INI FILE OVERVIEW

The INI file “dhcpsrv.ini” is the configuration and the “database” for the DHCP Server. Its location is in the same directory as the dhcpsrv.exe file. Here are typical examples of the INI file content:

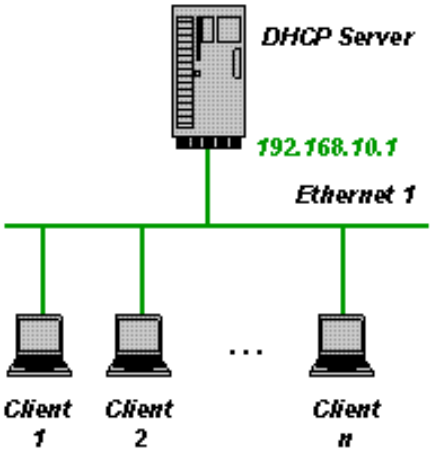
Example 1: The server has only one NIC and manages clients on the same subnet

This is a very simple example and the INI file looks like this:

[General]
SUBNETMASK=255.255.255.0
ROUTER_1=192.168.10.1
DNS_1=192.168.10.1
[Settings]
IPPOOL_1=192.168.10.2-49

This manages the IP addresses 192.168.10.2 until 192.168.10.49. But be aware that all clients connected to the subnet 192.168.10.X get IP

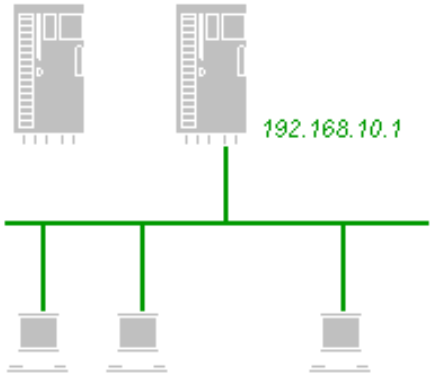
addresses assigned. Please make sure that you don't get in conflict with other DHCP Servers.



Example 2: The server has 2 or more NICs and manages clients on one of the subnets

The INI file looks like this:

[General]
SUBNETMASK=255.255.255.0
ROUTER_1=192.168.10.1
DNS_1=192.168.10.1
[Settings]
IPBIND_1=192.168.10.1
IPPOOL_1=192.168.10.2-49

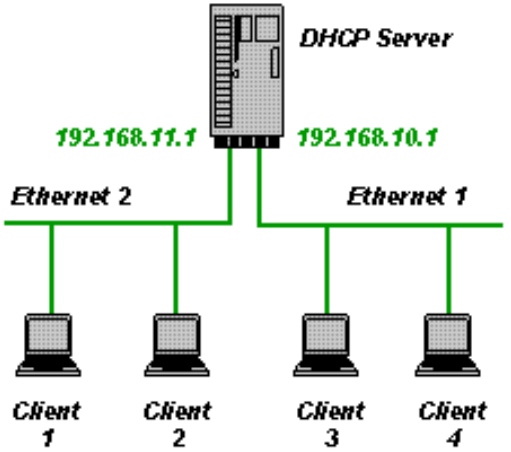


This manages the IP addresses 192.168.10.2 until 192.168.10.49. The DHCP Server binds only to one NIC with the IP address 192.168.10.1. The other NIC (Ethernet 2) is not touched by the DHCP Server. This is very useful if Ethernet 2 is your company LAN and you don't want to interfere with it.

Example 3: The server has 2 or more NICs and manages clients on two subnets

The INI file looks like this:

[General]
SUBNETMASK=255.255.255.0
ROUTER_1=192.168.10.1
DNS_1=192.168.10.1
[Settings]
AssociateBindsToPools=1
IPBIND_1=192.168.10.1
IPBIND_2=192.168.11.1
IPPOOL_1=192.168.10.2-49
IPPOOL_2=192.168.11.2-49



This manages the IP addresses 192.168.10.2 until 192.168.10.49 for Ethernet 1 and 192.168.11.2 until 192.168.11.49 for Ethernet 2. Please note the line AssociateBindsToPools=1. This option associates the IP Pools with the corresponding IP Binds. Without that option the DHCP Server would view all IPPOOL_x as one big IP pool independent of the subnet.

The different INI file sections

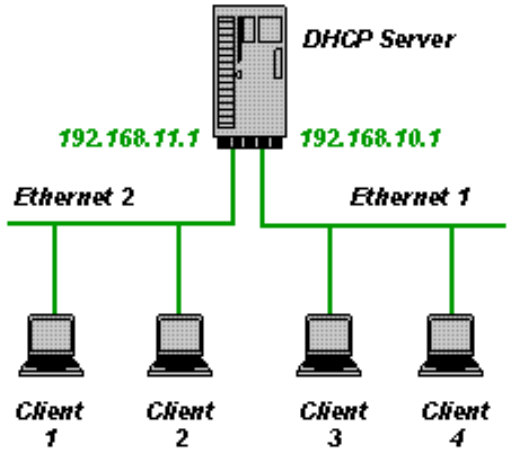
The INI file consists of sections for configuration purposes: [Settings], [DNS-Settings], [Servers] and database purposes: [General], [General_x], and [client section name].
The examples above contained the [Settings] section and the [General] section. The [Settings] section holds all the configuration data for the DHCP server itself. Most importantly the IP pools that the DHCP Server is supposed to manage. The idea behind the database sections is very simple. The DHCP Server tries first to obtain the requested information from the client section then form the [General_x] section and then from the [General] section. The client sections are created as clients connect. For example:

[00-E0-00-1C-AB-67]
IPADDR=192.168.10.11
AutoConfig=12/29/2007 18:37:26
LeaseEnd=1198953446

This section is added automatically by the DHCP Server when a client with the MAC address 00-E0-00-1C-AB-67 requests an IP address. Here, the IP address 192.168.10.11 has been assigned and the IP address will expire at a certain time which is stored as number of seconds since midnight (00:00:00), January 1, 1970. If the client section does not include all requested information, then the DHCP Server checks the [General_x] section and the [General] section.

The complete set of data that the DHCP Server transmits to the client is composed of the entries in the client section [00-E0-00-1C-AB-67], the [General_x] and the [General] section.
Please note that with V1.7 it is possible to use the client-identifier (option 61) instead of the mac address to specify client sections. See also the UseClientID setting defined in the INI file reference. In case of client-id based specifications, all the [00-E0-00-1C-AB-67] sections will actually be defined based on the client supplied option 61. This can be a name (ascii string) or a mac address.
The [General_x] sections are optional but needed to define NIC specific information. Where x is the same number as in IPBIND_x. It is possible to define NIC specific [General_x] sections, in case that the subnets have different subnet masks or different lease times. The following example is an extension of example 3 and defines a lease time of 1 hour for clients connected to Ethernet 1 and a lease time of 1 day for clients connected to Ethernet 2.

