.

Configure Server and Build Client

A UEFI HTTPS boot server has three main roles:

1. DHCP server
2. DNS server
3. HTTPS server

Depending on server requirements, two test-bed solutions are presented for reference: one simple approach for IPv4, and an advanced solution using IPv6. Users can select the proper scenario based on individual requirements. Self-generated certificates from Table 2 (‘ `rootcert.pem` ’ and ‘ `server.pfx` ’) are used for HTTPS one-way authentication.

Solution for IPv4

The solution documented in this section uses a single server for the DHCP, DNS and HTTPS functions. This is considered the simplest server configuration for UEFI HTTPS Boot.

Network Topology for IPv4

This example is based on Microsoft Windows Server 2012 R2. Internet Information Services (IIS) are used to configure HTTPS server. The server and NT32 simulator use the same IPv4 subnet (192.168.10.0) as shown in Figure 2.

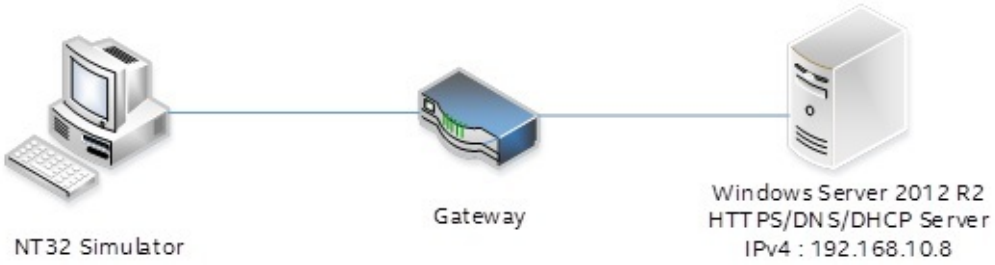

Figure 2 HTTPS Boot, IPv4 Configuration

Configure DHCPv4 Server

The steps to configure a DHCPv4 server are as follows:

1. Add a DHCP service in Windows Server. Please refer to the installation steps available here: http://thetechnosolution.com/installing-and-configuring-dhcp-on-windows-server-2012-r2/.
2. Right click on ‘IPv4 – New Scope’ to create a new scope option for IPv4 including the scope name, address range, and IP address lease duration. See Figure 3 for details.

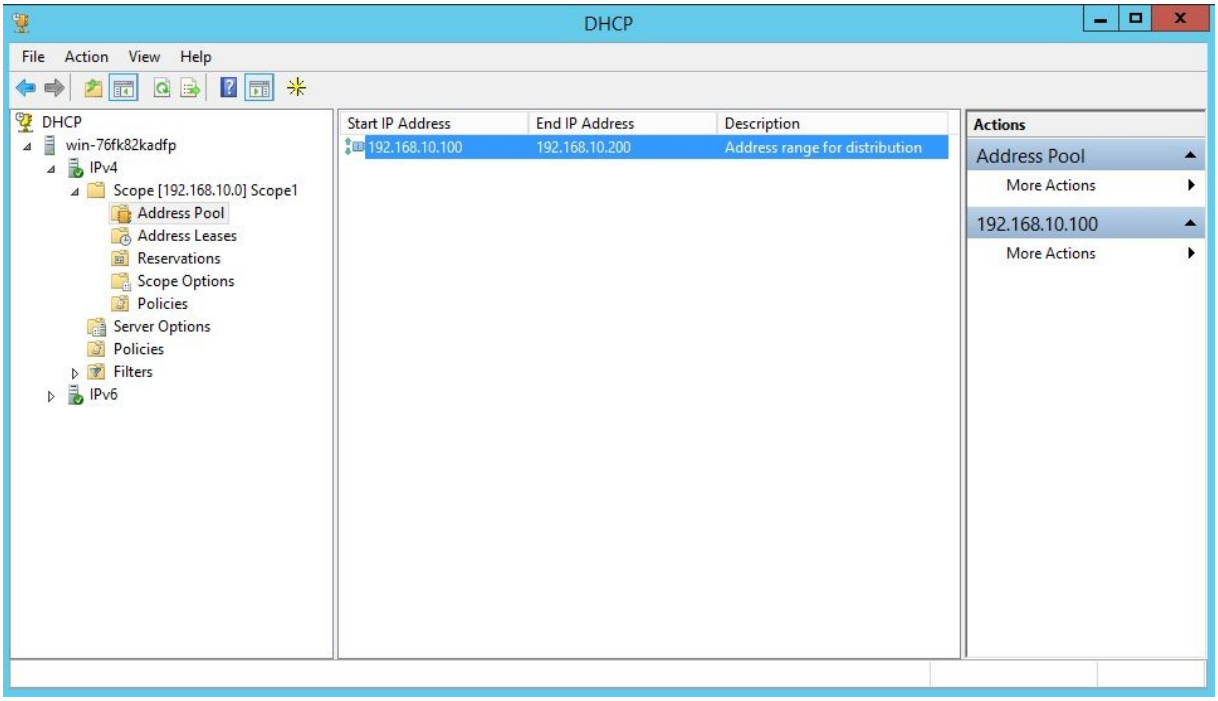

Figure 3 DHCPv4 Server Scope

3. Right click ‘Server Options – Configure Options…’ to configure IPv4 options including option 6, 60 and 67. These options must be configured for proper functionality. After configuration, the options should appear as shown in Figure 4. If the corresponding option code doesn’t appear in ‘Server Options – Configure Options…’ then right click ‘IPv4 – Set Predefined Options’, and click the ‘Add’ button to add the predefined option.
   a. Option 6 indicates the DNS server address.
   b. Option 60 defines the vendor Class ID. The value should be set to ‘HTTPClient’.
   c. Option 67 contains the corresponding boot file URI.

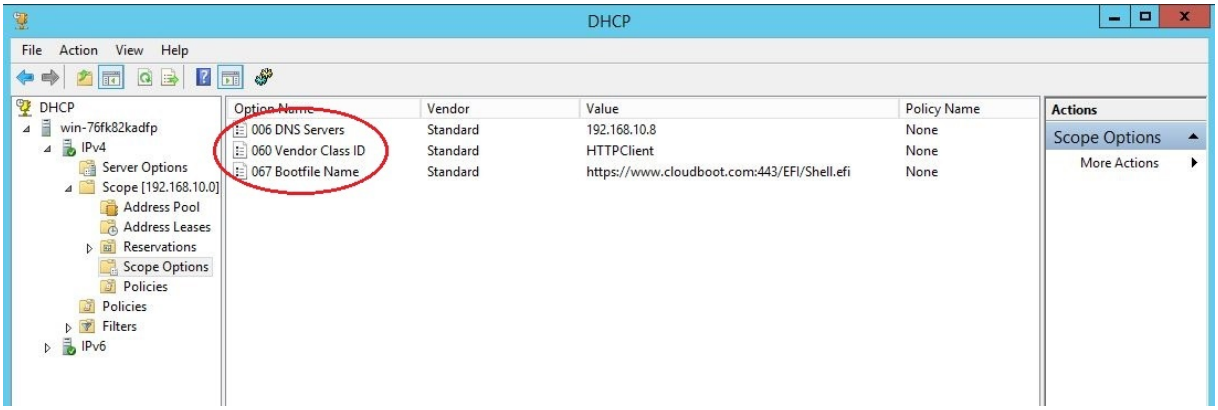

Figure 4 DHCPv4 Server Options

4. Right click the DHCP server name and select the ‘All Tasks – Restart’ option to restart the DHCPv4 service.

Configure DNSv4 Server

The steps to configure the DNSv4 server are as follows:

1. Add the DNS service in Windows Server Manager – ‘Add roles and features’.
2. Add a new forward lookup zone named ‘cloudboot.com’.
3. Add a new Host “www” for IPv4 (192.168.10.8). See Figure 5 for reference.

