ICON-PEP-1.1

Icon-PEP Administration Guide

Table of Contents

1 Software Version

This guide is published in support of Isode Icon-PEP 1.1. It may also be pertinent to later releases. Please consult the release notes for further details.

2 Readership

This guide is intended for an administrator to set up and operate Icon-PEP.

3 How to use this guide

It is recommended that all administrators read the entire manual.

4 Typographical conventions

The text of this manual uses different typefaces to identify different types of objects, such as file names and input to the system. The typeface conventions are shown in the table below.

Object	Example
File and directory names	*isoentities*
Program and macro names	**mkpasswd**
Input to the system	`cd newdir`

Arrows are used to indicate options from the menu system that should be selected in sequence.

For example, **File** → **New** means to select the **File** menu and then select the **New** option from it.

5 File System placeholders

A number of directory names are given in the text, and the actual locations (below) vary depending on the target platform.

Name	Place holder for the directory used to store...	UNIX
(ETCDIR)	System-specific configuration files.	*/etc/isode/icon-pep*
(SBINDIR)	Programs run by the system administrators.	*/opt/isode/icon-pep/sbin*
(LOGDIR)	Log files.	*/var/log/isode/icon-pep*

6 Support queries and bug reporting

A number of email addresses are available for contacting Isode. Please use the address relevant to the content of your message.

1. For all account-related inquiries and issues: customer-service@isode.com [mailto:customer-service@isode.com]. If customers are unsure of which list to use then they should send to this list. The list is monitored daily, and all messages will be responded to.
2. To provide keys necessary to activate products, send the generated string to support@isode.com [mailto:support@isode.com] along with information on what is being evaluated or what has been purchased.
3. For all technical inquiries and problem reports, including documentation issues from customers and evaluators: support@isode.com [mailto:support@isode.com]. Customers should include relevant contact details in initial calls to speed processing. Messages which are continuations of an existing call should include the call ID in the subject line.
4. Customers may also submit support queries through the customer section of the Isode web site using the URL provided. Customers with silver or gold support may also submit support queries by telephone.
5. For all sales inquiries and similar communication: sales@isode.com [mailto:sales@isode.com].

Bug reports on software releases are welcomed. These may be sent by any means, but electronic mail to the support address listed above is preferred.

Isode sends release announcements and other information to the Isode News email list, which can be subscribed to from the address: http://www.isode.com/company/contact.html

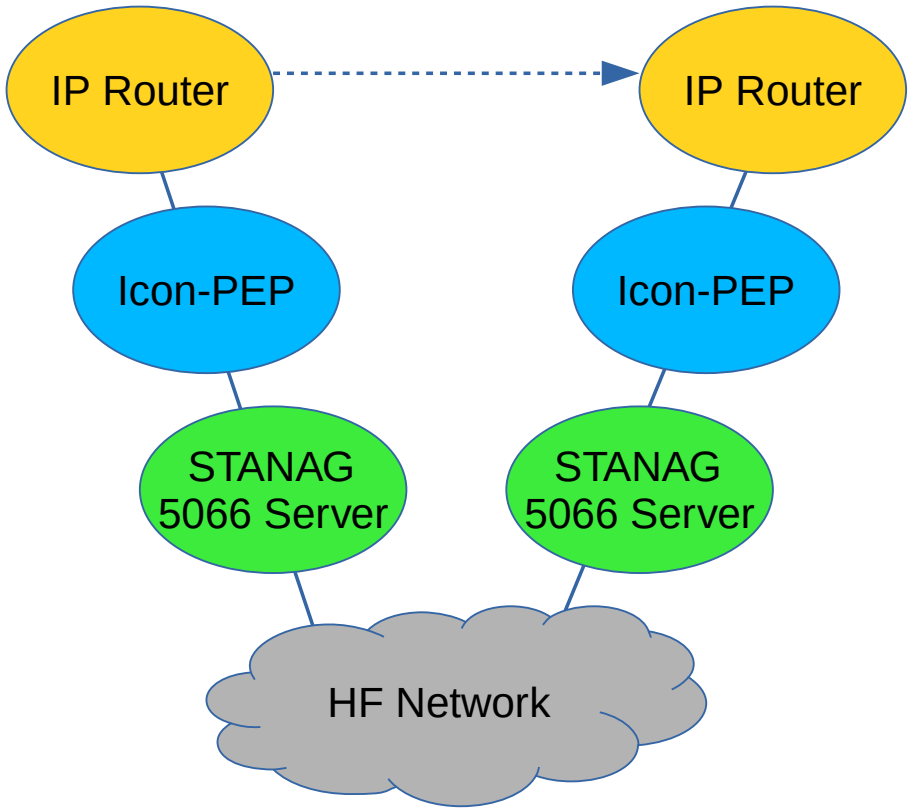
7 Export controls

UK Export Controls do not apply to Icon-PEP 1.1. It is anticipated the UK Export Controls will apply to future releases of Icon-PEP.

