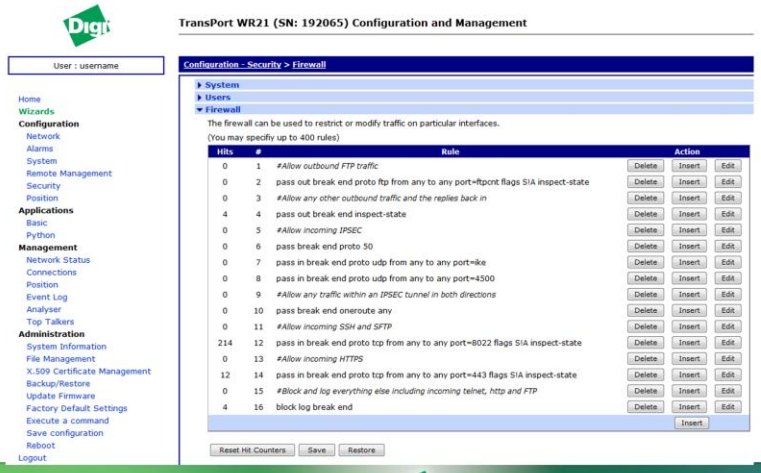

TransPort
Training
Program
TT006 – Firewall ABC
Features of the firewall
The firewall syntax
Public - © Digi International, Inc.
www.digi.com

Third Generation Firewall

- The TransPort firewall is a third generation *stateful* firewall (Enterprise firmware versions).
- Its primary (original) use is for security. Typically all packets entering or leaving the private network pass through the firewall, which examines each packet and blocks those that do not meet the specified security criteria.
- It can also detect problems and put routes/interfaces out of service (OOS)
- It can be test an interface before bringing it back into service
- NAT - It can be used to translate both source and destination IP addresses and port numbers

The Firewall Configuration Page

- The firewall configuration is implemented as a scripting language that can be accessed via the ***Configuration - Security > Firewall*** *web page*



The fw.txt can be edited in a text editor on a PC, etc. Using the WebUI will show syntax errors.

Notice the Hit Counter column.

Note: The default firewall rule set will show errors if loaded on a TransPort WR21 *Standard* version. This is because the Standard profile does not support “inspect-state”.

The Firewall Configuration Page

- ***Configuration - Security > Firewall***
  - Clicking the "***Insert***" button allows a new firewall rule to be entered
  - Clicking the "***Edit***" button allows an existing firewall rule to be edited
  - Clicking the "***Save FW → FW.txt***" button causes the current firewall rule set to be saved in the fw.txt file on the TransPort's flash RAM.
  - Clicking the "***Restore FW.TXT → FW***" button causes the current live firewall rule set to be replaced with the saved firewall rule set. Any recent edits will be lost.

The Firewall Status and Logs

- Firewall status and logs via ***Management - Network Status***

  Firewall shows statistics

  Allowed packets

  Blocked packets

  Stateful packets

  Stateful inspection table

  Firewall Trace shows packets that matched rules containing the “log” option

  Notice the Clear Trace option.

  Trace data is retained in the file fwlog.txt

  This is a virtual file, contents are cleared when power is disconnected.

Criteria Used for Packet Inspection

- Source and destination IP addresses
- Source and destination IP port or port ranges
- Type of protocol in use
- Direction of the packet (in or out)
- Interface type
- The eroute (IPsec VPN) the packet is on
- If an interface is in service or OOS (out of service)
- ICMP message type
- TCP flags (SYN, ACK, URG, RST, PSH, FIN)
- Status of a link and/or data packets on UDP/TCP and ICMP protocols

NOTE: Not all features are supported on the Standard edition firmware

Introduction To The Scripting Language

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

- Action
  block, pass, pass-ifup
- In-out
  in or out
- Options
  log, break, on, oneroute, oosed
- Type of service
- Protocol
- DNS list
- IP-range
- Inspect-state

Note: **`[inspect-state]`** is *not* supported on Digi TransPort *Standard* edition.

Introduction To The Scripting Language

- Firewall rules are processed in order from top to bottom.
- The first matching rule is used to process the packet.
- The most active rules should be placed at the top.
  - Reduces lookup times
  - Reduces CPU usage
  - Faster processing of packets

- If the packet does not match any rules, the firewall will silently drop the packet.
- When possible, always block traffic closest to the source (inboound).
- To log all dropped packets, the last rule should be:

```
block log break end
```


Note: **`[inspect-state]`** is *not* supported on Digi TransPort *Standard* edition.