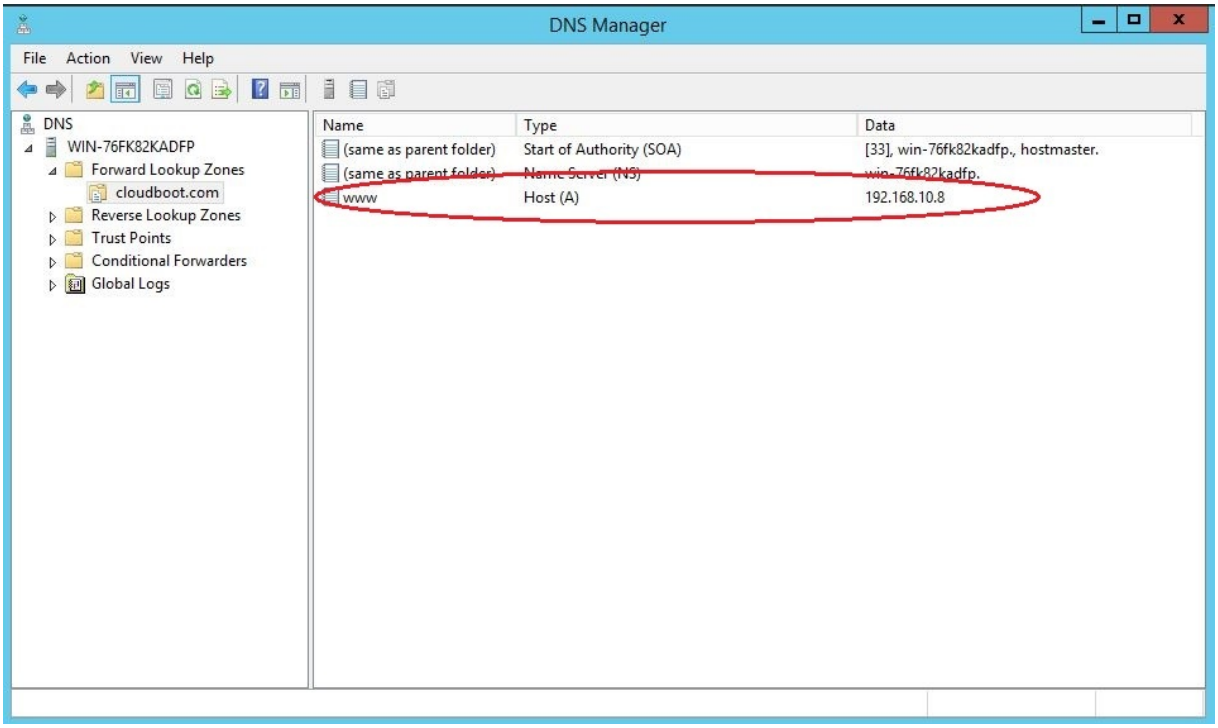

Figure 5 Configure New Host for IPv4

4. Right click the DHCP server name and select the 'All Tasks – Restart' option to restart the DHCPv4 service.

Configure HTTPS Server for IPv4

The steps to configure the HTTPS server are as follows:

1. Enable the Internet Information Services (IIS) feature in Windows Server manager, based on installation steps available here: http://www.iis.net/learn/install/installing-iis-85/installing-iis-85-on-windows-server-2012-r2.
2. Open the Internet Information Services (IIS) Manager, and add a new MIME type for the resources required by the HTTPS server. For the approved media type by IANA (e.g. .efi/.img/*.iso), please refer to the http://www.iana.org/assignments/media-types. In this example, the client will boot to a UEFI Shell image provided by the server. This requires addition of the `.efi` file type. Figure 6 and Figure 7 show the detailed steps.