Chapter 1 Icon-PEP Overview

This chapter contains a general overview of Icon-PEP.

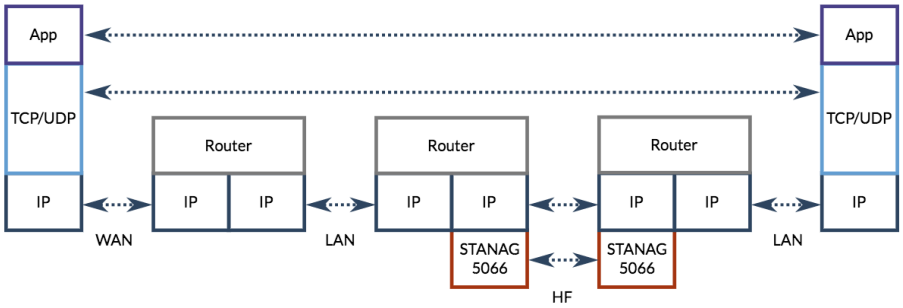
1.1 Icon-PEP Functionality



Icon-PEP enables provision of IP services over an HF network, with performance optimization for TCP and HTTP (Web) connections.

Icon-PEP operates over the STANAG 5066 HF Link layer, and connects to a STANAG 5066 server such as Isode's Icon-5066 product. Icon-PEP connects directly to an IP router, as illustrated above. Icon-PEP communicates with a peer server using standard protocols, to enable transfer of IP packets between a pair of routers.

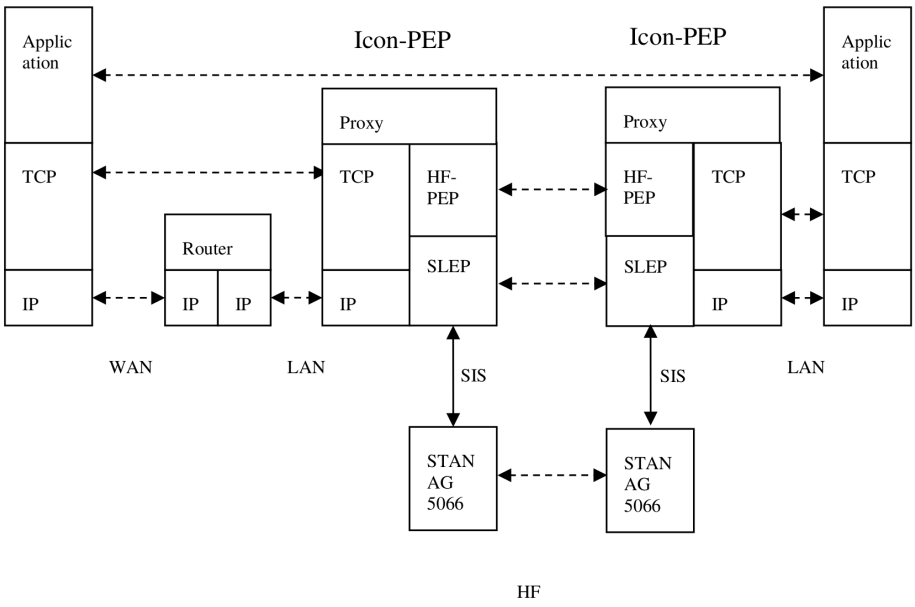
1.1.1 STANAG 5066 IP Client Service



STANAG 5066 Annex F.12 defines “IP Client” services for operating an IP subnet over HF, as shown in the diagram above. The approach is discussed in detail in the Isode white paper Measuring and Analysing STANAG 5066 F.12 IP Client [https://www.isode.com/whitepapers/measuring-stanag5066-f12-ip-client.html].

Icon-PEP supports this protocol, which is suitable for some low volume services such as ICMP Ping, and it works acceptably for some specialized military applications and services such as DNS Lookup.

1.1.2 HF-PEP Service



Use of IP Client leads to poor performance for bulk applications and in particular for TCP and HTTP. To address this, Icon-PEP includes a PEP (Performance Enhancing Proxy) architecture to efficiently provide support.

In this TCP Proxy architecture, an application is communicating over TCP, running over IP in the normal manner. The TCP connection from each application is peered with Icon-PEP, rather than the other application. Icon-PEP then communicates using the HF-PEP protocol specified in HF-PEP: STANAG 5066 TCP Performance Enhancing Proxy Protocol (S5066-APP9) [https://www.isode.com/whitepapers/S5066-APP9.html].

HF-PEP operates over SLEP (SIS Layer Extension Protocol), specified in S5066-APP3 [https://www.isode.com/whitepapers/S5066-APP3.html]. SLEP provides the Stream Services used by HF-PEP. SLEP communicates over STANAG 5066, using the local STANAG

5066 SIS (Subnet Interface Service) to connect. STANAG 5066 peers communicate over an HF network, as shown.

Icon-PEP can multiplex TCP connections over a single HF link, so that a single STANAG 5066 SAP can be shared by multiple TCP connections and multiple applications running over TCP.