Example script showing stateful behaviour

- Allow web access without stateful packet inspection

```
pass out break end on PPP 1 from any to any port = 80
pass in break end on PPP 1 from any port = 80 to any
block log break end
```

- Stateful rule with packet inspection

```
pass out break end on ppp 1 proto tcp from any to any
port=http flags S!A inspect-state
```

Packet fragments will not be pass through the firewall unless stateful inspection is used. No port numbers in fragment headers.

Demonstrate how the following firewall rule set does not allow web access:

pass out break end on PPP 1 from any to any port = 80

Add the following to debug and see the reason why:

→ pass out break end on PPP 1 from any to any port = 80
block log break end

After viewing the firewall trace add the following rule to fix the problem:

→ pass out break end on PPP 1 from any to any port = 80
pass in break end on PPP 1 from any port = 80 to any
block log break end

Demonstrate how the same can be achieved with a stateful rule:

pass out break end on ppp 1 proto tcp from any to any port=http flags S!A inspect-state

Explain that the “stateful” version is more secure because it only allows traffic in from IP addresses to which IP packets have already been sent, and also only to port numbers that match the source port of the original traffic.

Finally explain what S!A does and that it is optional. It may be necessary to leave this out in certain backup scenarios where a link comes into service and is expected to transfer data for a pre-existing TCP session.

Useful Quick Rules

```
pass break end
```

Allows all IP traffic in and out, stops processing further rules. Good for using as the last rule when developing a new rule set.

```
pass out break end inspect-state
```

Allows all IP traffic out but only allows packets in that are in reply to data originally sent out

```
# This is a comment line
```

Lines starting with a # are treated as a comment.

IP address translation

```
pass in|out from [source IP] [source port] to [destination
IP] [destination port] -> [new source IP] [new source
port] to [new destination IP] [new destination port]
inspect-state
```

- The above template can be used to construct a rule capable of changing
  - Source IP Address
  - Source IP Port
  - Destination IP Address
  - Destination IP Port

  Whilst still allowing the higher level protocol (e.g. TCP) to work
- Can be used for TCP, UDP and ICMP packets

FTP is a special case, using the FTP keyword in the

Firewall NAT example

```
pass in break end on ppp 1 proto tcp from any to 192.168.1.1
port=1234 -> to 172.16.0.10 port=http flags S!A inspect-state
```

The above example will demonstrate how a firewall translation rule can be used to divert traffic to a different destination IP address and port number

Pointing a web browser at http://192.168.1.1:1234/ will see packets redirected to 172.16.0.10 using TCP port 80

The Default Rule Set

- All TransPort routers have a ready to use set of firewall rules that can simply be enabled on the WAN interface.
- The default rules allow:

  All traffic from the LAN outbound and reply traffic back in.

  Inbound IPsec VPNs to be established.

  Inbound SSH, SFTP and HTTPS for secure router management.
- The default rules block:

  Everything else

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```


The “proto ftp” keyword informs the router that the traffic will be FTP control. Thus it will automatically set up stateful rules for PORT mode FTP data connections.

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```

The Default Rule Set

```
#Allow outbound FTP traffic
pass out break end proto ftp from any to any port=ftpcnt flags S!A inspect-state
#Allow any other outbound traffic and the replies back in
pass out break end inspect-state
#Allow incoming IPSEC
pass break end proto 50
pass in break end proto udp from any to any port=ike
pass in break end proto udp from any to any port=4500
#Allow any traffic within an IPSEC tunnel in both directions
pass break end oneroute any
#Allow incoming SSH and SFTP
pass in break end proto tcp from any to any port=22 flags S!A inspect-state
#Allow incoming HTTPS
pass in break end proto tcp from any to any port=443 flags S!A inspect-state
#Block and log everything else including incoming telnet, http and FTP
block log break end
```

Hands-On Practical Session – Firewall Basics

1. Configure the Firewall to allow all outbound traffic and block unauthorised inbound traffic on the WAN interface.
2. Add a rule to allow full management access to the TransPort router from only your IP address, enable the firewall on the LAN interface.
3. Add a rule to allow another person HTTP management access only.
4. Add a rule to allow another person Telnet management access only.
5. Review the firewall log for blocked traffic.

The Firewall

- You should now have a basic understanding of the firewall language and how the firewall works.
- The next slides go into more detail on the options and syntax available.

Modifying the firewall rules

- GUI - ***Security > Firewall***

Has syntax checking enabled

When adding a new rule, it is active immediately after clicking OK

When editing an existing rule, it is active immediately after clicking OK

- GUI - ***Administration - File Management > File Editor***

Simple GUI based file editor

No syntax checking whilst writing firewall rules

Rules will not become active until router is rebooted or firewall rules are activated. GUI, click Restore on firewall page; CLI, 'fw'.