[General]
SUBNETMASK=255.255.255.0
ROUTER_1=192.168.10.1
DNS_1=192.168.10.1
[General_1]
LEASETIME=3600 ; default lease time of 1 hour
[General_2]
LEASETIME=86400 ; default lease time of 1 day
[Settings]
AssociateBindsToPools=1
IPBIND_1=192.168.10.1
IPBIND_2=192.168.11.1
IPPOOL_1=192.168.10.2-49
IPPOOL_2=192.168.11.2-49



Lease times are important when a client requests an IP address and all addresses in the IP pool are already assigned to other clients. In that case the client whose lease time has expired least recently is deleted from the INI file and the available IP address is used for the request. The lease time is calculated as the minimum of what the client requests and the LEASETIME configured in the INI file. Clients will request an extension of the lease in time before the lease expires. If the LEASETIME is set to only a few seconds then the network will be constantly full of traffic with lease extension requests.

It is also possible to add client sections to the INI file manually. The following client section assigns the client with the MAC address 00-E0-00-1C-AB-68 the IP address 192.168.10.50.

[00-E0-00-1C-AB-68]
IPADDR=192.168.10.50

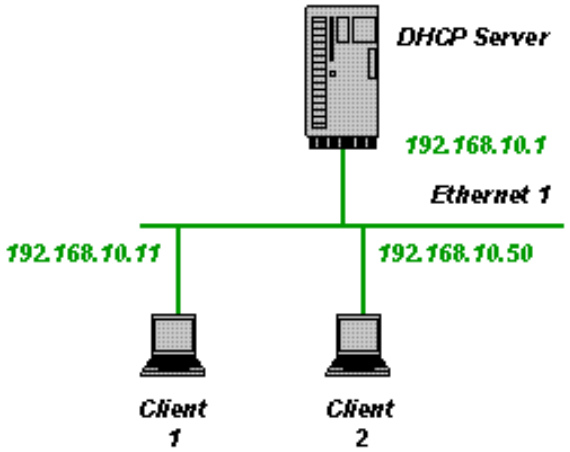
This is a static assignment and is not changed or deleted by the DHCP Server. It is important to not specify the AutoConfig=xxxxx entry in the client section, otherwise the DHCP Server would consider this as an automatically created client section. The INI file may contain static

entries only. This can be a useful way of limiting the clients on the network to only known and allowed computers. The IPPOOL_x should be removed from the INI file in case of fully static assignments. Here is an example for fully static configurations:

Example 4: Static configuration on one subnet

The INI file looks like this:

[General]
SUBNETMASK=255.255.255.0
LEASETIME=32000000 ; approx. 1 year
ROUTER_1=192.168.10.1
DNS_1=192.168.10.1
[Settings]
IPBIND_1=192.168.10.1
IgnoreUnknownClients=1
[00-E0-00-1C-AB-68] ; Client 2
IPADDR=192.168.10.50
[00-E0-00-1C-AB-67] ; Client 1
IPADDR=192.168.10.11



Please note the line IgnoreUnknownClients=1. This option prevents the DHCP Server from sending answers to clients that are not configured in the INI file.

DNS Server configuration

The [DNS-Settings] section consists of the settings for the integrated DNS server. This is a new feature of V1.8 and lets the DHCP Server on the same data (ini file) also handle DNS requests. This is done based on the IPADDR and Hostname entries of the configured clients. The supported DNS requests are A (IP->hostname) and PTR (hostname -> IP). If a client section in the ini file does not include the Hostname entry, then this client is not visible to the DNS Server (it has no name !). Example for a DNS configuration:

[DNS-Settings]
EnableDNS=1 ; if not set or set to 0, then DNS is not enabled
FORWARD=192.168.2.1 ; if set, then all requests that can not be fulfilled are forwarded to this address
DEFAULTIPADDR=192.168.10.1 ; if set, then all requests from not configured clients get this IP Address as an answer

[General]
DOMAINNAME=mydomain.local

...

Please note the DOMAINNAME in the example above. This really has to be stored in the [General] section (not [General_x]) and specifies the domain name suffix for the DNS requests. If a client is configured with hostname=linux_box_1, then linux_box_1 and linux_box_1. mydomain.local are both valid names for the client machine. EnableDNS is self explenatory, I guess.

The FORWARD setting defines one external DNS server that gets all the requests that the integrated DNS server can not answer. This feature is fairly rudimentary and works only with UDP.

The DEFAULTIPADDR setting is a very simple but effective security feature. If DEFAULTIPADDR is not set then the DNS server serves everybody as expected with resovled names. If it is set, then requesters with an unknown IP address get always the default ip address as an answer to all name resolution requests. The intention of this feature is to forward all unknown clients to a predefined default address (e.g. a web server with registration facility).

TFTP Server configuration

The [TFTP-SETTINGS] section consists of the settings for the integrated TFTP Server. This is a new feature of V1.9 and lets the DHCP server act as a tftp server. This is useful in network boot scenarios in combination with the BOOTP protocol. Please see the following ini file snippet as an example for a tftp configuration.

[TFTP-Settings]
EnableTFTP=1 ; if not set or set to 0, then TFTP is not enabled
PortRange=AUTO ; the ports used for TFTP are automatically chosen
WritePermission=0 ; if not set or set to 0, then write operations are denied
Root="c:\tftproot\" ; the server side pathname to the files served through the tftp protocol

...

No further configuration is required in the ini file. Please make sure that port 69 (tftp protocol port) is available through the firewalls.