Further details on HF-PEP and performance measurements are provided in the Isode white paper “Measuring and Analysing HF-PEP for TCP communication and Web Browsing over HF”.

1.1.3 Connections to STANAG 5066 Server

Icon-PEP connects to one or more STANAG 5066 servers using one or more connections using the STANAG 5066 SIS protocol. Two types of SIS connection are supported:

1. IP Client (default SAP: 9)
2. HF-PEP (default SAP: 13)

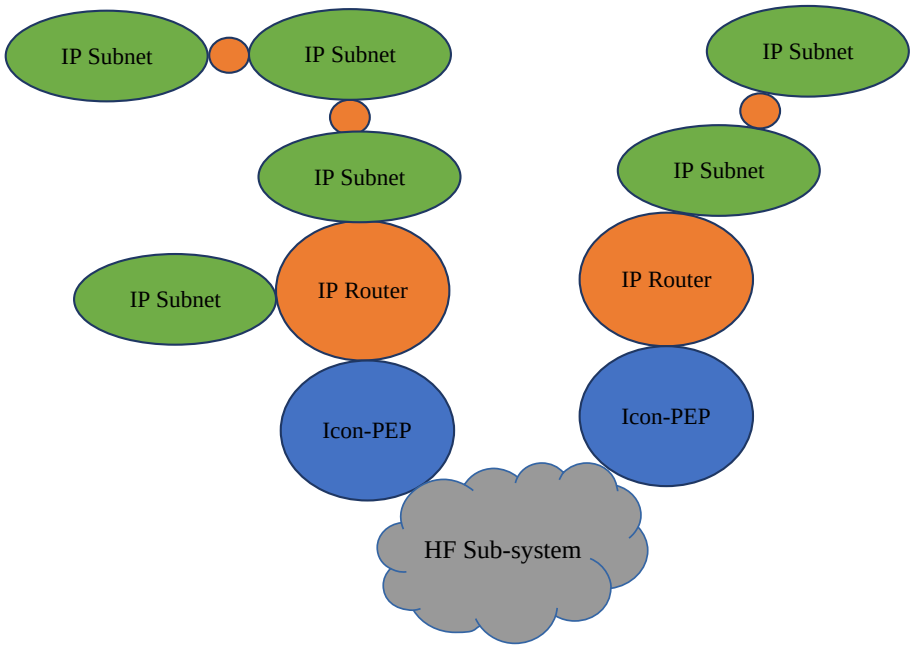
1.1.4 Connections to Router (GRE)

Icon-PEP communicates with one or more IP Routers using Generic Routing Encapsulation (GRE), specified in RFC 2784. GRE provides a simple mechanism to exchange packets between routers over IP, often described as a “GRE Tunnel”.

Icon-PEP terminates the GRE tunnel from a connected IP router. This means that there is no requirement for a peer system to use GRE. Routers commonly monitor GRE tunnel availability using ICMP Ping; Icon-PEP responds to these pings.

GRE is widely supported by Edge routers, and it is anticipated that Icon-PEP will generally connect to the deployed router. For subnets or single hosts that do not deploy a router, or where the deployed router does not support GRE, it is possible to use a software router, such as the one provided on many Linux systems or a product such as pfSense running in a virtual machine.

1.1.5 IP Routing and Deployment Model



The IP world comprises multiple IP Subnets, interconnected by IP routers. The routing configuration of an IP Router enables any IP packet to be correctly routed to its destination host via the IP Subnet to which the host is attached.

Icon-PEP core model enables connection of a pair of routers over HF. Icon-PEP simply takes a packet from the local router and sends it over HF to the peer IP router. This is an entirely mechanical function of switching data between STANAG 5066 and the local IP Router.

1.1.6 Icon-PEP Routing

The architecturally simple model described above requires that an IP Router connecting over HF to multiple peer routers requires an instance of Icon-PEP for every peer. This would add significantly to deployment complexity and in practice this approach is rarely needed.

Icon-PEP includes a static IP routing capability. This can be used for systems where all bar one node has a known fixed list of IP subnets. One node can be treated as a default route. This reflects a typical HF configuration, with multiple Mobile Units with a known set of IP Subnets, using a default route to a single shore system, which can have complex IP connectivity.

For performance reasons, it is desirable to avoid use of dynamic routing protocols over HF, so this approach is efficient.

Chapter 2 Configuring and Operating Icon-PEP

This chapter describes how to configure and operate Icon-PEP.

2.1 Installation

Installation of Icon-PEP is covered in the release notes.