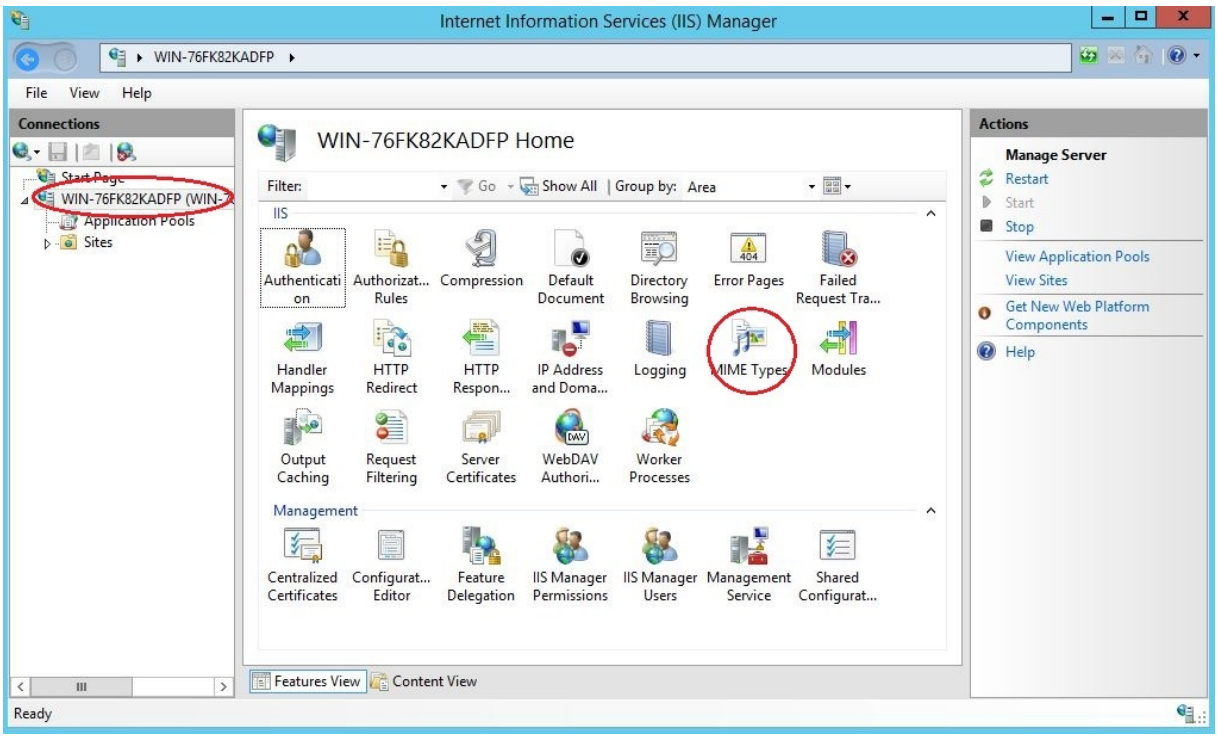

Figure 6 Add MIME Type

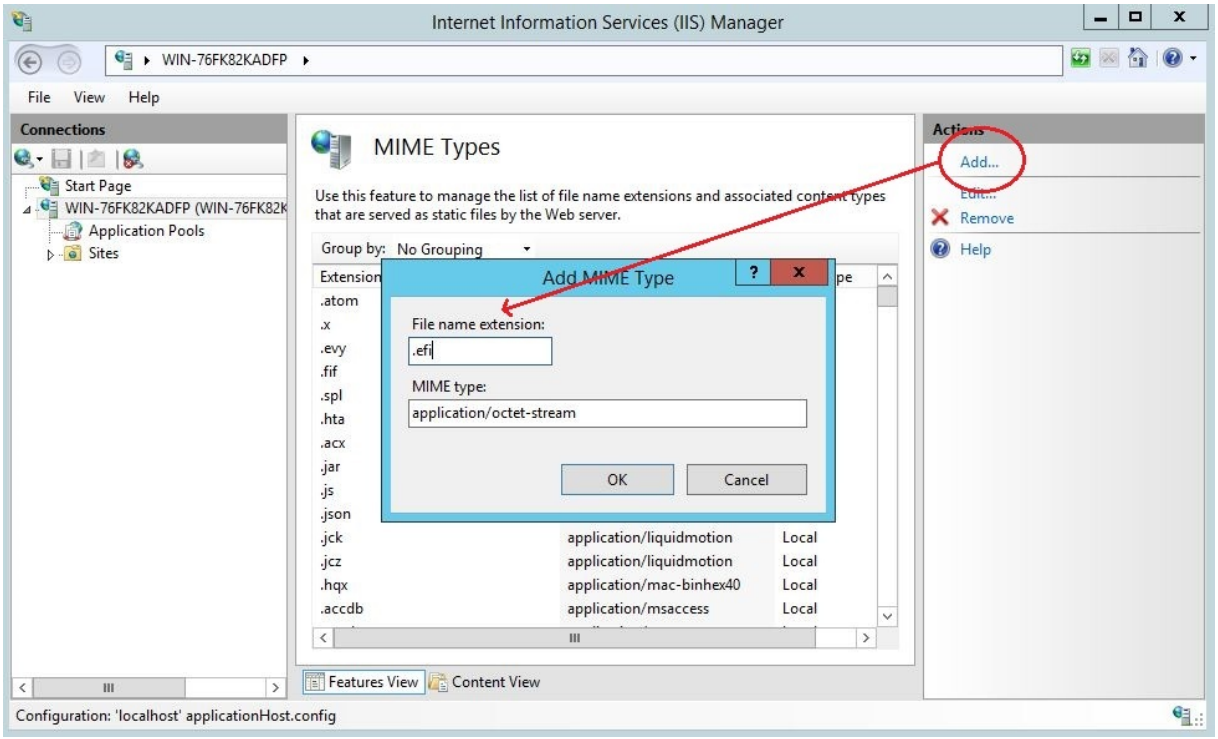

Figure 7 Add a New MIME Type to IIS

3. Enroll the Server key pair (`server.pfx`) in ‘Server Certificates’. Refer to Figure 8 and Figure 9 for details. Here, we assume the `server.pfx` has been generated. For detailed steps, please refer to section Self-Generated Certificate