HTTP Server configuration

The [HTTP-SETTINGS] section consists of the settings for the integrated HTTP Server. This is a new feature of V2.0 and lets the DHCP server act as a Web server. The main purpose of the Web server is to support life diagnostic information shown in a Web browser, but it can be used as a regular Web server as well (does not support CGI). Please see the following ini file snippet as an example for a http configuration.

```
[Settings]
PORT_80=50556 ; overrides the default port 80 to prevent collisions with other web servers

[HTTP-Settings]
EnableHTTP=1 ; if not set or set to 0, then HTTP is not enabled
Root="c:\httproot\" ; the server side pathname to the files served through the http protocol
Logfile="c:\dhcprv\httplog.txt" ; the http server logs all web access in this file

[content-type] ; mapping of filename extensions to content-types
.htm=text/html
.html=text/html
.css=text/css
.xsl=text/xsl
.jpg=image/jpeg
.png=image/png
.gif=image/gif
.ico=image/x-icon
.xml=application/xml
.txt=text/plain
```

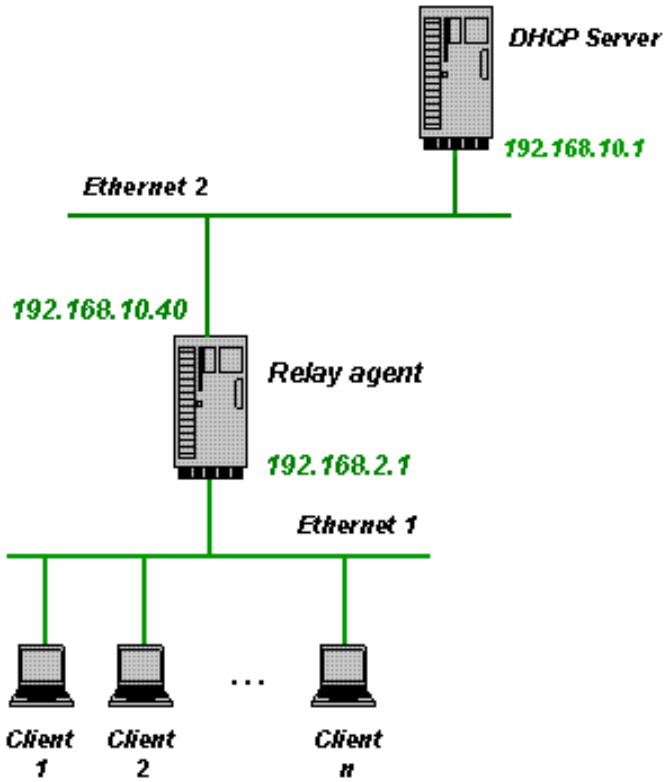
The http server will serve all files (including sub directories) in c:\httproot. In addition to this two special "files" are supported and can be reached from the web browser: dhcpstatus.xml and dhcptrace.txt. These files do not exist in the http root folder but its content is generated on the fly by the DHCP Server. Please note that dhcpstatus.xml makes use of xsl translations to transform xml to html. The file dhcpstyle.xsl is part of the zip file and needs to be stored in the http root folder.
The DHCP status web page can be accessed though the following URL: http://127.0.0.1:50556/dhcpstatus.xml.

Relay agent

The relay agent support was added in V2.0. This includes the actual relay agent and the support of multiple scopes on the DHCP server side. A relay agent is needed if the clients to be served with IP addresses are in a different subnet than the DHCP server. This is almost always the case when a central DHCP Server is responsible for many different subnets. Instead of having as many DHCP servers as there are subnets, a relay agent per subset is used. To setup the dhcpsrv.exe to act as a relay agent requires the following INI file:

```
[Settings]
IPBIND_0=192.168.2.1 ; local IP address on client subnet
IPRELAY_0=192.168.10.1 ; IP address of central DHCP server
IPBIND_1=192.168.10.40 ; local IP address on server subnet
IPRELAY_1=192.168.10.1 ; IP address of central DHCP server
AssociateBindsToPools=1
```

All DHCP requests coming in from the client subnet will be forwarded to the DHCP server 192.168.10.1 and back to the client in return. The DHCP server itself needs to be configured with the following INI file:




```
[Settings]
IPBIND_0=192.168.10.1
IPPOOL_0=192.168.10.40-59 ; IP pool for directly connected clients
IPBIND_1=192.168.10.1
IPPOOL_1=192.168.2.2-254 ; IP pool for clients behind relay agent
AssociateBindsToPools=1

[General_0]
SUBNETMASK=255.255.255.0 ; subnet mask for directly connected clients
[General_1]
SUBNETMASK=255.255.255.0 ; subnet mask for clients behind relay agent
```

The DHCP server automatically matches the scope according to the network id of the client subnet. A total of 10 scopes (IPPOOL_0 to IPPOOL_9) is supported. Please note that the local scope needs to be the first defined. Easiest is to always make the local scope IPPOOL_0.

In both cases, relay agent and dhcp server, the IPBIND_x always defines the local IP address of the network interface card for receiving request and sending responses. Local scopes have the same network ID as the IPBIND_x and define their respective pool in IPPOOL_x. If the scopes have different subnet masks, then use the [General_x] sections for that.

The relay agent function is realized in conformance to RFC5107: DHCP server identifier override, RFC3046: DHCP Relay Agent Information Option, RFC3527: Link selection sub-option for the Relay Agent.

Multiple Server configurations

The [Servers] section is a new functionality in V1.8 and allows to virtualize the DHCP Server. It was already a suggested procedure to run the DHCP Server executable multiple times with seperate ini files, if more than one subnet with multiple IPPOOLs each need to be supported. This was not very user friendly and didn't work for service installations. With the [Servers] settings this is now an integrated functionality. Here is an example on how the dhcpsrv.ini file looks like in this case:

```
[Settings]
InstallAsService=2 ; run as application
ShowBalloonMessages=1
NotifyTimeout=10000 ; delay after network changes detected in milliseconds

[Servers]
SERVER_1=dhcp1.ini
SERVER_2=dhcp2.ini
```

Two "virtual" DHCP Server instances are created with their own ini-file. All the sections and configurations can be specified in the dhcp1.ini or dhcp2.ini as usual. The settings and databases are kept seperately and behave as if two instances of the dhcpsr.exe would have been started with a dedicated ini file for each of them. The only [Settings] that are not "virtualized" are the ones shown in the example above: InstallAsService, ShowBalloonMessages and NotifyTimeout. They can not be "virtualized", because they really define the behavior of the executable process (e.g. installation or icon in the tray).
Please be aware that all IP addresses specified in the examples above need to be adopted to your particular network setup. See the complete INI file reference for all other configuration options here: INI file reference .

Mulitple Scopes (available since V2.3)

What are scopes?

Scopes are a way to define what information is passed from the DHCP server to the client. Not all clients are the same. There may be mobile devices, test equipment, IP phones, guest laptops, you name it. Sometimes it is necessery to use in example a different range of IP address depending on what type of device it is. This can be achieved with the DHCP Server for Windows with the following INI file:

```
[Settings]
AssociateBindsToPools=1
Trace=1
IPBIND_0=192.168.5.1
IPPOOL_0=192.168.5.2-49
IPSCOPE_0=if(vendorclass=="PHONE-VENDOR","Phones", null) ;

IPBIND_1=192.168.5.1
IPPOOL_1=192.168.5.50-99

[General]
SUBNETMASK=255.255.255.0
```

With that INI file all devices with a vendorclass "PHONE-VENDOR" get an IP address from the range 192.168.5.2 to 192.168.5.49.

