Scripting

Overview

You can create a script to automate a sequence of iPXE commands. Any command that can be typed at the iPXE command line can also be used in a script. You can find a full list of commands in the iPXE command reference.

An iPXE script is a plain text file starting with the magic line `#!ipxe` and containing a sequence of iPXE commands. For example, here is a simple script that acquires an IP address via DHCP and then boots the iPXE demonstration image:

```
#!ipxe

dhcp
chain http://boot.ipxe.org/demo/boot.php
```

Here is another simple script that creates a VLAN and then boots from it:

```
#!ipxe

vcreate --tag 24 net0
autoboot net0-24
```

Here is a slightly more sophisticated script that persistently retries DHCP until it succeeds in obtaining a boot filename:

```
#!ipxe

:retry_dhcp
dhcp && isset ${filename} || goto retry_dhcp
echo Booting from ${filename}
chain ${filename}
```

You can create an iPXE script using any text editor, such as emacs [http://www.gnu.org/software/emacs/], or vi [http://www.vim.org/], or even Windows Notepad [http://en.wikipedia.org/wiki/Notepad_%28software%29]. An iPXE script does not need to have any particular file extension (such as `.txt` or `.ipxe`); iPXE will recognise it as a script provided that it starts with the magic line `#!ipxe`.[1)]

Flow control

You can use the goto command to jump to a predefined script label. You can define a label using

```
:<label>
```

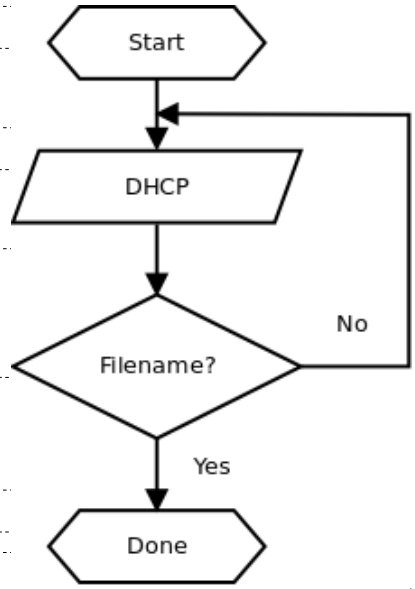
and jump to this label using

```
goto label
```

For example:

```
#!ipxe

:loop
echo Hello world
goto loop
```



You can use the && and || operators to conditionally execute commands depending on the status of previous commands. For example:

```
dhcp && echo DHCP succeeded
```

```
dhcp || echo DHCP failed
```

These operators can usefully be combined with the goto command to implement simple conditional flow control. For example, to keep retrying DHCP until it succeeds:

```
#!ipxe
```

```
:retry_dhcp
dhcp || goto retry_dhcp
```

You can use the ; operator to execute commands regardless of the status of previous commands. For example:

```
echo IP address: ${net0/ip} ; echo Subnet mask: ${net0/netmask}
```

You can terminate a script at any point using the exit command.

Error handling

iPXE will terminate a script immediately if any line of the script fails. For example, if you have the script:

```
#!ipxe

dhcp
route
```

then the script will terminate immediately if the dhcp command fails, without proceeding to the next line. You can override this behaviour using the || operator:

```
#!ipxe

dhcp ||
route
```

In this example, the empty command after the || operator is treated as "do nothing, successfully" (similar to `/bin/true` on a Unix-like operating system). Even if the dhcp command fails, the overall status of this line of the script will therefore always be "successful".

Comments

You can start a comment using the # symbol. For example:

```
# Obtain an address using DHCP
:retry
dhcp || goto retry # Keep retrying indefinitely
```

Advanced topics

Embedded scripts

You can embed a script within iPXE to override its default behaviour. For example, you may wish to build a version of iPXE containing an embedded script that uses DHCP to obtain an IP address but then boots from a predefined SAN target.

Dynamic scripts

An iPXE script does not have to be a static text file. For example, you could direct iPXE to boot from the URL

```
http://192.168.0.1/boot.php?mac=${net0/mac}&asset=${asset:uristring}
```

which would expand to a URL such as

```
http://192.168.0.1/boot.php?mac=52:54:00:12:34:56&asset=BKQ42M1
```

The `boot.php` program running on the web server could dynamically generate a script based on the information provided in the URL. For example, `boot.php` could look up the asset tag in a MySQL database to determine the correct iSCSI target to boot from, and then dynamically generate a script such as

```
#!ipxe

set initiator-iqn iqn.2010-04.org.ipxe:BKQ42M1
```

```
sanboot iscsi:192.168.0.20::::iqn.2010-04.org.ipxe:winxp
```

1) For the sake of backwards compatibility, iPXE will also recognise legacy gPXE scripts starting with the magic line `#!gpxe`. However, gPXE is not capable of running iPXE scripts, since the iPXE script language is substantially more advanced than the gPXE script language.