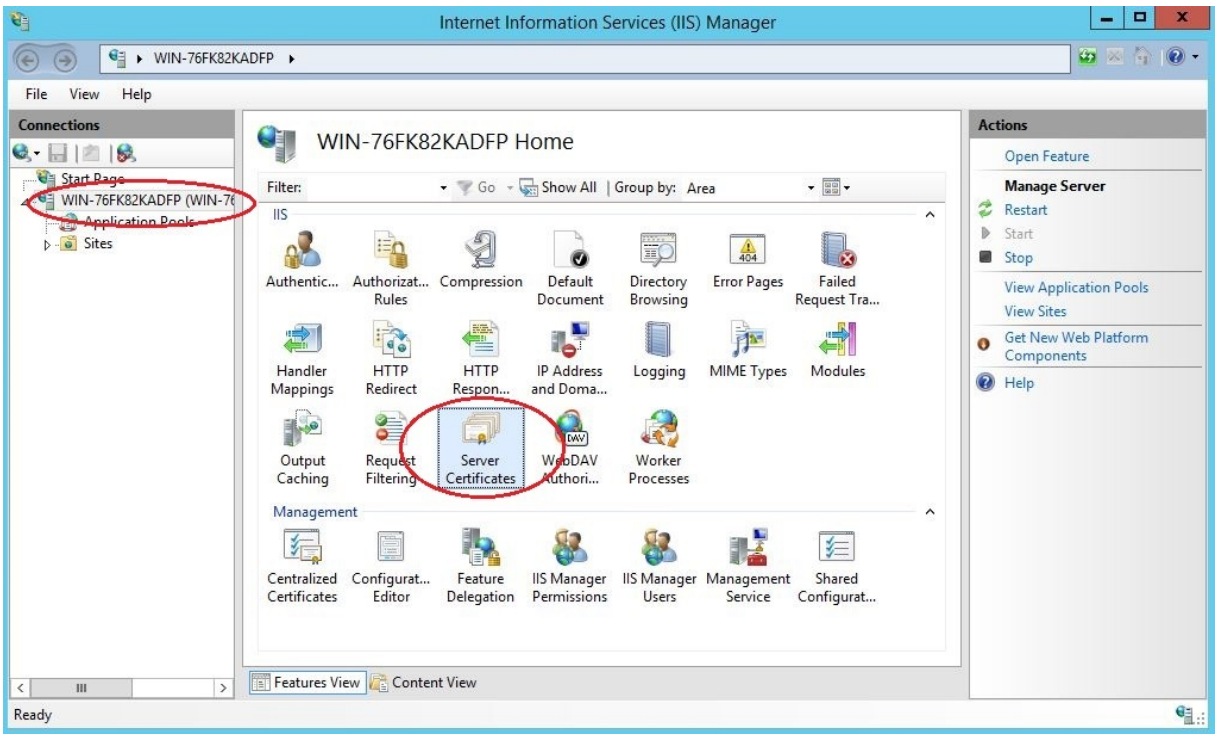

Figure 8 Add Server Certificates

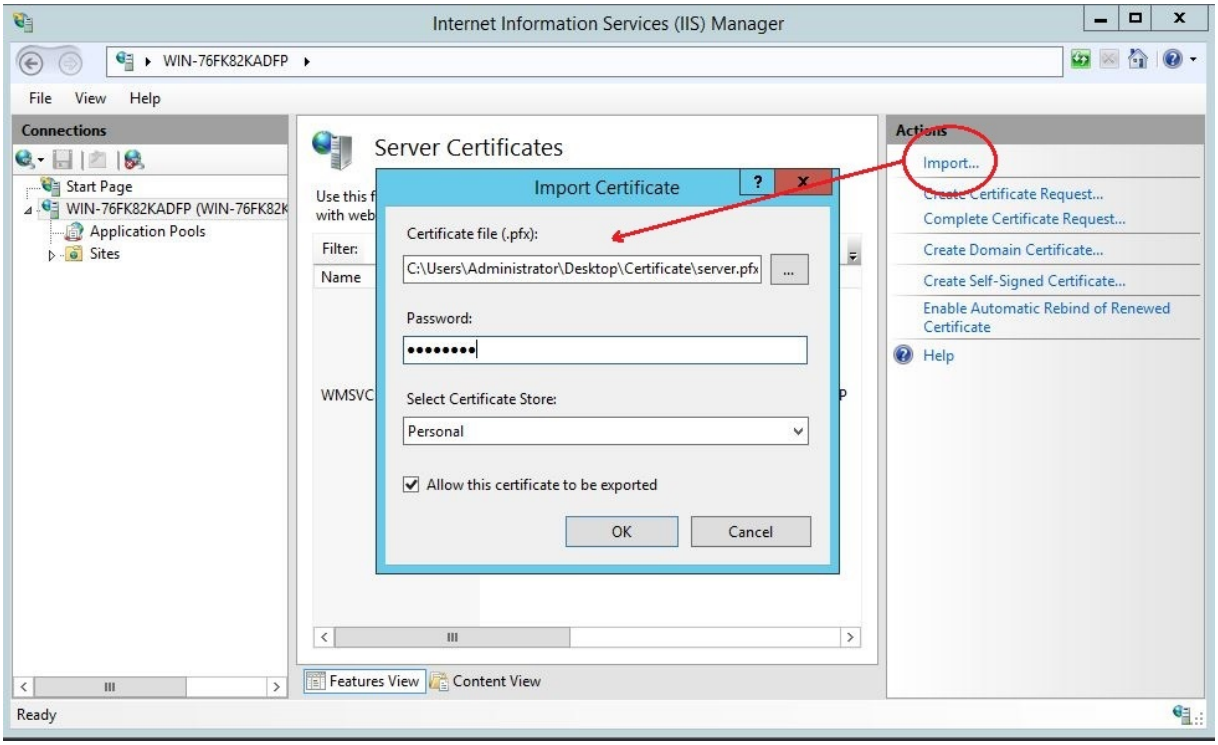

Figure 9 Enroll a Certificate for the HTTPS Server

4. Create a `'httpsroot'` folder in '`C:\inetpub`' as a default root path (`C:\inetpub\httpsroot`).
5. Right-click on 'Sites – Add Website' to create a new website for the HTTPS server. The areas highlighted in Figure 10 are required fields. The 'Physical path' is the default root path for the website. The 'SSL certificate' is the server key's (`server.pfx`) common name (`192.168.10.8`), which was enrolled in Step 3. The binding type is 'https' and the binding port value is '`443`'.