All other devices get an IP address out of the range from 192.168.5.50 to 192.168.5.99.

The DHCP server checks all scopes starting with IPSCOPE_0 to IPSCOPE_9 if it matches to the client request. The first match is picked and the corresponding IPPOOL_n is used as the IP address range. The key to all this is obviously the IPSCOPE_n setting which was introduced with V2.3 of the DHCP server.

IPSCOPE_n syntax

The definition of IPSCOPE_n is like it's own little programming language. The result of the expression formulated in that language is a string. If that string results to a null value, then that particular scope does not match to the client request (e.g. wrong vendor class). If the resulting string is not null then we have a match.

The following is the definition of the IPSCOPE_n syntax:

```
string-expression = simple-string-expression
     | if ( logical-expression, string-expression, string-expression )
     | concat ( string-expression { , string-expression } )
     | substring ( string-expression , start-offset, length )
     | firstvalue( string-expression { , string-expression } )
     ;
```

if	With if(expr, then, else) you can define the resulting string based on the logical-expression result. If the logical-expression is true then the first string-expression (then) is returned, otherwise the second string-expression (else) is returned.
concat	The result of concat(string1, ..., stringN) is the concatanation of all strings. Example: concat("prefix_", "string") rersults to "prefix_string".
substring	The result of substring(string, start, len) is a sub-string of len characters starting with the character at offset start. Example: substring("hello tom", 5, 3) is "tom". If the input string does not contain enough characters, then the result is cut accordingly. The result of substring("hello tom", 5, 10) is still "tom".
firstvalue	firstvalue is actually a convinience function. It shortens if/then/else chains such as: if (vendorclass!=null, vendorclass, if(userclass!=null, userclass, null)) This returns vendorclass or userclass if it is defined (not null) and null if both are not defined. This can be expressed very easy with firstvalue: firstvalue(vendorclass, userclass, null) firstvalue basically results in the first parameter that is not null.

```
simple-string-expression = "simple string in quotes"
     | vendorclass
```

```
      | userclass
      | hostname
      | chaddr
      | ciaddr
      | null
      | OPTION_nnn
      ;
```

vendorclass Returns the dhcp option 60 string from the current dhcp request.

userclass Returns the dhcp option 77 string from the current dhcp request.

hostname Returns the dhcp option 12 string from the current dhcp request.

chaddr Returns the mac address as it is supplied in the chaddr field of the current dhcp request header as a string. Format of string is "00:01:02:03:04:05". Including the colons (:) without the quotes (").

ciaddr Returns the ip address as it is supplied in the ciaddr field of the current dhcp request header as a string. Format of string is "123.456.78.9". Including the colons (:) without the quotes (").

OPTION_nnn Returns the content of the dhcp option nnn as a string. No conversion of any kind is performed. nnn is a decimal number with no leading 0's. Example: substring(option_61,1,9) results into a string of (max) 9 characters starting at offset 1 of the dhcp option 61. Option 61 is the client identifier. Client identifier uses the first character as a tag value, therefore the string starts at offset 1.

```
logical-expression = bool-expression
     | bool-expression || logical-expression
     | bool-expression && logical-expression
     ;
```

expr1 || expr2 The result is false, if both expressions are false. It's true otherwise.

expr1 && expr2 The result is true, if both expressions are true. It's false otherwise.

```
bool-expression = simple-bool-expression
     | ! simple-bool-expression
```

```
      | string-expression == string-expression
      | string-expression != string-expression
      | string-expression ~= string-expression
      | string-expression ~~ string-expression
      ;
```

string1 == string2	The result is true, if both strings are identical. The comparison is case sensitive.
string1 != string2	The result is true, if both strings are not identical. The comparison is case sensitive.
string ~= templ	The result is true, if the string matches the template. The template can include DOS-like wildcard characters (?, *). The comparison is case sensitive.
string ~~ templ	The result is true, if the string matches the template. The template can include DOS-like wildcard characters (?, *). The comparison is case insensitive. Example: if (substring(option_61,1,30) ~~ "PXE*", "pxe-client", null). The result of this is "pxe-client", if the client identifier starts with "PXE", otherwise null.

```
simple-bool-expression = true
      | false
      | ( logical-expression )
      ;
```

The complete language is case insensitive. So all examples and definitions above work lower case as well as in upper case letters.

Scope sections

The result of the IPSCOPE_n string-expression is a string. This string determines whether or not this particular scope matches to the client request or not. If the string-expression results to a null string, then the scope does not match. If the string is not null, then the scope does not only match, the resulting string also defines the scope section.

The order in which the DHCP result options are compiled into the DHCP response is:

1. Client section [00-01-02-03-04-05] or [client-id]
2. Scope section [result of IPSCOPE_n]
3. [General_n] section
4. [General] section

Here is an example:

```
[Settings]
AssociateBindsToPools=1
Trace=1
IPBIND_0=192.168.5.1
IPPOOL_0=192.168.5.2-49
IPSCOPE_0=if(vendorclass=="PHONE-VENDOR","Phones", null) ;

IPBIND_1=192.168.5.1
IPPOOL_1=192.168.5.50-99

[General]
SUBNETMASK=255.255.255.0

[Phones]
OPTION_66=192.168.5.1 ;
```

IPSCOPE_0 is null for everything but phones from vendor "PHONE-VENDOR". Phones have a scope section called "Phones". All other devices do not get the options returned that are defined in the [Phones] scope section.

You are welcome to E-Mail me if you have any questions or suggestions.

INI FILE REFERENCE

Entries in the [Settings] section

[Settings]
AssociateBindsToPools=1

Default:0

Associate the IP pools with the respective IP binds. If the DHCP Server is configured to use more than one NIC with a separate subnet each (see Example 3 in the overview) then this entry associates the IPPOOL_x with the IPBIND_x. The two entries with the same number “x” belong together. Clients on subnet IPBIND_x get IP addresses assigned only from the pool defined by IPPOOL_x.

The default behavior (AssociateBindsToPools=0) is to view all IPPOOL_x as one big pool of IP addresses which can be assigned to any client regardless of the subnet IPBIND_x.

