Teldat Router

IP Tunnel Interface (TNIP)

Doc. *DM719-I* Rev. *10.10*
April, 2003

INDEX

Chapter 1
IP Tunnel Interface (TNIP)

1. Description

1.1. Introduction

This is known as *Tunnel process* (Tunneling), the procedure through which diverse protocol packets are encapsulated in another protocol. This function is implemented through a virtual interface which is known as *Tunnel Interface*. The Tunnel interface is not initially linked to any transport protocol, encapsulation or internally set, but is an architecture providing the necessary services to implement any standard encapsulation scheme. As the tunnels are end-to-end links, you must configure independent tunnels for each link.

The Tunnel process has three components:

- Internal Protocol, payload protocol or transport protocol: this is the protocol being encapsulated (IP or SRT).
- Carrier Protocol: This is the protocol which encapsulates.
    - ◊ Generic Routing Encapsulation (GRE).
- Transport protocol, Delivery protocol: This is the protocol which transports the encapsulated payload protocol (Only IP).

1.2. Advantages of tunneling

There are various situations where encapsulating one protocol in another is useful:

- To interconnect multiprotocol local networks through a backbone with a single protocol.
- To resolve interconnection problems for networks containing protocols with a limited number of hops and without this procedure cannot connect.
- To connect two non-consecutive subnets.
- To permit Virtual Private Networks throughout the WAN networks.

1.3. Special considerations

The following points describe considerations and precautions that should be borne in mind when configuring tunnels:

- Encapsulation and decapsulation produced at the tunnel ends are slow operations.
- You must take care when configuring and bear in mind the possible security and topology problems. E.g. you could configure a tunnel whose source and destination are not restricted by Firewalls.
- You must correctly select the methods through which the tunnel will go. Cases could arise such as crossing Fast FDDI networks and slow links of 9600 bauds. Some payload protocols do not behave well in networks composed of mixed methods.
- Many end-to-end tunnels can saturate the link with routing information.
- Those routing protocols which decide the best route solely based on the number of hops prefer the tunnel even if a better route exists. The tunnel always appears to be one hop even though the cost may be higher.

- An even worse problem that can occur is if the routing information on the networks connected through the tunnel gets mixed up with the information on the networks transporting this information. In these cases, the best route towards the tunnel destination is through the tunnel. This type of route is known as *recursive route* and provokes a temporary drop in the tunnel. In order to avoid this problem, you must maintain the routing information independently;
    - ◊ Using a distinct AS number or TAG.
    - ◊ Using a different routing protocol.
    - ◊ Using static routes for the first hop (but being careful with route loops).

2. Structure of the encapsulated frame

In the case of IP Tunnel, the transport or delivery protocol is IP therefore the structure of the encapsulated frame is as follows:

Delivery protocol header: IP
Encapsulation protocol header
Payload protocol packet

2.1. IP over IP with GRE

In this case the encapsulated controlling protocol is GRE. And the payload protocol or the protocol being encapsulated is IP. The encapsulated protocol GRE (Generic Routing Encapsulation) is described in the RFC1701 and this particular case of IP over IP with GRE in the RFC1702.

Delivery protocol header: IP
Encapsulation protocol header: GRE
Payload protocol: IP