2.2 Operation from Command Line

Icon-PEP can be run from a Linux command prompt. Operation from the command line is recommended for initial setup and testing, so that the command line monitoring can be used to facilitate debugging the configuration. See Section 2.6, “Command Line Monitoring” for details of the interface.

First stop the Icon-PEP systemd service if it is running:

systemctl stop iconpepd

Then run Icon-PEP manually using the following command:

(SBINDIR)/**isode.iconpepd -T**

Use **Ctrl-C** to stop Icon-PEP when running on the terminal.

A usage message showing other command-line options can be obtained by running Icon-PEP with **-?**:

(SBINDIR)/**isode.iconpepd -?**

2.3 Operation as a Service

Icon-PEP can be operated as a Linux systemd service using the **systemctl** service management capability. This should be configured on a deployed system. To start Icon-PEP as a service, use the following command:

systemctl start iconpepd

2.4 Echo server

For testing, Icon-PEP provides a built-in TCP echo server on port 7 of any remote IP address. When a TCP client such as **telnet** connects to any IP address which is handled by another node, the traffic goes out over HF to that node, and then Icon-PEP on that node echoes the data back again. This provides a test that transmission over HF to that node is operating correctly, without having to set up any services on that remote network to test against. For example:

telnet 10.0.2.1 7

2.5 Icon-PEP Configuration

Icon-PEP is configured using a single JSON file stored in:

(ETCDIR)/pep.json

The rest of this section explains, by example, the information needed in this file.

2.5.1 Example Configurations

This section provides two complete example configuration files, to show how the whole configuration fits together. Subsequent sections describe features of these configurations, in most cases using extracts from these examples.

The first configuration supports IP Client and HF-PEP traffic in both directions via a GRE tunnel to/from a router:

```
{
    "units": [{
        "node": "4.0.0.1",
        "routes": [{
            "subnet": "default",
            "node": "4.0.0.1"
        }, {
            "subnet": "10.0.2.0/24",
            "node": "4.0.0.2"
        }],
        "routers": [{
            "subnet": "default",
            "router": "127.99.99.20"
        }],
        "gre": {
            "bind_addr": "127.99.99.21",
            "intercept_ping": [
                "10.0.2.1/32"
            ],
            "pcap": "gre.pcap"
        },
        "f12": {
            "sis_addr": "127.0.0.1",
            "sis_port": 5066,
```

```
            "sis_bind_rank": 8,
            "sis_ttl": 0,
            "traffic_rules": [{
                "protocol_port_match": [ "udp:57621" ],
                "priority": 8,
                "arq": true,
                "drop": true
            },{
                "priority": 8,
                "arq": true,
                "drop": false
            }],
            "dump": "dump.log"
        },
        "pep": {
            "tun_v4_subnet": "172.30.0.0/16",
            "sis_pri": 15,
            "compress": true,
            "slep": {
                "sis_addr": "127.0.0.1",
                "sis_port": 5066,
                "sis_sapid": 13,
                "sis_bind_rank": 8
            }
        }
    }]
}
```

The second configuration supports outgoing IP Client and HF-PEP connections to the LAN via NAT proxies, but does not support incoming connections from the LAN:

```
{
    "units": [{
        "node": "4.0.0.1",
        "routes": [{
            "subnet": "default",
            "node": "4.0.0.1"
        }, {
            "subnet": "10.0.2.0/24",
            "node": "4.0.0.2"
        }],
        "routers": [{
            "subnet": "default",
            "router": "127.99.99.20"
        }],
        "f12": {
            "sis_addr": "127.0.0.1",
            "sis_port": 5066,
            "sis_bind_rank": 8,
            "sis_ttl": 0,
            "traffic_rules": [{
                "priority": 8,
                "arq": true,
                "drop": false
            }]
        },
        "nat": {},
        "nattcp": {
            "compress": true,
            "slep": {
                "sis_addr": "127.0.0.1",
                "sis_port": 5066,
                "sis_sapid": 13,
                "sis_bind_rank": 8
```

```
            }
        }
    }]
}
```

2.5.2 Node Definition

```
{
    "units": [{
        "node": "4.0.0.1",
```

The start of the configuration file specifies the STANAG 5066 address of the node being configured, in the above example `"4.0.0.1"`.

For node addresses, where the first number of the dotted-quad is below 16, leading nybbles are stripped off and the remaining nybbles are taken to be the STANAG 5066 node address. So the shortest STANAG 5066 node address would be in the range 0.0.0.0 to 0.0.0.15, which corresponds to a single nybble (4 bits). If the first number is 16 or over, that represents either a group address or an unnormalized node address, but these are unlikely to be required in this application.

2.5.3 Routes

```
        "routes": [{
            "subnet": "default",
            "node": "4.0.0.1"
        }, {
            "subnet": "10.0.2.0/24",
            "node": "4.0.0.2"
        }],
```

This section maps from IP addresses to STANAG 5066 node addresses. It specifies which node to use to reach the given IP subnet. Any packet that is addressed to an IP address that is not found in the routes table will be unroutable. The routes table would normally be the same for all nodes on an HF network that wish to communicate. If any node in the group has a different routes table, then some return IP packets might be unroutable. Icon-PEP warns about unroutable packets.