- Offline method – download the fw.txt or copy & paste the rules to notepad

Edit the contents or create a new file named fw.txt.

Upload to the router using FTP, then activate the rules (GUI, Restore; CLI fw)

Interfaces with Firewall

- Firewall can be enabled on interfaces only
- ETH, PPP, TUN & OVPN.
- Eroute traffic is filtered on the interface like PPP or ETH or TUN. Syntax is ‘oneroute’.

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

Action:

Pass = Allow specified traffic

Block = Block specified traffic

Pass-ifup = Allow this traffic if the relevant interface is up.

This traffic will not activate an interface that is down/OOS.

in-out:

Imagine you are sat inside the router. Is the traffic coming in to an interface, or is it being sent out?

In = This rule relates to inbound traffic

Out = This rule relates to inbound traffic

Option negated – This rule applies to inbound and outbound traffic

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

options:

log = Log packets matching this rule to the firewall trace (fw.txt)

Creates a summary of the time stamp, the firewall rule, source and destination protocol, IP address & port.

Extended log options:

log snmp = matching packets create an SNMP trap

log syslog = matching packets create a syslog message

log event = matching packets create an event in the TransPort event log

log body = matching packets will be logged in the fw.txt with extra header and payload information can be used in conjunction with the above options.

```
block in log syslog break end on ppp 0 from 192.168.0.0/16 to any
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

options:
Break = If this rule matches, stop further processing of firewall rules and skip to the defined section.

break end = Stop processing rules and finish.

```
block in break end on eth 0 from 192.168.0.0/16 to any port=80
```

break label: = Stop processing rules and move to the **label:** section

```
# Check source address is from allowed subnets
pass in break allowed_traffic on eth 0 from 192.168.0.0/24 to any
pass in break allowed_traffic on eth 0 from 192.168.1.0/24 to any
# Block other sources
block in break end on eth 0 from any to any
allowed_traffic:
pass in break end from any to any port = 22 inspect-state
pass in break end from any to any port = 23 inspect-state
pass in break end from any to any port = 80 inspect-state
pass in break end from any to any port = 443 inspect-state
block log break end
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

options:

on = Allow or deny traffic ON the specified interface.

```
pass break end on ppp 1 from any to any inspect-state
pass in break end on tun 0 from any to any inspect-state
```

Option negated – This rule applies to all interfaces

oneroute = Allow or deny traffic on the specified IPsec eroute. Specify the eroute number, or 'any'.

```
pass out break end oneroute 0 192.168.0.0/24 to 10.10.10.0/24 inspect-state
pass in break end oneroute 0 10.10.10.0/24 to 192.168.0.0/24 inspect-state

pass break end oneroute any from any to any inspect-state
```

Firewall Syntax

[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]

options:

routeto = Redirect matching packets to the specified interface.

```
pass in break end routeto eth 1 from 10.1.0.0/16 to 172.16.0.1 port=telnet
```

oosed = Used to check the out of service status of the specified interface. Packets will only match this rule if PPP 1 is OOS.

```
block in break end oosed ppp 1 from 192.168.0.0/16 to any port=80
```

tos:

`tos` = Used to specify the Type of Service (TOS) to match.

```
block in on ppp 0 tos 0
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

proto:

Specifies the protocol that packets must match. The proto keyword must be followed by a protocol type.

Protocol type	Meaning
TCP, UDP	TCP or UDP packet
TCP	TCP packet
UDP	UDP packet
ICMP	ICMP packet
FTP	Monitor the TCP packets for FTP port numbers (control & data)
Decimal number	RFC protocol number. IE: 1=ICMP, 4=IPv4, 47=GRE, 50=ESP

```
pass break end on ppp 1 proto TCP from any to any port=22 inspect-state
```

Firewall Syntax

[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]

dnslist:

Specifies the full or partial domain or sub-domain that packets must match. The dnslist keyword must be followed by a list name.

Can be used as a simple white list or black list feature.