[Settings]
ConfigureUnknownClients=1

Default: 1

ConfigureUnknownClients is available since V1.7. Setting this to 1 tells the DHCP server to automatically configure unknown clients. Since this is the default, there is no change to previous versions. If ConfigureUnknownClients is set to 0, then a client is assigned an IP address only if the client is already specified in the INI file. Note: Important difference to IgnoreUnknownClients. IgnoreUnknownClients only controls how the DHCP Server behaves when no IP address is assigned (declining request or keep quiet). What makes a client a "known client" is an existing entry in the INI file such as:
[00-01-02-03-04-05] ; mac address
AutoConfig=1

or

[00-01-02-03-04-??] ; mac address with wildcards
AutoConfig=1

or

[client-identifier] ; client identifier (option 61)
AutoConfig=1

[Settings]
Database=d:\database.ini

Default: -

The Database setting allows to split the dhcpsrv.ini file into two parts. (Available since V1.7) Part 1 is for everything in the [Settings] section. This stays in the dhcpsrv.ini file. All the rest [General], [General_x] or client section goes into the Database.ini file. This allows a clean distinction between real settings and configuration for the DHCP server and the IP assignment database in which the client configurations are managed.
New in V1.9.2: If no [General] or [General_x] section is found in the Database.ini file, then the dhcpsrv.ini config file is used for that instead.
If no Database setting is defined, then the DHCP server assumes that the database is part of the dhcpsrv.ini file.

[Settings]
DeleteOnRelease=0

Default: 1

The DeleteOnRelease setting is new in V1.9. It defines whether a client entry in the databse is deleted upon DHCPRELEASE, or not. The default behavior is to automatically delete the entry in the database. The effect is that when the same client comes back after some time and asks for an IP address, no information about it's previous IP address is available. Even if the old IP address is available, it would most likely not being chosen. With DeleteOnRelease set to 0, the entry only gets an expired lease timestamp and can therefore be reactivated at a later time.

In addition to this the DeleteOnRelease setting also controls the bahvior for expired leases. When DeleteOnRelease is set then expired leases are also automatically deleted from the INI file same as DHCPRELEASE from client. This is new in V1.9.3 and is based on a timeout mechansim. The timeout is set to the time when the next client lease expires. This is updated every time a DHCP action is happening.

[Settings]
EnableSendRawUnicast=0

Default: 0

The EnableSendRawUnicast setting is new in V1.9.1. The DHCP protocol allows the client to specify a broadcast flag. If this flag is not set, then the DHCP server is asked to unicast the response. Unfortunatly, there is no standard mechanism based on the winsock programming interface that would allow the DHCP server to send unicasts solely based on mac address addressing scheme. In V1.9.1 the usage of the winpcap library got implemented. In situations where a unicast is required without having an IP address of the client at hand, the wpcap.dll function pcap_sendpacket is used. The EnableSendRawUnicast setting enables this functionality. Please set to 1 only if there are really problems with clients in that situation. Clients actually should be also fine with receiving responses as broadcast, which is the default behavior if EnableSendRawUnicast is not set or the wpcap dll is not installed. This has been tested with winpcap Version V4.1.1. Please also see the description of the OverwriteBroadcastFlag setting for even more advanced options to set.

[Settings]
ExpiredLeaseTimeout

Default: 4294967 (approx. 49 days)

ExpiredLeaseTimeout defines a time value in seconds. The mechanism that checks expired leases is triggered by this. Regardless of when the next lease expires, the mechanism is scheduled at least in the time interval specified by ExpiredLeaseTimeout. This is useful when the INI file entries get changed manually and the DHCP server may not be aware of already expired leases. By this, the INI file gets cleaned up cyclically.

[Settings]
HBA

Default: -

HBA provides a basic load balancing concept according to RFC3074. HBA stands for hash bucket allocation and defines whether the DHCP Server is responsible for a client or not. The HBA consists of 32 hex values separated by a single blank. Every hex number represents eight bits, one bit per hash bucket. The DHCP server calculates a hash value for every client that asks for an IP address. The hash value is between 0 and 255. If the correspondig HBA bit for that hash value is 1, then the client is serviced. If it is 0 then the client is not serviced. Please assume the following example:

[Settings]
HBA=FF FF FF FF FF FF 00 00 FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

This restricts the DHCP Server to service clients whos hash value is 0 through 47 or 64 through 127. The server servicing the rest of the clients should have the exact opposite HBA. The HBA logic kicks in only for clients that are unknown (not yet in the INI file) to the DHCP Server. Clients who are already known are serviced regardless of the HBA.

[Settings]
IgnoreUnknownClients=1

Default: 0

Setting this to 1 tells the DHCP server not to answer to requests from clients that are not configured in the client sections. DHCP requests from unknown clients are declined, if IgnoreUnknownClients is set to 0 (default).

[Settings]
InstallAsService=2

This is used internally by the DHCP Server to remember the checkbox “Don’t ask this question again” on startup of the software.

[Settings]
IPBIND_0=192.168.2.1
…
IPBIND_9=192.168.10.2

Default: -

Binding to certain IP addresses (or NIC cards).

[Settings]
IPBIND_0=123.45.56.78
IPBIND_1="Local Area Connection" ; name of connection

This restricts the DHCP server to bind only to 123.45.56.78. All other IP addresses are ignored. This is very important if you want to restrict the DHCP server to particular cards in your computer. You have already two IP addresses, as soon as the computer is on the internet using a dial up connection. The DHCP server will bind to all IP addresses if no IPBIND_x is found. IPBIND_0 to IPBIND_9 are supported.
Starting with V1.8 the value of an IPBIND_x entry can also be specified as a LAN connection name. These are the names Windows is using to specifiy network connections. In case a name based

IPBIND_x is configured, then subnet mask, ip address and all other windows configured data such as DNS server addresses etc. are obtained automatically and don't need to be explicitly specified in the ini file anymore. See also the AUTO feature in IPPOOL_x.

[Settings]
IPPOOL_0=192.168.2.2-49
…
IPPOOL_9=192.168.10.3-12

Default: -

From version 1.5 on the so called auto configuration is supported:
[Settings]
IPPOOL_0=123.45.56.78-90
IPPOOL_1=123.45.57.0-254
IPPOOL_2=AUTO

The above enables three IP pools for DHCP clients. Unknown clients are automatically configured by adding the appropriate client sections to the INI file. IP pools of size 1 (e.g. 123.45.56.78-78) are allowed and are very useful if you want to assign the same IP address to all clients.
The new feature in V1.8 is: AUTO. If AUTO instead of a numerical IP-Pool configuration is used then the IP Pool is automatically chosen to cover the complete IP Range based on the IPBIND_x IP address and the network mask. This works best together with a name based IPBIND_x configuration.