Subnets are specified in standard IPv4 or IPv6 network address/netmask format, with `"10.0.2.0/24"` in the above example. Each subnet is associated with a node identified by its STANAG 5066 address. Multiple subnets may be associated with a node.

One node can be specified with the special subnet `"default"`. No other subnets may be assigned to this node. A default route handles all IP addresses not handled by any other subnet. It would typically be used if there is a node that acts as a gateway to a wider network or to the internet.

2.5.4 Routers

```
        "routers": [{
            "subnet": "192.168.140.0/24",
            "router": "127.99.99.20"
        }, {
            "subnet": "192.168.150.0/24",
            "router": "127.99.99.20"
        }],
```

The routers section specifies the local LAN-side configuration, which will be different for each STANAG 5066 node. Each entry maps from an IPv4 or IPv6 subnet to the IP router on the LAN which handles routing packets to that subnet. Each entry in the routers section has two elements:

`subnet`
: This identifies a subnet in the standard format. It may also be `"default"` to handle all traffic not handled by any other entry.

`router`
: The IP address of the router which routes packets to/from that subnet. This is the address that GRE traffic will be directed to.

2.5.5 GRE Configuration

```
        "gre": {
            "bind_addr": "127.99.99.21",
            "intercept_ping": [
                "10.0.2.1/32"
            ],
            "pcap": "gre.pcap"
        },
```

GRE is configured by the `"gre"` element of the configuration, which has the following parameters:

`bind_addr`
: The IP address of this machine (the machine running Icon-PEP) on the network where the GRE routers are found. This should be the IP address that those routers will direct GRE traffic to in order to reach this machine.

`intercept_ping`
: Some routers will configure GRE tunnels with “internal” addresses, and will send ICMP Pings to the remote end of the GRE tunnel. Icon-PEP needs to intercept and respond to these pings, without sending them over HF. This configures a set of addresses as a subnet to which Icon-PEP will respond. This is typically a single IP address (subnet mask of 32).

`pcap`
: If configured, IP packets passed over GRE will be recorded in the file named in this parameter, in PCAP format, which can be used by the Wireshark tool to inspect the traffic. The PCAP file is written to the *(LOGDIR)*.

Advanced parameters, which default to safe values:

`seq_enable`
: If true, adds GRE sequence numbers. Defaults to false.

2.5.6 IP Client Connections

```
        "f12": {
            "sis_addr": "127.0.0.1",
            "sis_port": 5066,
            "sis_bind_rank": 8,
            "sis_ttl": 0,
            "dump": "dump.log"
        },
```

The IP Client (Annex F.12) connection is configured with the `"f12"` element, which has the following parameters:

`sis_addr`
The IP address of the STANAG 5066 Server.

`sis_port`
The IP port used by SIS.

`sis_bind_rank`
The STANAG 5066 rank of the connection.

`sis_ttl`
The TTL used in seconds. If unspecified, defaults to 3600s (1 hour).

`dump`
If specified, IP Client Traffic is written to the given filename, under *(LOGDIR)*

`traffic_rules`
See Section 2.5.10, “Annex F.12 traffic filtering”.

Advanced parameters, which default to safe values:

`purge_packet_age`
Purge queues if oldest packet reaches this age (in seconds). Defaults to 600 (10 minutes).

`purge_packet_count`
Purge queues if packet count reaches this level. Default is 20,000.

2.5.7 HF-PEP

```
        "pep": {
            "tun_v4_subnet": "172.30.0.0/16",
            "sis_pri": 15,
            "compress": true,
            "slep": {
                "sis_addr": "127.0.0.1",
                "sis_port": 5066,
                "sis_sapid": 13,
                "sis_bind_rank": 8
            }
        }
```

HF-PEP can be optionally configured by use of the `"pep"` element of the configuration. This terminates TCP connections locally and forwards only the stream data instead of the whole IP packet. Data is forwarded using the SLEP protocol. This has the following parameters:

`tun_v4_subnet`
This is the subnet used by Icon-PEP for the IPv4 TUN device, which must be specified if IPv4 traffic is to be handled. This must be a subnet within a private-use network range that is not already being used on the LAN, i.e. within one of the ranges: 10.0.0.0/8 or 172.16.0.0/12 or 192.168.0.0/16. Each IP address in the range allows 16384 TCP connections to be terminated. These addresses are only used locally for terminating TCP connections, and do not appear in packets that leave Icon-PEP.

`sis_pri`
The STANAG 5066 priority to be used by default for HF-PEP traffic.

`compress`
Controls whether SLEP stream compression is used for outgoing data.