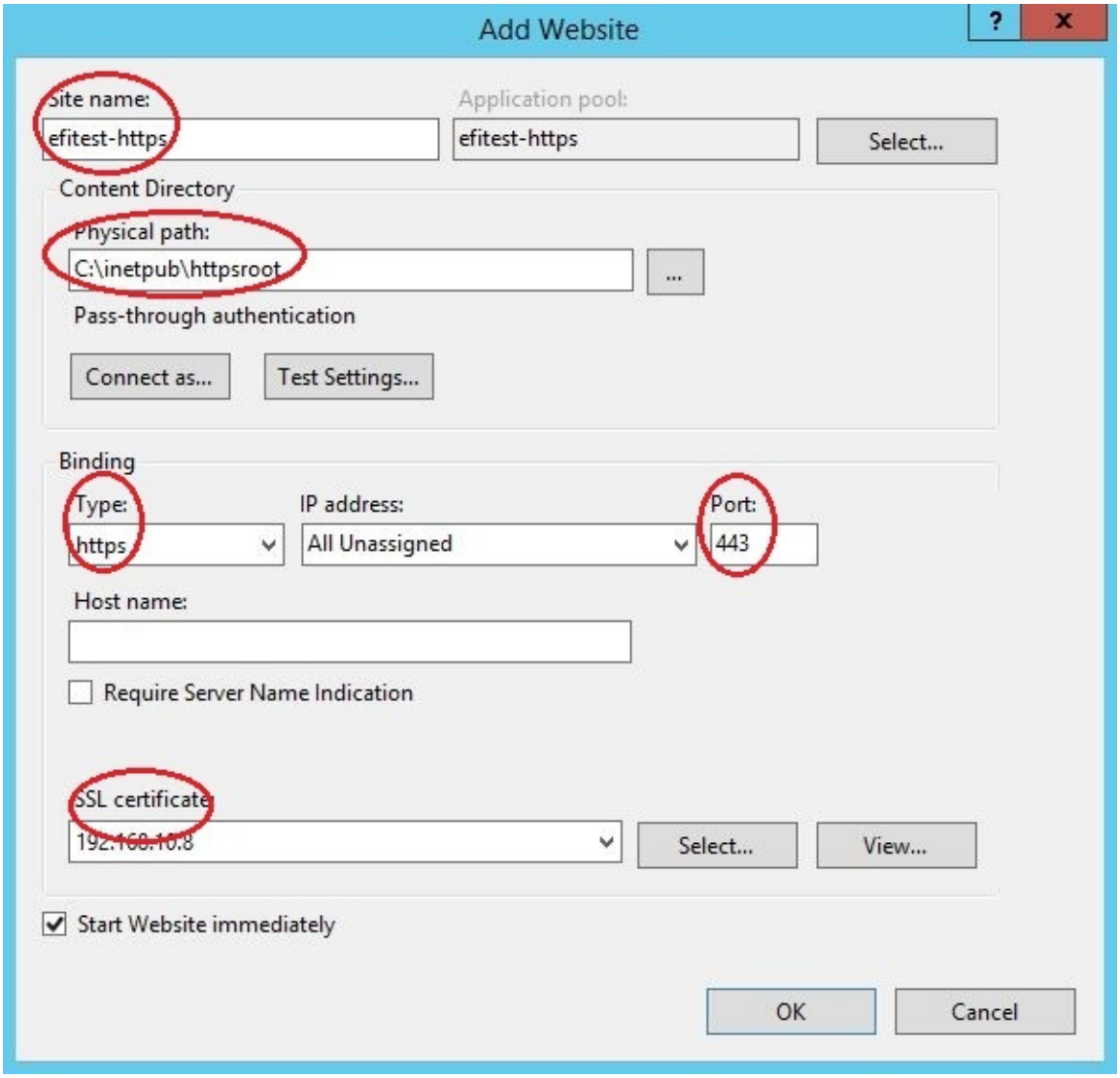

Figure 10 Create a New Website for the HTTPS Server

6. Create an 'EFI' folder in default root path, which was configured in Step 5. Copy the UEFI Shell binary that matches your firmware configuration into this folder (`C:\inetpub\httpsroot\EFI`). The UEFI Shell binary is in the `ShellBinPkg` package on EDK II (https://github.com/tianocore/edk2/tree/master/ShellBinPkg).The file should be renamed `Shell.efi` to match the configuration in DHCP option 67. This sets the UEFI Shell boot path as https://www.cloudboot.com:443/EFI/Shell.efi

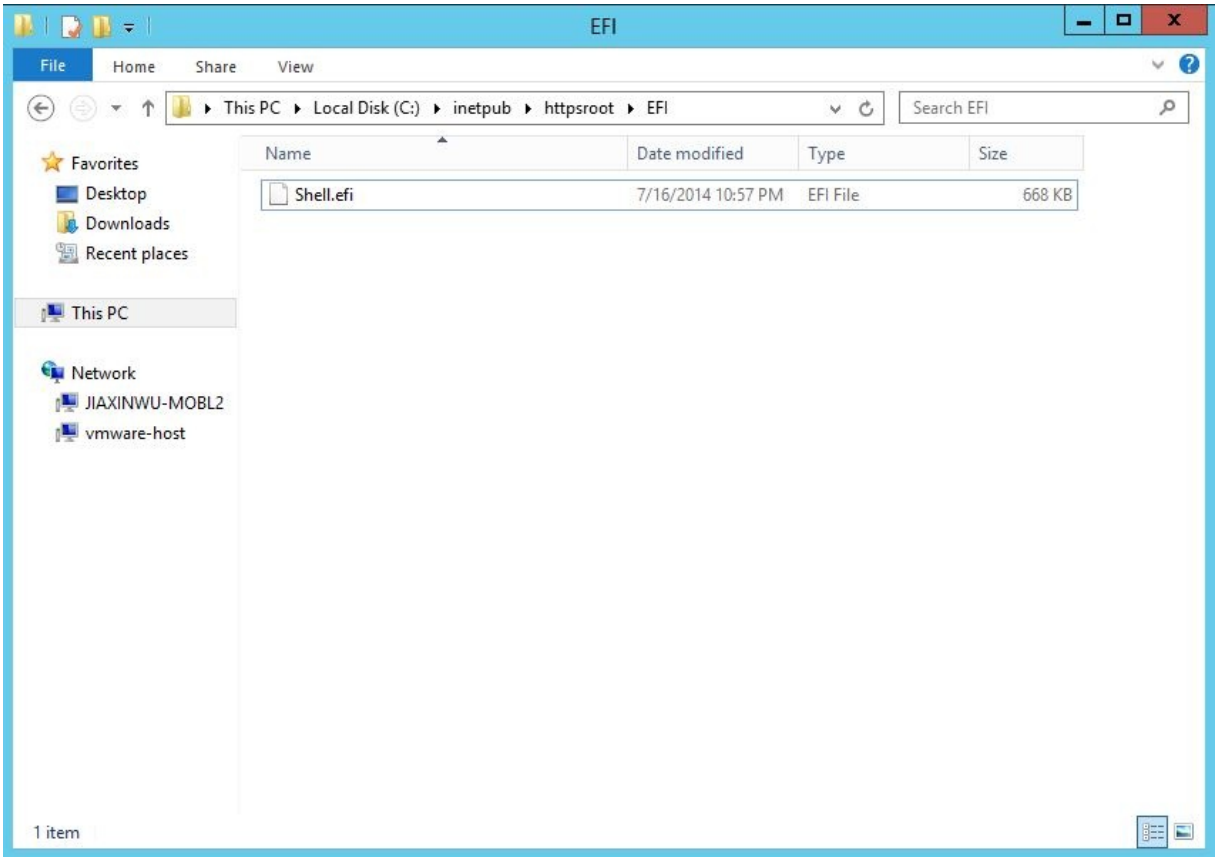

Figure 11 The UEFI Shell file, as viewed in IIS

Note: The NT32 Simulator uses the IA32 UEFI Shell binary, while most production systems require the x64 UEFI Shell to match the UEFI firmware configuration. This depends on your platform firmware configuration.

Enable NT32 Simulator for IPv4

To enable the UEFI HTTPSv4 Boot feature, the EDKII network stack (IPv4) must be built in your system firmware, which is located at MdeModulePkg/Universal/Network. Here, the verified version is: **d52f9163debb523e06d49ed8a4627a0317bab92c**.

Modules in DSC file

The following libraries and drivers are required by HTTPSv4 boot:

Add the following libraries to the `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
OpensslLib|CryptoPkg/Library/OpensslLib/OpensslLib.inf
BaseCryptLib|CryptoPkg/Library/BaseCryptLib/BaseCryptLib.inf
TlsLib|CryptoPkg/Library/TlsLib/TlsLib.inf
```

Add the following drivers to the `Components` section:

```
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
NetworkPkg/TlsDxe/TlsDxe.inf
NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Note: The network controller's UNDI driver also needs to be in the list of platform files.