```
#dns white-list www.digi.com,*.supplier.co.uk,*.banking.com
pass out break end on ppp 1 proto udp dnslist white-list from any to any port=dns
block out break end on ppp 1 proto udp from any to any port=dns
pass out break end on ppp 1 from any to any port=80 inspect-state
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

ip-range:
Can be used to specify:
Source and/or destination IP address
- Specific host (192.168.1.10)
- Subnet (192.168.1.0/24 or '192.168.1.0 mask 255.255.255.0')
- All hosts/subnets (any)

Source and/or destination TCP/UDP port number
- Specific port number, equal to *n* (port = 23)
- Specific port number, not equal to *n* (port != 23)
- Port numbers less than *n* (port < 1024)
- Port numbers less than or equal to *n* (port <= 1023)
- Port numbers greater than *n* (port > 1023)
- Port numbers greater than or equal to *n* (port >= 1024)
- Range of port numbers to INCLUDE, min port >< max port (port 20 >< 80)
- Range of port numbers to EXCLUDE, min port <> max port (port 20 <> 80)

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

ip-range examples:

Allow all addresses

```
pass in break end from any to any inspect-state
```

When OK is clicked, this becomes:

```
pass in break end inspect-state
```

It is fine to write the longer syntax as it can look more logical.

Specifying a source IP address, single host:

```
pass in break end from 192.168.0.10 to any inspect-state
```

Specifying a destination IP address, single host:

```
pass in break end from any to 172.16.0.1 inspect-state
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

ip-range examples:

Specifying a source IP subnet:

```
pass in break end from 192.168.0.0 mask 255.255.255.0 to any inspect-state
```

Or

```
pass in break end from 192.168.0.0/24 to any inspect-state
```

Specifying a destination IP subnet:

```
pass in break end from any to 172.16.0.0 mask 255.255.255.0 inspect-state
```

Or

```
pass in break end from any to 172.16.0.0/24 inspect-state
```

Both source and destination options specified:

```
pass in break end from 192.168.0.0/24 to 172.16.0.0/24 inspect-state
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

ip-range examples:

Specifying a destination port number:

```
pass in break end from any to any port=80 inspect-state
```

Specifying a minimum source port number (greater than 1023):

```
pass out break end from 192.168.0.0/24 port>1023 to any inspect-state
```

Specifying a minimum source port number (greater than or equal to 1024):

```
pass out break end from 192.168.0.0/24 port>=1024 to any inspect-state
```

Specifying a range of port numbers to include (20 through to 80 inclusive)

```
pass out break end from any to 192.168.0.1 port 20><80 inspect-state
```


Common port numbers can be specified at the service name.
20 = ftpdat
21 = ftpcnt
23 = telnet
25 = smtp
80 = http
110 = pop3
500 = ike

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

ip-range TCP flags:

IP address and port syntax can be optionally followed with syntax for specifying TCP flags

Flag options:

S = SYN A = ACK P = PSH R = RST U = URG F = FIN

Flags that **must** be set are specified with the syntax:

`flags s`

This will check for the SYN flag being set.

Flags that **must not** be set are specified with the ! in the syntax:

`flags s!a`

This will check for the SYN flag being set and the ACK flag not set.

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

ip-range TCP flags example:

IP address and port syntax can be optionally followed with syntax for specifying TCP flags

Flag options:

S = SYN A = ACK P = PSH R = RST U = URG F = FIN

```
pass out break end on ppp 1 proto tcp from any to any port 20><80
flags S!A inspect-state
```

This rule will allow all outbound traffic on PPP 1 if:

The destination port number is between 20 and 80 inclusive.

The SYN flag is set and the ACK flag is not set. This must be the 1st packet of the TCP 3 way handshake.

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

inspect-state:

Looks at the packet in detail to determine the state of the connection.

Specifying inspect-state will allow the initial (first matching) packet through the firewall.

The firewall will then create a dynamic rule allowing the reply packets back in.

Without inspect-state being specified, inbound and outbound rules must be created for a single flow of 2 way traffic.

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

Port forwarding, NAT and re-directing IP traffic with the firewall:

The firewall can also be used to NAT packets by using the -> syntax after the ip-range object.

It is possible to change one or a combination of following:

Source IP address

Source TCP/UDP port number

Destination IP address

Destination TCP/UDP port number

When using the firewall to modify the IP address or port, **inspect-state is automatically implied** and does not need adding to the firewall rule.

pass out break end on ppp 1 from [src] [src_port] to [dest] [dest_port]-> [new_src] [new_port] to [new_dest] [new_port]

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

Port forwarding, NAT and re-directing IP traffic with the firewall examples:

To redirect from an old destination IP address to a new destination IP address (server change):

```
pass out break end on ppp 1 from any to 192.168.12.100 -> to 192.168.20.250
```

To redirect with port numbers included (IP address and port translation):

```
pass out break end on ppp 1 from any to 192.168.12.100 port=80 -> to
192.168.20.250 port=8080
```

To change the source IP address IP address (same as PPP NAT being enabled):

```
pass out break end on ppp 1 from 192.168.0.0/24 to any -> 213.218.111.27
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

Port forwarding, NAT and re-directing IP traffic with the firewall examples:

To change the source IP address and destination IP address:

```
pass out break end on ppp 1 from 192.168.0.0/24 to 192.168.12.100 ->
213.218.111.27 to 188.27.33.121
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

IP address place holders:

Anywhere there is an IP address or IP subnet specified, this can be replaced with a place holder that refers to an interface.