`slep`
Configuration of SLEP. See Section 2.5.11, “SLEP parameters”.

Advanced parameters, which default to safe values:

`tun_v6_subnet`
: This is the subnet used by Icon-PEP for the IPv6 TUN device, which must be specified if IPv6 traffic is to be handled. This must be a subnet within a private-use network range that is not already being used on the LAN, i.e. within the range: fd00::/8. Each IP address in the range allows 16384 TCP connections to be terminated.

`connection_limit`
: If specified, then Icon-PEP aborts any new incoming TCP connection from the LAN once the given number of simultaneous connections is reached, until it drops back again. This does not limit new incoming connections coming from the HF side.

`start_delay`
: Delay in milliseconds before making a SLEP connection, to help initial TCP data to be included in the same transmission as the SLEP Init Request. This should be larger than the maximum round-trip-time between clients on the LAN and this server. Default is 200ms.

2.5.8 NAT proxy for UDP and ICMP ping

```
        "nat": {},
```

This sets up a NAT proxy for UDP connections and ICMP pings. When these are received from HF as IP Client packets, they are forwarded to the local network. A GRE tunnel cannot be configured along with this option.

This has no required parameters, but there are two optional advanced parameters for tuning:

`udp_timeout_s`
: UDP timeout in seconds. If there has been no activity for this length of time, the UDP connection is dropped, meaning that any further return packets will be lost. The default is 300 seconds.

`forward_pings`
: Whether to forward ICMP pings to the local network or not, either true or false. Defaults to true.

2.5.9 NAT proxy for TCP traffic

```
        "nattcp": {
            "compress": true,
            "slep": {
                "sis_addr": "127.0.0.1",
                "sis_port": 5066,
                "sis_sapid": 13,
                "sis_bind_rank": 8
            }
        }
```

This sets up a NAT proxy for TCP connections relayed over HF using HF-PEP. It cannot be used in combination with an HF-PEP or GRE configuration in this Icon-PEP instance, since it takes charge of forwarding the TCP connections to the LAN. However the Icon-PEP instance at the remote station would typically need to be configured with both HF-PEP and GRE.

`compress`
: Controls whether SLEP stream compression is used for outgoing data.

`slep`
: Configuration of SLEP. See Section 2.5.11, “SLEP parameters”.

2.5.10 Annex F.12 traffic filtering

```
            "traffic_rules": [{
                "protocol_port_match": [ "udp:57621" ],
                "priority": 8,
                "arq": true,
                "drop": true
            },{
                "priority": 8,
                "arq": true,
                "drop": false
            }],
```

Incoming traffic is matched against the given list of traffic rules. The first rule that matches is used. If a packet doesn't match any rule, then it is dropped. Within each rule the following match filters limit the match. If no filters are specified, then the rule matches all packets:

`protocol_port_match`
: Matches according to protocol and port number. For example, `tcp:80` matches TCP traffic addressed to port 80, `udp:53` matches UDP traffic addressed to port 53, `icmp` matches all ICMP traffic, and `other:2` matches IP protocol number 2 (which is IGMP). Note that `other:???` is used for any IP protocol number apart from TCP, UDP or ICMP.

`dst_addr_match`
: Matches according to the packet's destination IP address. The value is a list of subnets, and the IP address matches if it is within any one of those subnets. For example: `"dst_addr_match": ["192.168.50.0/24", "192.168.57.0/24"]`

`src_addr_match`
: Matches according to the packet's source IP address. The value is a list of subnets, and the IP address matches if it is within any one of those subnets.

`dscp_match`
: Matches according to the packet's IP header DSCP value (0-63). If the DSCP value is within the list of integers provided, then the packet matches.

If a rule matches, then the following values control what happens to the packet:

`drop`
: `true`: drop the packet, `false`: keep the packet

`priority`
: SIS priority (0-15) to use for this packet

`arq`
: `true`: use ARQ, `false`: don't use ARQ

Advanced parameters, that default to safe values:

`arq_in_order`
: `true`: enable in-order delivery for ARQ. By default this is disabled because it holds up all of the traffic if any single DPDU has an error.

2.5.11 SLEP parameters

```
            "slep": {
                "sis_addr": "127.0.0.1",
                "sis_port": 5066,
                "sis_sapid": 13,
                "sis_bind_rank": 8
            }
```

SLEP is the stream transport layer used to transfer TCP stream data over HF. It has the following parameters:

`sis_addr`
: The IP address of the STANAG 5066 Server.

`sis_port`
: The IP port used by SIS.

`sis_sapid`
: The SAP used by HF-PEP. SAP 13 is recommended.

`sis_bind_rank`
: The STANAG 5066 rank of the connection.

`sis_ttl`
: The TTL used in seconds. If unspecified, defaults to 3600s (1 hour).