Modules in FDF file

The following drivers should be added to the `FV` section for HTTPSv4 boot:

```
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
INF NetworkPkg/TlsDxe/TlsDxe.inf
INF NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Build the NT32 Simulator

The following command is used to build NT32 using Microsoft Visual Studio 2013:

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

Solution for IPv6

For IPv6, the DHCP, DNS and HTTPS server are deployed on different systems. This solution provides a more flexible configuration for the DHCP server, DNS server and HTTPS Server.

Network Topology for IPv6

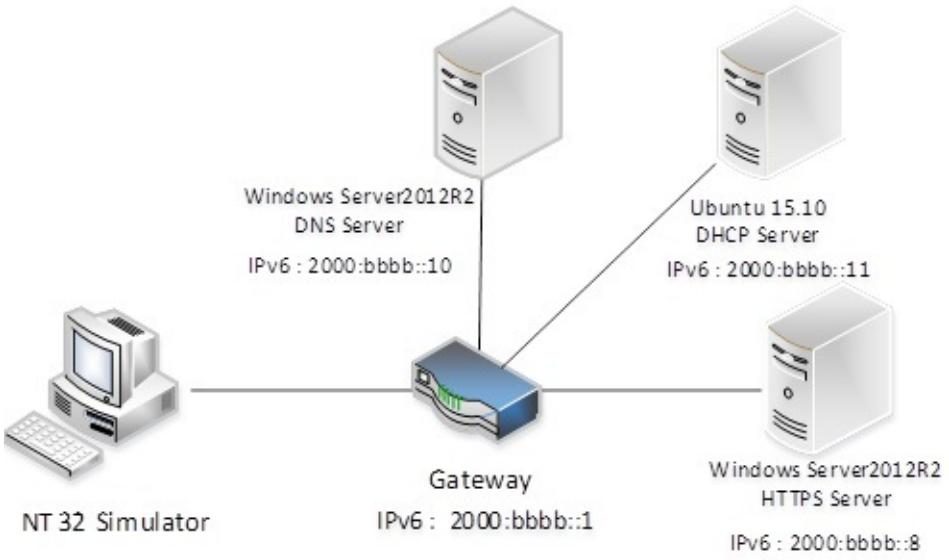

Figure 12 HTTPS boot, IPv6 Configuration

Configure the DHCPv6 Server

The steps to configure DHCPv6 on an Ubuntu 15.10 server are shown as follows:

1. Install the DHCP server: `sudo apt-get install isc-dhcp-server`
2. Edit `/etc/dhcp/dhcpd6.conf` as shown below
   Note: If there is no dhcpd6.conf file in `/etc/dhcp/`, create it first.

```
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, tring};
option dhcp6.bootfile-url code 59 = string;
subnet6 2000:bbbb::/64 {
#Range for clients
  range6 2000:bbbb::100 2000:bbbb::ffff;
  option dhcp6.domain-search “cloudboot.com”;
  option dhcp6.name-servers 2000:bbbb::10;
  option dhcp6.vendor-class 0 0 “HTTPClient”;
  “https://www.cloudboot.com:443/EFI/Shell.efi”;
}
```

3. Configure the server to listen for DHCP requests on the correct network interface. This example assumes eth0 is the primary interface. Edit the `/etc/default/isc-dhcp-server` file to configure `INTERFACE = “eth0”;`
4. Restart the DHCPv6 service: `sudo service isc-dhcp-server6 restart`

Configure DNSv6 Server

The steps to configure DNSv6 for Microsoft Windows Server 2002 R2 are as follows:

1. Add the DNS service in Windows Server Manager – ‘Add roles and features’.
2. Add a new forward lookup zone ‘`cloudboot.com`’ (see Figure 13).

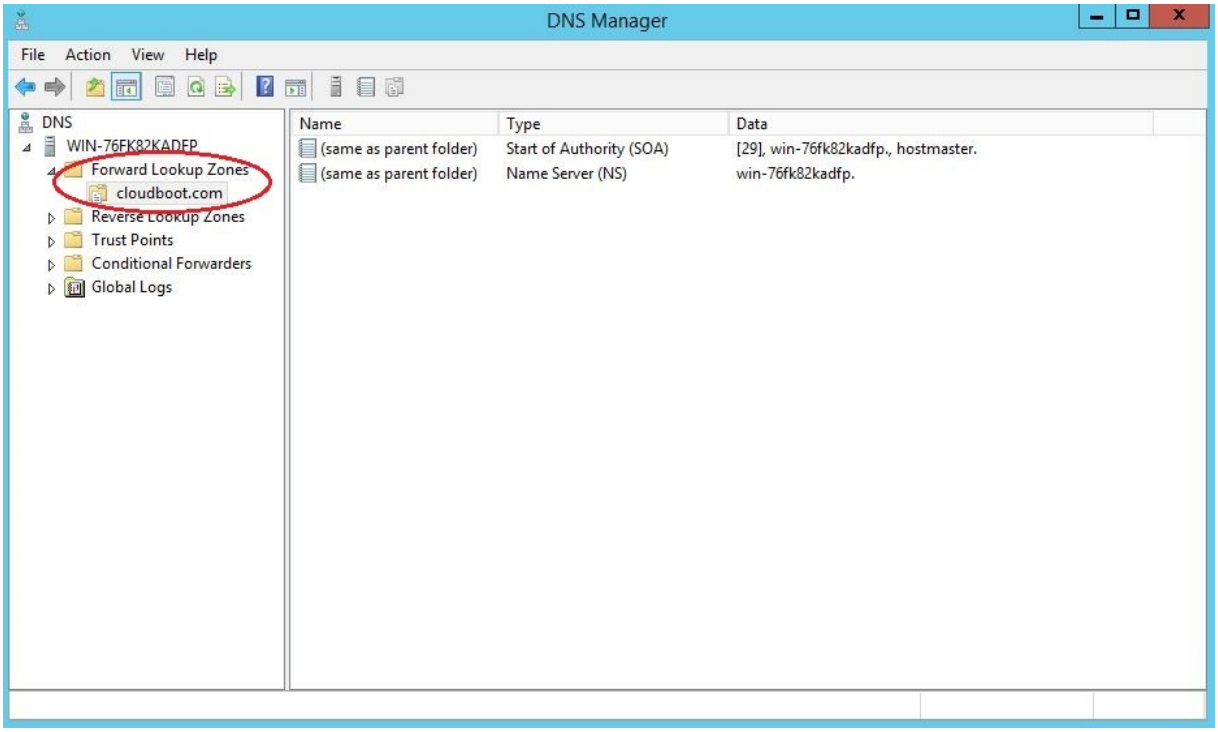

Figure 13 Configure Forward Lookup Zone for IPv6

3. Add a new Host “www” for IPv6 (`2000:bbbb::8`) as shown in Figure 14.

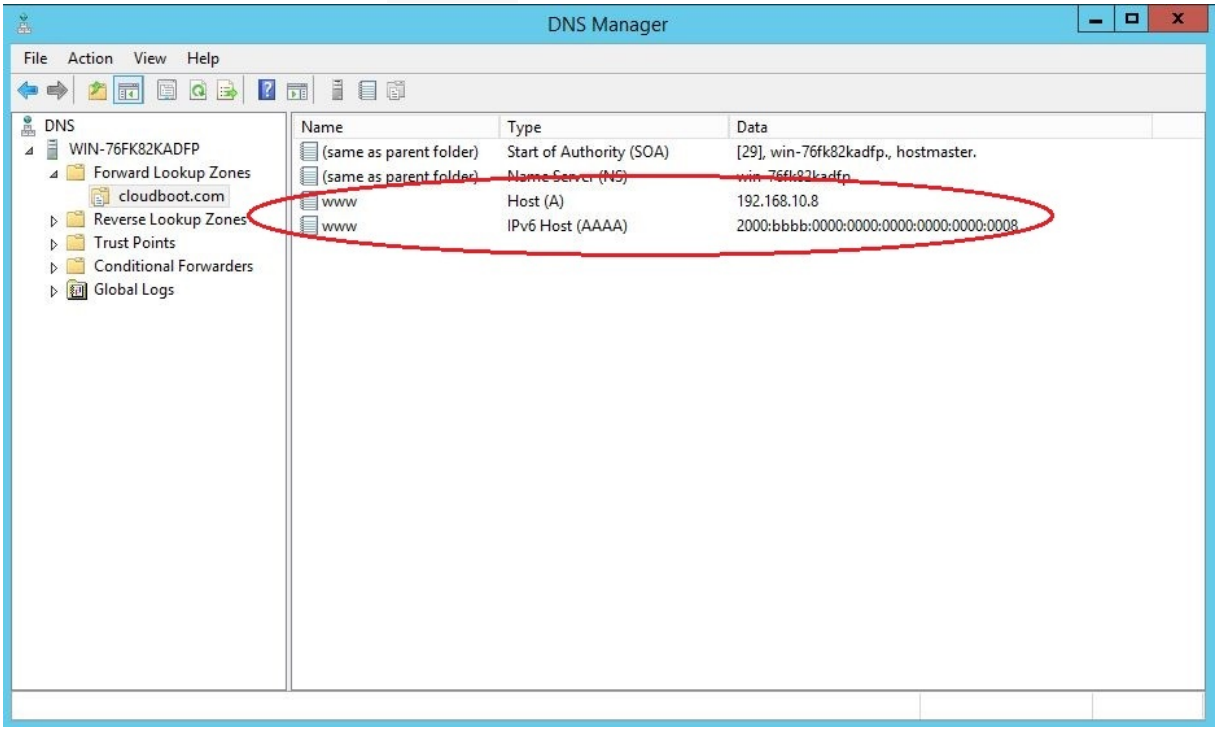

Figure 14 Configure New Host for IPv6

4. Right click the DNS server name and select the ‘All Tasks – Restart’ option to restart the DNSv6 service.

Configure HTTPS Server for IPv6

Please refer to Section Configure HTTPS Server for IPv4 above, as this step is not dependent on IPv4 or IPv6.

Enable NT32 Simulator for IPv6

To enable the UEFI Boot feature for HTTPSv6, the EDKII network stack (IPv6) must be built in your system firmware, which is located at NetworkPkg. Here, the verified version is: 7cf59c854f35c9680965fe83e9cfd863079ddd73.

Modules in DSC file

The following libraries and drivers are required by HTTPSv6 boot:

Add the following libraries to the `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
OpensslLib|CryptoPkg/Library/OpensslLib/OpensslLib.inf
BaseCryptLib|CryptoPkg/Library/BaseCryptLib/BaseCryptLib.inf
TlsLib|CryptoPkg/Library/TlsLib/TlsLib.inf
```