From V1.9 on the IPPOOL_x numerical IP-Pool definition is enhanced with a comma syntax. This allows to define IP pools such as:
[Settings]
IPPOOL_0=123.45.56.78-90,92,94

This adds 123.45.56.92 and 123.45.56.94 to the IPPOOL_0.

Since V2.3.1 it is also possible to define a class A/B network by using IPPOOL_1=10.45.3.1-10.45.5.254. This defines, in accordance with the subnet mask, all IP addresses from 10.45.3.1 through 10.45..5.254 as the IP pool.

[Settings]
IPRELAY_0=192.168.2.1
…
IPRELAY_9=192.168.10.1

Default: -

IPRELAY_x is the definition of the DHCP Server address where the relay agent sends all requests coming in from IPBIND_x. Please see the description in the INI file overview about relay agents to learn more. IPRELAY_x is supported since V2.0.

[Settings]
IPSCOPE_0=*string-expression*
...
IPSCOPE_9=*string-expression*

Default: -

IPSCOPE_n defines based on the client request if this scope matches or not. Please see the description in the INI file overview about mulitple scopes to learn more. IPSCOPE_n is supported since V2.3.

[Settings]
MINPACKETLENGTH=300

Default: 300

This option is new in V1.6 and allows to specify a minimum packet length that the DHCP server is supposed to respond with. Default is 300 bytes. Some clients were not happy with a too short response. The remainder of the packet is filled with 0.

[Settings]
MINPACKETLENGTH=300

Default: 300

[Settings]
NAKMessage="Info text for DHCPNAK" ;

Default: -

Since V2.2.1 NAKMessage allows to specify an info text that is added to DHCPNAK replies as option 56. This info text is visible in some DHCP clients. It is very rarely used and should usually be omitted. Default behavior if NAKMessage is not specified is to not add option 56 in the reply.

[Settings]
NotifyTimeout=5000

Default: 1000

The DHCP Server automatically recognizes changes in the IP configuration in Windows. This is done based on the Windows API NotifyAddrChange. This API tells the DHCP Server when an IP configuration is changing. The DHCP Server acts on this by internally shutting down and restarting itself. Upon restart, the new configuration is used. In order to not shutdown and restart unneccessarily the DHCP Server uses a timeout mechanism to delay the shutdown after NotifyAddrChange events. The default of this timeout is 1 second (1000 millisends). Some network cards are very slow and require a longer delay before they are actually ready to run. Set the NotifyTimeout value to a larger timeout period to compensate for this.

[Settings]
OverwriteBroadcastFlag=0

Default: broadcast-flag as set by the client request

OverwriteBroadcastFlag is new in V1.9.1. It allows to overwrite the broadcast flag set by the client. (See also EnableSendRawUnicast setting). The default behavior if this setting is not present or commented out, is to act according to the DHCP specification. If OverwriteBroadcastFlag is present then the behavior is to always broadcast the response (= 0) or to always try to unicast the response first (= 1). If unicast fails, then the DHCP Server defaults to broadcast.

[Settings]
PADDING=0

Default: 0

PADDING is new in V1.8 and enables to add PAD bytes (0) after each DHCP option, if the next option would otherwise be stored at a non WORD aligned offset. Some older BOOTP clients require this.

[Settings]
PacketValidation=0

Default: 1

PacketValidation allows to switch off the packet validation routine that filters malformed DHCP requests. Added in V2.2. This setting exists only to bypass a to aggressive validation that falsly identifies correct requests as malformed.

[Settings]
PORT_67=67

Default: 67

PORT_68=68

Default: 68

PORT_53=53

Default: 53

PORT_69=69

Default: 69

This options are new in V1.6.5 and allows to customize the IP ports the DHCP Server is supposed to use. Default is 67 or 68 respectively. Port 53 is the default port for the DNS Server. Port 69 is the default port of the TFTP Server. Port 80 is the default port of the HTTP Server (available since V2.0). If no relay agent function is in use, then this setting has no effect.

PORT_80=80

Default: 80

[Settings]
RelayAgentAdvanced=1

Default: 1

Since V2.1.3 this setting allows to alter the behavior of the relay agent. With RelayAgentAdvanced=0, the relay agent runs in a compatible mode and can be used in conjunction with other DHCP servers such as Windows 2003 Server. With the default setting of RelayAgentAdvanced=1, the relay agent adds option 82 to all requests. This allows the DHCP Server to distinguish between the link selection information and the address of the relay agent.

[Settings]
ShowBalloonMessages=1

Default: 1

Since V1.7 the DHCP server shows balloon messages in the tray if run in application mode. This can be disabled by setting ShowBalloonMessages=0.

[Settings]
SERVICENAME=DHCP Server
SERVICEDISPLAYNAME=DHCP Server

Default: DHCP Server

Since V1.9 the DHCP server supports customizing the service name entries. This is useful when the service is to be installed several times (different INI file needed) or in combination with an already existing DHCP server with the same serivce name.

[Settings]
SingleInstanceApp=1

Default: 0

Since V2.2.3 it is possible to limit the DHCP Server to one instance if run in application mode. If SingleInstanceApp is not configured or set to 0 then the server can be started multiple times. If set to 1 then the second instance will only show a dialog inidicating that it's a second instance and will terminate after confirming the dialog box.

[Settings]
TRAYICON=someicon.ico

Default: Default

Since V2.1 it is possible to change the tray icon of the dhcp server. The default tray icon is still avaible as a default. You can either choose "None", "Default" or a filename.
TRAYICON=Default uses the built in default icon.
TRAYICON=None uses no icon at all. This is not recommended because the process can only be stopped using task manager if there is no tray icon.
TRAYICON=filname.ico loads the given icon file as the try icon.

All this is of course only effective if dhcpsrv.exe is started as an application (not as a service). The intended purposes to choose the tray icon is to bo able to distinguish the processes in the tray in case you are running dhcpsrv.exe multiple times.

[Settings]
Trace=1

Default: 0

[Settings]
TraceFile=c:\temp\dhcptrc.txt

You can switch the trace on by putting a Trace=1 entry into the [Settings] section (not the general section as I stated in former documentation). The trace file is called dhcptrc.txt and is written into the same folder as the dhcpsrv.exe and the dhcpsrv.ini files. I recommend to switch the trace on, if you encounter any problems and attach the trace file to the e-mail that you are sending to me to report it. If you want the trace to have a different name (and/or directory) then use TraceFile for that. Example:
[Settings]
Trace=1 ; 1= enable, 0=disable
TraceFile=c:\temp\dhcptrc.txt ; this is where the trace goes
If the TraceFile setting in not set then the default is dhcptrc.txt in the dhcpsrv.exe directory. TraceFile is a new feature in V1.5.2.