SLEP has a number of advanced parameters for tuning. These default to safe values, and in general shouldn't be modified without a good understanding of SLEP. These are the advanced parameters:

`ack_after_blocks`
: Send a SLEP ACK after this many blocks. Defaults to 10.

`queued_peer_delay`
: QUEUED_PEER_DELAY timer from SLEP standard. Defaults to 10s.

`stream_init_timeout`
: STREAM_INIT_TIMEOUT timer from SLEP standard. Defaults to 300s.

`stream_init_count`
: Maximum number of times to send the Stream Init. Defaults to 3.

`empty_block_timeout`
: EMPTY_BLOCK timer from SLEP standard. Defaults to 180s.

`wait_for_next_data_timeout`
: WAIT_FOR_NEXT_DATA timer from SLEP standard. Defaults to 5s.

`stream_broken_timeout`
: STREAM_BROKEN timer from SLEP standard. Defaults to 900s (15 minutes).

`stream_close_timeout`
: STREAM_CLOSE timer from SLEP standard. Defaults to 300s (5 minutes).

`stream_close_count`
: Number of times to send Stream Close if there is no response. Defaults to 3.

`final_data_wait_timeout`
: FINAL_DATA_WAIT timer from SLEP standard. Defaults to 1200s (20 minutes).

`repeat_block_timeout`
: Time after which to request a repeat of blocks that are missing. Defaults to 600s (10 minutes).

`force_close_after_eof_timeout`
: Time after which to force close of a TCP connection if the remote end has closed, but this end hasn't. Defaults to 60s.

`hold_node_xfid_timeout`
: Time to hold onto a SLEP XFID after the stream has closed. Defaults to 600s (10 minutes).

`immconn_tx`
: Defaults to true, which enables Immediate Connect for outgoing connections. This allows transmitting data along with the SLEP Stream Init Request, which avoids an HF round-trip for protocols which connect and immediately send a request, such as HTTP. The remote end must also support Immediate Connect. This will improve performance in most situations.

`immconn_rx`
: Defaults to true, which allows Immediate Connect from remote nodes. This will improve performance in most situations.

`immconn_timeout`
: Time to wait for retransmission of the Stream Init Request if data arrives without a Stream Init Request. Defaults to 300s.

`immconn_rx_limit`
: Maximum amount of early data accepted. Both ends must use the same setting. Defaults to 5000 bytes.

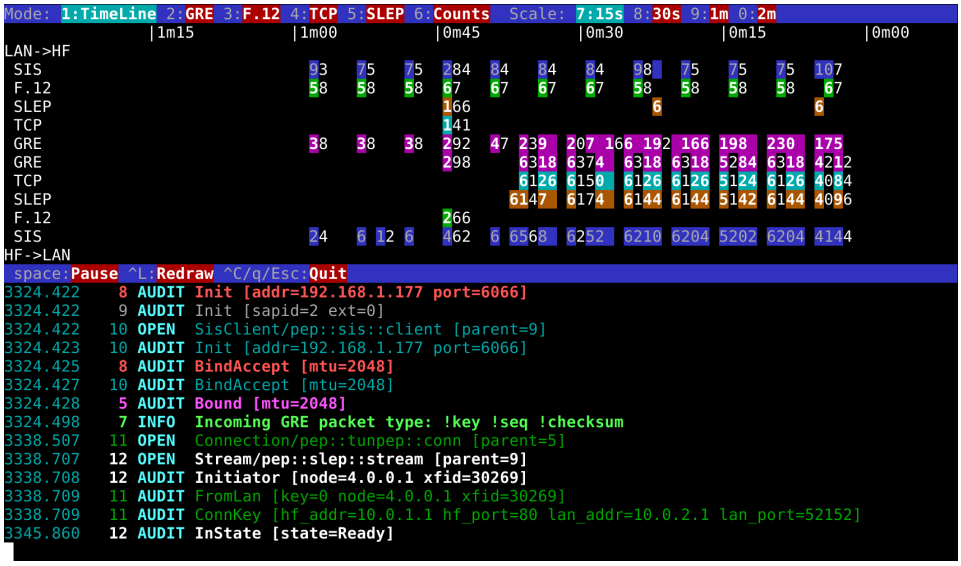
`immconn_tx_limit`
: Maximum amount of early data sent. Both ends must use the same setting. Defaults to 5000 bytes.

`retry_stream_init`
: If true, retries a Stream Init even if the connection appears to be down. Defaults to false.

`pcap`
: If configured, the SIS conversation is dumped to the named PCAP file, within *(LOGDIR)*.

2.6 Command Line Monitoring

Figure 2.1. Command Line Monitoring



The command mode of Icon-PEP provides a simple character based GUI to visualize data flowing over GRE, TCP, SLEP, IP Client and STANAG 5066 SIS. This gives a front-panel display to Icon-PEP that is useful during initial setup or whilst debugging a configuration. There are the following options, accessed by keyboard control:

1. Timeline (shown above), gives a graphical flow of inbound and outbound traffic at each level. Traffic moving from the LAN towards HF appears in the upper part, and from HF towards the LAN in the lower part.
2. GRE. Data flowing at the Generic Routing Encapsulation layer (GRE).
3. IP Client (F.12). Data flowing over STANAG 5066 Annex F.12 IP Client.
4. TCP. Data flowing in TCP Tunnel.