Add the following drivers to the `Component`s section:

```
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
NetworkPkg/TlsDxe/TlsDxe.inf
NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Note: The network controller’s UNDI driver also needs to be in the list of platform files

Modules in FDF file

The following drivers are required in the `FV` section for HTTPSv6 boot:

```
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
INF NetworkPkg/TlsDxe/TlsDxe.inf
INF NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Build the NT32 Simulator

The following command is used to build NT32 using Microsoft Visual Studio* 2013:

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

Run HTTPS Boot

Currently the UEFI HTTPS Boot feature only supports server authentication with an unauthenticated client. To support this mode, the Server CA certificate (`rootcert.pem`) is required by the Client. A private variable is used to configure the CA certificate on the client system. The `EFI_SIGNATURE_LIST` format is used for this variable: `TlsCaCertificate, {0xfd2340D0, 0x3dab, 0x4349, {0xa6, 0xc7, 0x3b, 0x4f, 0x12, 0xb4, 0x8e, 0xae}}`

Configure the Certificate

The server CA certificate must first be configured to enable UEFI HTTPS Boot. The `TlsAuthConfigDxe` driver provides a user interface to support the required certificate configuration. Figure 15 shows the UEFI Client configuration in Boot Manager.

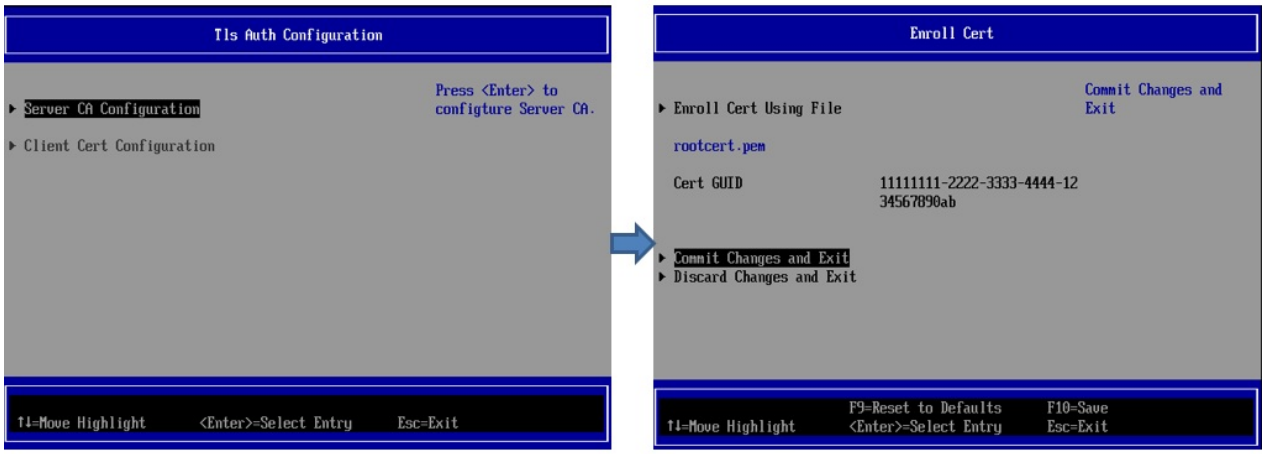

Figure 15: UEFI Client Certificate Configuration

Run HTTPS Boot on the UEFI Client

After the Server CA certificate (`rootcert.pem`) has been configured, the NT32 simulator can perform a HTTPS Boot. Start the NT32 simulator, enter Boot Manager, and select “UEFI HTTPv4” or “UEFI HTTPv6” depending on the server configuration (see Figure 16).