[Settings]
UseClientID=1

Default: 0

If UseClientID is set to 1, then clients can be recognized based on their Option 61 specification "client-identifier". Option 61 allows several client id formats. Two of them are supported by the DHCP Server: type=0 and type=1. Type 1 is treated as the mac address of the client and just uses the mac address given by option 61 instead of the chaddr field of the DHCP packet.
Type 0 is treated as an ascii string. The client section is created with the client identifier instead of the mac address. Also the manual specification of client sections need to be based on the client-idendifier, if UseClientID is set to 1. This option is new in V1.7.

[Settings]
VENDORCLASS=MSFT 5.0

Default: -

If a VENDORCLASS is specified as an entry in the [Settings] section, then only clients with the same vendor class defined in their DHCP request get an IP address assigned.

Entries in the [DNS-Settings] section

[DNS-Settings] **EnableDNS=1** Default:0	EnableDNS enables the integrated Domain Name Service (DNS) functionality of the DHCP Server.
[DNS-Settings] **FORWARD=192.168.2.1** Default:none	The FORWARD setting defines the IP address of one external DNS server that gets all the requests that the integrated DNS server can not answer. This feature is fairly rudimentary and works only with UDP.
[DNS-Settings] **DEFAULTIPADDR=192.168.17.18** Default:none	The DEFAULTIPADDR setting is a very simple but effective security feature. If DEFAULTIPADDR is not set then the DNS server serves everybody as expected with resovled names. If it is set, then requesters with an unknown IP address get always the default ip address as an answer to all name resolution requests. The intention of this feature is to forward all unknown clients to a predefined default address (e.g. a web server with registration facility). Since V1.9.3, this can be also defined on a per client basis. Please see AllowDNSQuery for details.

Entries in the [TFTP-Settings] section

[TFTP-Settings] **EnableTFTP=1** Default:0	EnableTFTP enables the integrated Trivial File Transfer Protocol (TFTP) server functionality of the DHCP Server.
[TFTP-Settings] **MaxBlockSize=1024** Default:-	The tftp protocol allows to negotiate the blksize. This is the number of bytes transfered between client and server in each packet. It has been shown that if the client asks for a blksize that is bigger than the MTU, that the tftp communication does not work. Therefore it is since V2.2.3 possible to limit the blksize on the server side. MaxBlockSize should be set to a value between 512 and the MTU.
[TFTP-Settings] **PortRange=51000-51100** Default:AUTO	The PortRange setting defines the ports the TFTP server will use for sending and receiving data. The default is AUTO and specifies that the port numbers are allocated automatically on demand as they are made available by windows sockets (winsock). This can be a problem with firewalls, because the port numbers can not be explicitly opened in the firewall if they are unknown at configuration time. Therefore it is possible to specify a range of ports like PortRange=51000-51100. This tells the TFTP function to select port numbers out of the given range for sending and receiving data. Please make sure that the specified port numbers are not confilicting with other network services running on the server.

[TFTP-Settings] **WritePermission=1** Default:0	If not set or set to 0 WritePermission is not granted. Tftp clients can not write files in this case.

[TFTP-Settings] **Root=c:\tftproot** Default:none	Only files located in or under the given Root path are served by the tftp server. A tftp client request asking for a file x automatically translates to a file access to c:\tftproot\x.

[TFTP-Settings] **TransferWindow=1** Default:4	The tftp-setting TransferWindow sets the maximum number of tftp data packets that are sent before an acknowledge is received. TFTP protocol standard behavior is achieved by setting TransferWindow=1. The default of 4 should increase TFTP performance by about 50%. In case of problems try to set to 1.

Entries in the [HTTP-Settings] section

[HTTP-Settings] **EnableHTTP=1** Default:0	EnableHTTP enables the integrated HTTP Protocol server functionality of the DHCP Server.

[HTTP-Settings]

Root=c:\httproot

Default:none

Only files located in or under the given Root path are served by the http server. A Web browser request asking for a file x automatically translates to a file access to c:\httproot\x.

[HTTP-Settings]

Logfile=c:\httplog.txt

Default: none

The http server logs all activities in this file.

[content-type]

.htm=text/html

Default: -

The http server knows the content-type definitions for .htm, .html, .css, .xsl, .jpg, .png, .gif, .ico, .xml and .txt. Additional file extensions and their respective content types can be specified in the [content-type] section.

Entries in the [General], [General_x] or client section

All of the following entries can be specified in the client section [00-01-02-03-04-05], in the [General] or the [General_x] section. The response to the client is composed of all entries in these sections. Every information that was not found in the client section is taken from the general section. If it's in neither of both an appropriate default value is taken. See also the INI file overview.

AllowDNSQuery=1

AllowDNSQuery goes along with the built-in DNS functionality. A client who performs DNS queries that is not known to the DHCP Server (not in the ini file) can get a default IP address returned regardless of the name it wants to resolve. This is a security feature that limits the DNS function to known clients. Since V1.9.3 AllowDNSQuery defines this behavior on a per client basis. If AllowDNSQuery is set to 1 then the client can resolve names to IP addresses with DNS queries. If AllowDNSQuery is set to 0 then whatever is configured as DEFAULTIPADDR is returned for every name. Please assume the following INI file:

```
[General]
...
AllowDNSQuery=0
...

[DNS-Settings]
EnableDNS=1
DEFAULTIPADDR=192.168.0.1

[00-00-00-00-00-01]
IPADDR=192.168.0.10
Hostname=Computer_1

[00-00-00-00-00-02]
IPADDR=192.168.0.11
Hostname=Computer_2
AllowDNSQuery=1
```

In this example, only Computer_2 will be able to resolve DNS queries to the real IP addresses. Computer_1 will observe that regardless of the name he wants to resolve, he will always get 192.168.0.1 in return.

AutoConfig=12/29/2007 18:37:26

This entry marks a client section as being created by the DHCP Server (auto configuration). Don't touch it.

BOOTFILE=bootimage.bin

A string that typically defines the boot file which is fetched from a TFTP server by the client.

DNS_0=123.4.56.78
…
DNS_9=192.168.2.1

This entry sets the IP address of the DNS server. You can setup up to 10 DNS servers.