5. SLEP. Data from TCP Tunnel flowing over STANAG 5066 SLEP Streaming Service.
6. Counts. Show summary of packet counts at each layer.

2.7 Logging

Logging appears in dated files under *(LOGDIR)*. Files roll over daily. In the following example, some lines have been wrapped to fit the page width.

```
20201125-092038.769    0 INFO  Timezone is -0500
20201125-092038.769    0 INFO  Icon-PEP startup arguments: -T
20201125-092038.770    1 OPEN  Unit/iconpepd::unit
20201125-092038.771    2 OPEN  Tui/iconpepd::tui
20201125-092038.772    3 OPEN  GreThreads/pep::gre::threads
  [parent=1]
20201125-092038.772    4 OPEN  AnnexF12/pep::annex_f12 [parent=1]
20201125-092038.773    5 OPEN  TunPep/pep::tunpep::api [parent=1]
20201125-092038.774    6 OPEN  Terminal/stakker_tui::terminal
  [parent=2]
20201125-092038.774    3 AUDIT Init [bind_addr=127.99.99.21]
20201125-092038.775    0 INFO  PCAP file created
  [fnam=gre-20201125-092038.pcap]
20201125-092038.775    7 OPEN  GreDecode/pep::gre::decode
  [parent=3]
20201125-092038.776    8 OPEN  SisClient/pep::sis::client
  [parent=4]
20201125-092038.777    0 INFO  DUMP file created
  [fnam=dump-20201125-092038.log]
20201125-092038.777    5 AUDIT Init [subnet=Subnet(172.30.0.0/16)]
20201125-092038.778    9 OPEN  Slep/pep::slep::api [parent=5]
20201125-092038.790    8 AUDIT Init [addr=127.0.0.1 port=5066]
20201125-092038.791    9 AUDIT Init [sapid=2 ext=0]
20201125-092038.791   10 OPEN  SisClient/pep::sis::client
  [parent=9]
20201125-092038.792   10 AUDIT Init [addr=127.0.0.1 port=5066]
20201125-092038.795    8 CLOSE SIS failure: Disconnect [failed]
20201125-092038.798   10 CLOSE SIS failure: Disconnect [failed]
20201125-092038.798    4 AUDIT SisRestart [after=2
  cause="Actor failed: SIS failure: Disconnect"]
20201125-092038.799    9 AUDIT SisRestart [after=2
  cause="Actor failed: SIS failure: Disconnect"]
20201125-092040.794   11 OPEN  SisClient/pep::sis::client
  [parent=4]
20201125-092040.794   12 OPEN  SisClient/pep::sis::client
  [parent=9]
20201125-092040.795   11 AUDIT Init [addr=127.0.0.1 port=5066]
20201125-092040.796   12 AUDIT Init [addr=127.0.0.1 port=5066]
20201125-092040.809   11 AUDIT BindAccept [mtu=2048]
20201125-092040.818   12 AUDIT BindAccept [mtu=2048]
20201125-092040.818    5 AUDIT Bound [mtu=2048]
20201125-092041.775    7 INFO  Incoming GRE packet type: !key !seq
  !checksum
20201125-092100.491   13 OPEN  Stream/pep::slep::stream [parent=9]
20201125-092100.493   13 AUDIT Responder [node=4.0.0.2 xfid=37519]
20201125-092100.493   14 OPEN  Connection/pep::tunpep::conn
  [parent=5]
20201125-092100.494   14 AUDIT FromSlep [key=0 node=4.0.0.2
  xfid=37519]
20201125-092100.495   13 AUDIT InState [state=Ready]
20201125-092100.498   14 AUDIT ConnKey [hf_addr=10.0.2.1
```

```
  hf_port=53096 lan_addr=10.0.1.1 lan_port=80]
20201125-092106.509   13 AUDIT OutState [state=Flushing]
```

The logging columns are as follows:

`20201125-092106.509`
: Date as YYYYMMDD, time as HHMMSS and subsecond time in milliseconds.

`13`
: Logging span number. Each instance of a component has a different logging span number. All logging from that component, between `OPEN` and `CLOSE`, is logged against the same span.

`AUDIT`
: Type of logging record: `OPEN` and `CLOSE` record start and end of a span, `AUDIT` logs machine-readable data, `TRACE`, `DEBUG`, `INFO`, `WARN` and `ERROR` log human-readable information at various severity levels.

`OutState [state=Flushing]`
: Data associated with the logging record. The values within square brackets are machine-readable key-value pairs.