When the specified interface is active, the router uses the IP address of that interface in the firewall rule.

For example:

With PPP NAT enabled, all outbound traffic on PPP 1 will use the PPP 1 IP address as the source.

This is useful where an interface IP address can change.

```
pass out break end on ppp 1 from addr-ppp 1 to any inspect-state
```

Firewall Syntax

```
[Action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

IP address place holders:

The available options are…

- addr-ppp n
- addr-eth n
- addr-tun n

Where n = the interface number.

These options can also be followed by a CIDR notation to specify a subnet rather than the single interface IP address.

For example:

```
pass in break end on eth 0 from addr-eth 0/24 to any inspect-state
```

Hands-On Practical session - Firewall Syntax

Using the syntax we have discussed:

Allow all traffic inbound and replies back out. Simplest rule possible. Firewall enabled on ETH 0.

Block telnet traffic to ETH 0 and log attempts to connect.

Allow TCP 23 & 80 from your own PC IP address only.

hint: use a section label

Use the firewall to NAT the traffic exiting PPP 1, disable NAT on PPP 1.

Hands-On Practical session - Firewall Syntax

Allow all traffic inbound and replies back out. Simplest rule possible. Firewall enabled on ETH 0.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Allow all traffic inbound and replies back out. Simplest rule possible. Firewall enabled on ETH 0.

pass in break end inspect-state

Hands-On Practical session - Firewall Syntax

Block telnet traffic to ETH 0 and log attempts to connect.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Block telnet traffic to ETH 0 and log attempts to connect.

block in log break end on eth 0 from any to addr-eth 0 port = 23
pass break end

Hands-On Practical session - Firewall Syntax

Allow TCP 23 & 80 from your own PC IP address only.

Use a section label.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Allow TCP 23 & 80 from your own PC IP address only.

Use a section label.

```
pass in break allowed_traffic on eth 0 from 192.168.0.2 to addr-eth 0
block log break end
allowed_traffic:
pass in break end from any to any port=23 inspect-state
pass in break end from any to any port=80 inspect-state
block log break end
```

Hands-On Practical session - Firewall Syntax

Use the firewall to NAT the traffic exiting PPP 1, disable NAT on PPP 1.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Use the firewall to NAT the traffic exiting PPP 1, disable NAT on PPP 1.

pass out break end on ppp 1 from any to any -> addr-ppp 1

Hands-On Practical session - Firewall Syntax

Using the syntax we have discussed:

Allow telnet (TCP 23) inbound on Ethernet to ETH 0 IP address

Use source address, proto and flags.

Redirect TCP 2023 to TCP 23

Check you can telnet to port 2023 and log in.

Redirect TCP 8080 to the central router port 80.

Block inbound connections on ports 1 to 1024, log them.

Use 8023 & 8080 to manage the router.

Hands-On Practical session - Firewall Syntax

Allow telnet (TCP 23) inbound on Ethernet

Use source address, proto and flags.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Allow telnet (TCP 23) inbound on Ethernet

Use source address, proto and flags.

pass in break end on eth 0 proto tcp from addr-eth0/24
to addr-eth 0 port=23 flags s!a inspect-state

Hands-On Practical session - Firewall Syntax

Redirect TCP 2023 to TCP 23

Check you can telnet to port 2023 and log in.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Redirect TCP 2023 to TCP 23

Check you can telnet to port 2023 and log in.

pass in break end on eth 0 from any to addr-eth 0 port=2023 -> to addr-eth 0 port=23

Hands-On Practical session - Firewall Syntax

Redirect TCP 8080 to the central router port 80.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Redirect TCP 8080 to the central router port 80.

pass in break end on eth 0 from any to addr-eth 0
port=8080 -> to [IP address of router] port=80

Hands-On Practical session - Firewall Syntax

Block inbound connections on ports 1 to 1024, log them.

Use 8023 & 8080 to manage the router.

Answer coming up!

Hands-On Practical session - Firewall Syntax

Block inbound connections on ports 1 to 1024, log them.

Use 8023 & 8080 to manage the router.

block in log break end on eth 0 from any to addr-eth 0 port 1><1024

pass in break end on eth 0 inspect-state

End of Hans-On Practical Session