DOMAINNAME=mydomain	This entry should define the domain name that is send to the client machine. This is in most cases the name of the domain or workgroup that your server machine is in.
IPADDR=192.168.10.11	This is the IP address that has to be assigned to the client. This is the most important entry in the client section.
LeaseEnd=1198953446	This entry is automatically created by the DHCP Server. LeaseEnd is specified as number of seconds since midnight (00:00:00), January 1, 1970 and defines the time at which this particular IP address expires. Lease times are important when a client requests an IP address and all addresses in the IP pool are already assigned to other clients. In that case the client whose lease time (LeaseEnd) has expired least recently is deleted from the INI file and the available IP address is used for the request.
LEASETIME=3600 ; 1 hour Default: infinite	Lease time in seconds (decimal value). E.g. 86400 is the lease time for 1 day. Default is an infinite lease. The actual lease period for the clients IP address is the minimum of the configured LEASETIME and the lease duration the client is asking for.
NAME=ClientPC	“ClientPC” is send to the client as the hostname entry in the DHCP options. This is supposed to be the name of the client machine. This entry doesn't work with Windows clients, because they do not change the computer name.
NEXTSERVER=123.45.6.7	IP address of the "next server". This is typically used to specify a TFTP server in the BOOTP protocol.
NODETYPE=8	Default: 8, which means hybrid. I’ve never used something else than 8.
ROUTER_0=123.4.56.78 **...** **ROUTER_9=192.168.2.1**	This entry sets the IP address of the router or standard gateway. You can setup up to 10 routers by using the entries ROUTER_0 to ROUTER_9.
SUBNETMASK=255.255.255.0	The subnet mask is usually common to all clients and should be placed in the general section. This has to be the same as the subnet mask of the server machine that is setup in your network settings.

WINS_0=123.4.56.78 **...** **WINS_9=192.168.2.1**	This entry sets the IP address of the WINS servers. You can setup up the 10 WINS servers using the entries WINS_0 to WINS_9.
WPAD=`http://server.domain/config.pac`	Location of a proxy server (Web Proxy Auto Detection) used by Internet browsers to automatically detect proxy settings.

Custom options

The DHCP server supports custom options in addition to the above options. This allows to specifiy all possible DHCP options in client and general sections. (New in V1.7)

The syntax for custom options is:
[00-01-02-03-04-6A]
OPTION_nn="whatever text" ; text
OPTION_nn=02:03:04:05 ; hex bytes
OPTION_nn=192.168.2.1 ; IP address
OPTION_nn= 01 "whatever text" ; combination of hex byte and text

nn is the option number (decimal) such as OPTION_66 for TFTP server IP address.

Please note that all the examples above have a trailing comment in each line. This is necessary and is not optional. The comment even if it is only the semi-colon (;) is needed.

OPTION_nn="whatever text" ;

If there is no comment symbol at the end then the OPTION_nn setting will not be recognized correctly and is treated as malformed and therefore ignored.

Client sections with wildcards

A client section can be specified based on wildcards (since V1.7). Please use this with care. Recommendation is to not use client-id based specifications (UseClientID=0) and also to make sure that no interference with IPOOLs is possible, because the DHCP Server is not able to check this. This is how it works:

A client section can be defined based on wildcards like this:
[00-01-02-03-04-??]
IPADDR=192.168.17.%m5

What happens is that if a client that matches the above wildcard asks for an IP address, that the IP address assigned is automatically composed by using mac address byte 5 (%m5). The resulting mac address for client [00-01-02-03-04-6A] will be 192.168.17.106. (6A hex is 106 dec).
Mac address byte 0 through 5 are accessed accordingly with %m0 through %m5. The wildcard match algorithm searches in the following order and takes the first matching entry:
[00-01-02-03-04-6A]
[00-01-02-03-04-??]
[00-01-02-03-??-??]
[00-01-02-??-??-??]
[00-01-??-??-??-??]
[00-??-??-??-??-??]
[??-??-??-??-??-??]

In addition to the %m0 through %m5, since V2.3.1 there are further macros available. These are %ip0 through %ip3. They allow to access the IP address of the IPBIND_n address used by the current DHCP request. If IPBIND_1 is defined as IPBIND_1=192.168.5.1, then the macros are extended to: %ip0=192, %ip1=168, %ip2=5 and %ip3=1.

$(section\name) syntax for INI file variables

Since version 2.0 of the DHCP Server the $(section\name) syntax can be used in the INI file. This helps to keep the INI file simple and prevents repeating information such as IP addresses in many places. Here is an example of a typical INI file:

```
[Settings]
IPBIND_1=192.168.17.2
IPPOOL_1=192.168.17.2-50
AssociateBindsToPools=1

[DNS-Settings]
EnableDNS=1

[General]
SUBNETMASK=255.255.255.0
DNS_1=192.168.17.2
```

A clean and simple INI file with one IP pool and DNS enabled. If the IPBIND_1 interface ever changes, then one would need to change the IP address 192.168.17.2 in three places. No big deal but can be avoided with the following INI file utilizing the $(section\name) syntax:

```
[Settings]
IPBIND_1=192.168.17.2
IPPOOL_1=$(Settings\IPBIND_1)-50
AssociateBindsToPools=1

[DNS-Settings]
EnableDNS=1

[General]
SUBNETMASK=255.255.255.0
DNS_1=$(IPBIND_1)
```

If the interface ever changes then only IPBIND_1 needs to be edited. $(IPBIND_1) is equivalent to $(Settings\IPBIND_1). Settings is sort of the default section. $(section\name) can be used for everything in the INI file. It can even be used for INI file entries that the DHCP Server doesn't even know. Here is an example showing the usage for directories:

```
[Settings]
BaseDir="d:\dhcpsrv" ; dhcpsrv.exe resides here
IPBIND_1=192.168.17.2
IPPOOL_1=$(Settings\IPBIND_1)-50
AssociateBindsToPools=1
Trace=1
```

```
TraceFile="$(BaseDir)\dhcptrc.txt" ; trace file

[DNS-Settings]
EnableDNS=1

[General]
SUBNETMASK=255.255.255.0
DNS_1=$(IPBIND_1)

[TFTP-Settings]
EnableTFTP=1
Root="$(BaseDir)\wwwroot" ; use wwwroot for http and tftp

[HTTP-Settings]
EnableHTTP=1
Root="$(BaseDir)\wwwroot" ; use wwwroot for http and tftp
```

This way the base dírectory of DHCP Server can be changed easily in one single place and the trace file and all the other entries that refer to files are always correct.

You are welcome to E-Mail me if you have any questions or suggestions.