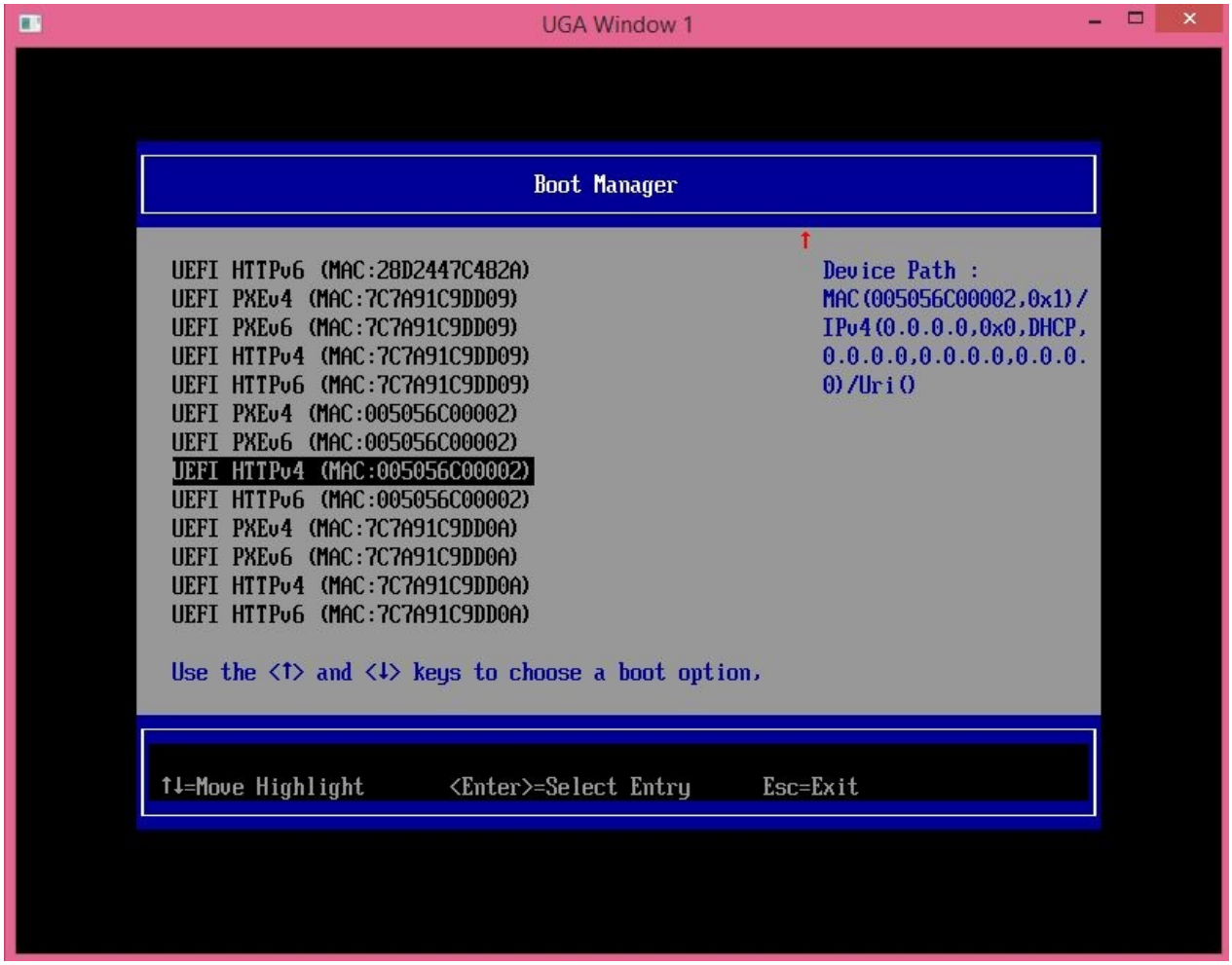

Figure 16: Select Boot Option

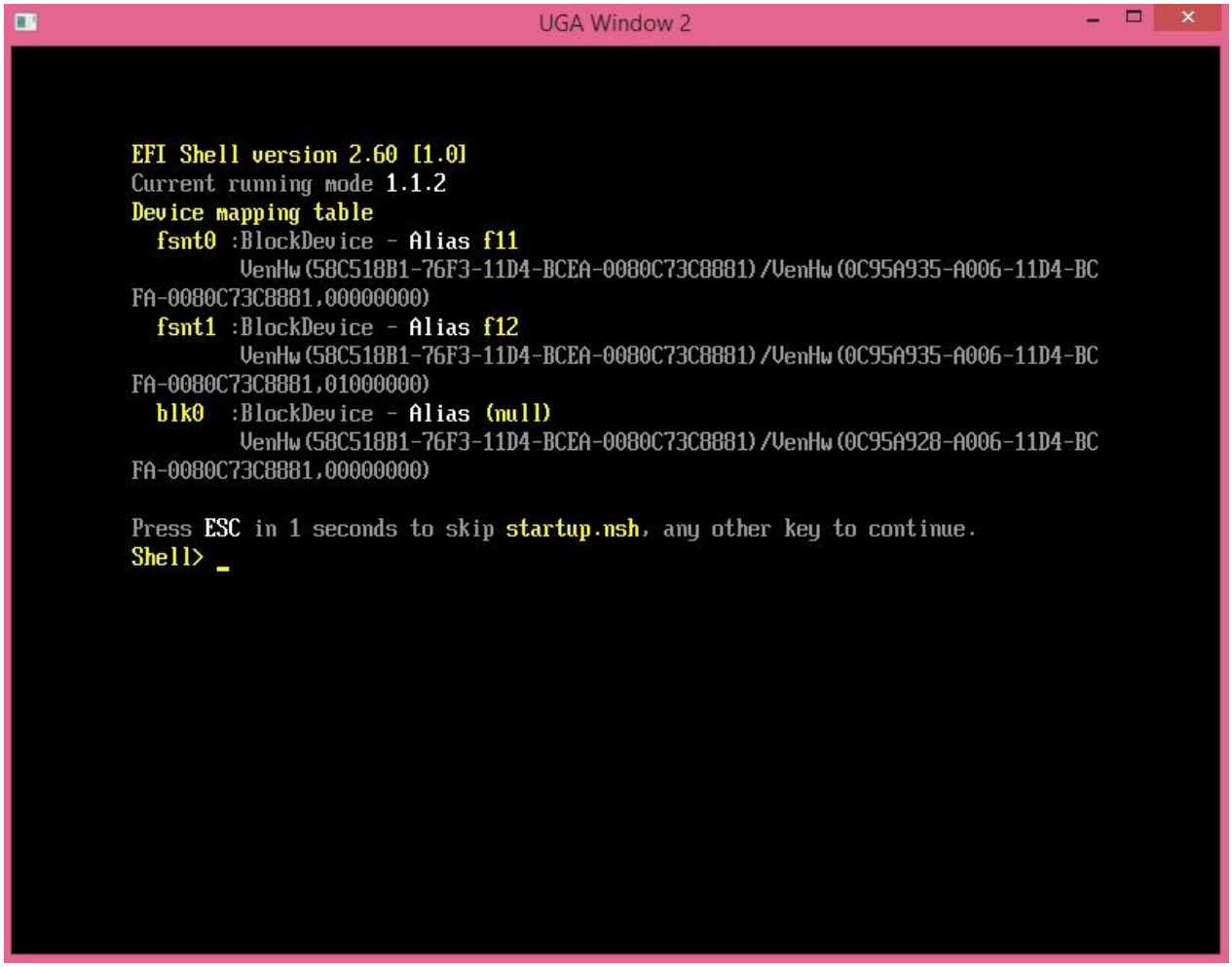

Figure 17: Boot the Downloaded UEFI Shell Image