The IP header is beyond the limits of this document although not the GRE header. The GRE header has the following form:

```
                    1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

C	R	K	S	s	Recur	Flags	Ver	Protocol Type
Checksum (optional)								Offset (optional)
Key (optional)								
Sequence Number (optional)								
Routing (optional)								

Checksum Present (bit 0) (C)

If this is set to 1, then the ***checksum*** field is present and contains valid information. If either the ***checksum*** or the ***routing*** bit is present, both the ***checksum*** and ***offset*** fields are present in the packet.

Routing Present (bit 1) (R)

Not used.

Key Present (bit 2)

If this is set to 1, then the ***key*** field (or identifier) is present in the packet and contains a valid value.

Sequence Number Present (bit 3)

If this is set to 1, then the ***Sequence Number*** field is present and contains a valid value.

Strict Source Route Present (bit 4)

Not used.

Recursion Control (bits 5-7)

Recursion control contains a three bit unsigned integer which contains the number of additional encapsulations which are permissible. This is always zero.

Version Number (bits 13-15)

Always 0.

Protocol Type (2 octets)

Contains the payload protocol type.

Offset (2 octets)

The offset field indicates the octet offset from the start of the ***routing*** field to the first route to be examined.

Checksum (2 octets)

Contains the IP ***checksum*** of the GRE header and the payload packet.

Key (4 octets)

Tunnel identifier.

Sequence Number (4 octets)

This is the Number used by the receiver to establish the correct order in which packets arrive.

Routing (variable length)

Does not exist.

When IP is encapsulated in IP using GRE, the TOS and the IP security options are copied from the payload protocol header to the delivery protocol header. The TTL however does not copy but establishes a default value used by the IP in order to prevent the RIP packets traveling through the tunnel timing out before reaching their destination.

2.2. IP over SRT with GRE

Again the encapsulation controlling protocol is GRE. In this case the payload protocol or protocol being encapsulated is SRT.

The GRE header fields are filled in and interpreted in the same way.

The TTL, TOS and the security options in the delivery protocol header are used by default in IP.

3. “Keepalive” maintenance packets

The IP Tunnel interface has a “keepalive” mechanism in order to monitor connectivity with the remote end of the tunnel. Through this mechanism the interface is only operative when real connectivity between the tunnel ends exists. In this way, alternative routes can be taken (backup routes) without needing to use routing protocols such as RIP or OSPF.

Connectivity monitoring is carried out by sending maintenance packets and checking that a response is received. The following sections describe the “keepalive” maintenance packets used in the IP Tunnel interfaces.

3.1. “Keepalive” petition packet

The packet sent by the device to determine connectivity with the remote end is made up of the following:

1. IP header with:

   Precedence (TOS field) = “Internetwork Control”

   Source address = Tunnel source address

   Destination address = Tunnel destination address
2. GRE header with the configured parameters, and the internal packet protocol = IP
3. IP header with:

   Precedence (TOS field) = “Internetwork Control”

   Source address = Tunnel destination address

   Destination address = Tunnel source address
4. GRE header with the configured parameters (with the exception of the sequence number), and the internal packet protocol = 0x0000.

3.2. “Keepalive” response packet

When a “keepalive” petition packet reaches the tunnel destination end (tunnel destination IP address) the corresponding device processes the frame and decapsulates the internal packet. This internal packet is returned to the tunnel source end using conventional routing.

In this way, the “keepalive” petition packet is a conventional IP packet encapsulated in GRE. Therefore the remote device will conventionally route this even though it does not have “keepalive” functionality.

The “keepalive” response packet can be distinguished as the internal packet protocol field (in the GRE header) has the value 0x0000. The complete format is as follows:

1. IP header with:

   Precedence (TOS field) = “Internetwork Control”

   Source address = Tunnel destination address

   Destination address = Tunnel source address
2. GRE header with the configured parameters (with the exception of the sequence number), and the internal packet protocol = 0x0000.

4. References

RFC-1701: Generic Routing Encapsulation (GRE), S. Hanks, Octubre-1994

RFC-1702: Generic Routing Encapsulation over IPv4 networks, S. Hanks, Octubre-1994

Chapter 2
IP tunnel interface configuration (TNIP)

1. Creating an IP tunnel Interface (TNIP)

In order to create an IP tunnel interface, you need to enter ***"ADD DEVICE tnip <tunnel identifier>"*** in the global configuration menu:

```
Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
Config>
```

To subsequently access the configuration, simply enter ***"NETWORK tnipX"***, where **X** represents the tunnel identifier:

```
Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
TNIP config>
```

The protocol supported over the TNIP interface is IP. In order to activate the IP over the TNIP interface you need to assign an IP address to the selected interface or configure it as an unnumbered interface.

Example with a known IP address:

```
*P 4
Config>PROTOCOL IP

-- Internet protocol user configuration --
IP config>ADDRESS tnip1 5.5.5.1 255.255.0.0
IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     172.16.200.15   255.255.255.0   NETWORK broadcast,  fill 0
 serial0/0                                       IP disabled on this ifc
 atm0/0                                          IP disabled on this ifc
 bri0/0                                          IP disabled on this ifc
 x25-node                                        IP disabled on this ifc
 tnip1           5.5.5.1         255.255.0.0     NETWORK broadcast,  fill 0
IP config>
```

Example with an unnumbered interface:

```
*P 4
Config>PROTOCOL IP

-- Internet protocol user configuration --
IP config>ADDRESS tnip1 unnumbered
IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     17.16.200.15    255.255.255.0   NETWORK broadcast,  fill 0
 serial0/0                                       IP disabled on this ifc
 atm0/0                                          IP disabled on this ifc
 bri0/0                                          IP disabled on this ifc
 x25-node                                        IP disabled on this ifc
 tnip1           unnumbered      0.0.0.0         NETWORK broadcast,  fill 0
IP config>
```

2. IP Tunnel interface configuration (TNIP)

The TNIP interface configuration commands are described in this chapter. In order to access the TNIP configuration environment, you need to enter ***"NETWORK <TNIP interface>"***:

```
Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
TNIP config>
```

The available commands are as follows:

Command	**Function**
DESTINATION	Configures the tunnel destination IP address.
DISABLE	Disables the tunnel interface.
ENABLE	Enables the tunnel interface.
ENCAPSULATION	Accesses the encapsulation protocol configuration menu.
KEEPALIVE	Enables the "keepalive" maintenance.
LIST	Displays the configured parameters.
MODE	Selects the encapsulation mode in the tunnel interface. (Encapsulation protocol).
QOS-PRE-CLASSIFY	Enables the pre-classification for BRS packets.
SOURCE	Configures the tunnel source IP address.

2.1. DESTINATION

Configures the IP tunnel destination IP address. This must coincide with the IP address configured as the tunnel source in the router at the other end. If the tunnel destination IP address does not coincide with the configured source address at the other end, the packets routed to this router will be discarded as not pertaining to the tunnel.

A route must exist towards this destination IP address or else the tunnel packets cannot be rerouted. As a precaution, this route must be a static route to avoid recursive problems in the routing tables as explained in chapter 1.

Example:

```
TNIP config>DESTINATION 66.187.232.56
TNIP config>
```

2.2. DISABLE

Disables the tunnel interface. By default the tunnel interface is disabled.

Example:

```
TNIP config>DISABLE
TNIP config>
```

2.3. ENABLE

Enables the tunnel interface. By default the tunnel interface is not active.

Example:

```
TNIP config>ENABLE
TNIP config>
```

2.4. ENCAPSULATION

Accesses the encapsulation protocol configuration. Currently, the only encapsulation protocol supported is GRE (Generic Routing Encapsulation).

Example:

```
TNIP config>ENCAPSULATION

-- GRE Configuration --
GRE config>
```

2.5. KEEPALIVE

Enables the "keepalive" maintenance for the IP Tunnel. This maintenance consists of the periodic sending of "keepalive" request packets. If these are not received within the configured time, tunnel connectivity loss is determined and the IP Tunnel interface is left inoperative (down state) until connectivity is reestablished.

The command format is "**KEEPALIVE** [<*period*> [<*attempts*>]]". These parameters are described below:

Parameter	**Description**
period	Number of seconds between successive keepalive request packet transmissions. This also acts as the maximum response time as only the responses to the last keepalive request packet sent are considered. The permitted range is from 1 to 32767 seconds. Default value is 10 seconds.
attempts	Number of consecutive keepalive request packets without receiving a response to determine if connectivity has been lost. The permitted range is from 1 to 255 transmissions without a response. Default is 3.

In order to disable the "keepalive" maintenance, use the "**NO KEEPALIVE**" command.

Example:

```
TNIP config>KEEPALIVE 30 5
TNIP config>
```

2.6. LIST

Displays the IP tunnel configuration.

Example:

```
TNIP config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 212.95.195.132, destination 66.187.232.56
QoS preclassify: disabled
Keepalive enabled with period 10, 3 retries
TNIP config>
```

Tunnel mode: indicates the type of encapsulation and the state (enabled/disabled).

Tunnel source / destination: tunnel source / destination IP addresses

QoS preclassify: indicates if the BRS pre-classification is enabled.

Keepalive: displays the "keepalive" maintenance configuration.

2.7. MODE

Selects the encapsulation mode. Currently, the only encapsulation protocol supported is GRE (Generic Routing Encapsulation).

Example:

```
TNIP config>MODE GRE
TNIP config>
```

2.8. QOS-PRE-CLASSIFY

Enables the BRS packet pre-classification. Enabling this option means that the packets reaching the tunnel are classified through BRS (please see the BRS manual, Dm715-I) before being encapsulated by the tunnel. This permits you to distinguish between the different types of IP traffic transmitted through the tunnel. If this option is disabled, the packets will be classified once they have been encapsulated, therefore all the traffic processed by the tunnel will have the same IP header (given by the tunnel) and will be all classified in the same BRS class.

To disable this parameter use ***"NO QOS-PRE-CLASSIFY"***.

Example:

```
TNIP config>QOS-PRE-CLASSIFY
TNIP config>
```

2.9. SOURCE

Configures the IP tunnel source IP address. This must coincide with the IP address of one of the router's configured interfaces (Ethernet, PPP, Loopback etc.) **except** that of the tunnel itself. Additionally this must coincide with the IP address configured as destination in the device at the other end of the tunnel.

If the source IP address does not coincide with any of the router's interfaces, the packets destined to this IP address will not be received by the router as its own and it will try to route them towards another device.

If the configured source IP address does not coincide with the destination address configured at the router's other end, the link will never exist.

If the tunnel source is a PPP interface that receives dynamical assignation of the IP address (please see the PPP Interface manual, Dm710-I), then you need to configure as IP tunnel source address "**0.0.0.0**".

Example:

```
TNIP config>SOURCE 212.95.195.132
TNIP config>
```

3. Configuring the GRE encapsulation protocol (Generic Routing Encapsulation)

The GRE encapsulation protocol configuration commands are described in this section. In order to access the GRE configuration environment you need to introduce the ***"ENCAPSULATION"*** command in the tunnel interface configuration menu (with the interface configured in GRE encapsulation mode).

```
Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
TNIP config>ENCAPSULATION

-- GRE Configuration --
GRE config>
```

The available commands are as follows:

Command	**Function**
CHECKSUM	Enables the end-to-end checksum (GRE).
CIPHER	Enables RC4 cipher in the GRE tunnel.
CIPHER-KEY	Configures the RC4 cipher key.
KEY	Configures the tunnel identifier.
LIST	Displays the configured parameters.
SEQUENCE-DATAGRAMS	Drops datagrams received out of order.

3.1. CHECKSUM

Enables the option to send checksum in the GRE packet. By default the tunnel does not guarantee the integrity of the packets. By enabling this option, the router sends the GRE packets with a checksum field. If a packet is received with checksum, the device always checks this discarding those packets whose checksum is invalid even if the device has this particular option disabled.

To disable checksum, use ***"NO CHECKSUM"***.

Example:

```
GRE config>CHECKSUM
GRE config>
```

3.2. CIPHER

Activates the RC4 cipher for those packets encapsulated in the GRE tunnel. By default cipher is not enabled.

Although the keepalive request packets are ciphered, the response packets are not as they are not encapsulated in the tunnel.

To disable the RC3 cipher, use ***"NO CIPHER"***.

Example:

```
GRE config>CIPHER
GRE config>
```

3.3. CIPHER-KEY

Configures the tunnel interface cipher key. This key admits a maximum of 32 alphanumerical characters.

To reestablish the cipher default key in the GRE tunnels, use ***"NO CIPHER-KEY"***.

Example:

```
GRE config>CIPHER-KEY thisIsAnExample
GRE config>
```

3.4. KEY

Enables the tunnel identifier check. On enabling this option, the device prompts for an identifier for the tunnel in question. This tunnel identifier **must be the same at both ends** of the tunnel. The identifier is a whole number between 0 and 4294967295 (32 bits). This option is disabled by default.

When the tunnel identifier is enabled, the router discards those packets containing a different identifier to that configured.

Example:

```
GRE config>KEY 5
GRE config>
```

3.5. LIST

Displays the GRE protocol configuration.

Example:

```
GRE config>LIST
RC4 Cipher.................: enabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [5]
Drop Out-of-Order Datagrams: disabled
GRE config>
```

RC4 Cipher: indicates if the RC4 cipher is enabled.

End-to-End Checksumming: indicates if the end-to-end checksum is enabled.

Tunnel identification key: tunnel identifier (if this is enabled).

Drop Out-of-Order Datagrams: drops datagrams received out of order.

3.6. SEQUENCE-DATAGRAMS

Enables the option to ensure order in the incoming datagrams. On enabling this option, the router checks the sequence number included in the GRE header and drops those packets which arrive out of order. By default this option is disabled in the GRE tunnel.

To disable the sequence number, use the ***"NO SEQUENCE-DATAGRAMS"*** command.

Example:

```
GRE config>SEQUENCE-DATAGRAMS
GRE config>
```

Chapter 3
Dynamic Tunnels (Internet)

1. Description

If we apply tunnel technology to the public networks and Internet, interconnection between the diverse local networks becomes possible in both an efficient and cost effective way. From tunnel technology seen in prior chapters this can be carried out, but we do come up against some problems which are dealt with in this chapter among which are the following:

1. In order to establish a tunnel between two points is imperative that both know the end IP address. This means that access is only possible with authenticated accesses and fixed IPs (this is a limited resource within Internet and expensive).
2. Connecting *n* local networks require you to configure and establish *n-1* tunnels for each one.

The solution to these problems can be found in a central device (with a fixed known IP address) which supports *n* tunnels and manages the inter-tunnel traffic thus automatically resolving the second problem. The solution to the first problem consists in giving this device the capability to adapt the configuration of its tunnels so it can support connections to devices whose IP address is different each time they connect. **This dynamic tunnel reconfiguration for each new connection is the reason why these tunnels are known as dynamic tunnels**.

From this point onwards, the central device will be referred to as the ISP router as its normal location is to be found in an Internet Server Provider, while the routers connecting to it will be referred to as client routers.

1.1. Scenarios/Presented problems

- **Scenario 1:** Access of local network devices to Internet for network services (html, ftp etc.) through a router.
- **Scenario 2**: Interconnection of remote local networks with ISP local network through routers with tunnels.
- **Scenario 3:** Interconnection between local networks and with the ISP local network through routers with tunnels.

If your objective is scenario 2/3, the configuration for the client routers is relatively simple as you can predict that any IP address which is not local is accessible through the tunnel. However if the intention is to simultaneously permit scenario 1 as well, the client routers must be able to distinguish if a specific destination address is accessible through the tunnel or out of it (Internet address). This means that the client routers must know which networks can be reached through the tunnel. The solution rests in configuring all the possible routes in all the clients or configure them in the ISP central router and that the ISP router informs the clients through a routing protocol (RIP).

Furthermore, the ISP device needs to know which networks are accessible through the client routers.

1.2. Types of tunnels

The following tunnel classification is carried out with the idea of analyzing the distinct behavior in each case:

- **Static**: These are Tunnels where the source and destination addresses are fixed. These tunnels have been previously dealt with.
- **Dynamic**: When one of the tunnel's addresses (source or destination) is unknown before making the connection and the device being connected to is unknown.
- **Semi-dynamic**: This deals with a special dynamic tunnel case where despite not knowing the address (source or destination) of the tunnel, we know the device being connected to, as the tunnel identifier (GRE *key* field) is unique.
- **Promiscuous**: This deals with a special case of static tunnel where you do not know the source or the destination addresses of the tunnel. The operates in a default tunnel interface mode, receiving traffic that is not destined to any other tunnel interface but not permitting traffic to be transmitted (encapsulate).

(Dynamic and semi-dynamic tunnels are normal in Internet when the address acquired by the remote device is not pre-assigned).

a) Dynamic Tunnels

These are the easiest tunnels to configure and at the same time the most flexible. These are more recommendable than the other types.

The ISP device offers *n* tunnels to the clients which these use as they connect: when a client stops sending information, the tunnel becomes free for another connection.

The strength of this configuration is based in the use of RIP, essential so that the client informs the ISP of the networks accessible through it and vice-versa.

b) Semi-dynamic Tunnels

These are an extension of the dynamic tunnels where the remote devices permitted to connect in each ISP tunnel are discriminated. In order to do this you configure a unique identifier in the tunnel which must coincide with that configured in the remote device (GRE *key* field). In reality this is like configuring a dynamic tunnel for each remote device.

These tunnels add two functions:

- Remote device identification.
- As the client that connects to each tunnel is already known, the routes can be aggregated and therefore it is unnecessary to enable RIP.

This means that these tunnels are only recommended when you wish to dispense with RIP or the security and access control are essential, as the configuration is more complex as identifiers must be associated with the remote devices.

c) Promiscuous Tunnels

This deals with a special case of static tunnels where you do not know the source or the destination addresses of the tunnel.

These tunnels receive all traffic that does not correspond to any other tunnel interface provided that the configuration of the tunnel identifier (GRE *key* field) corresponds to the received packet. However no type of traffic can be transmitted (encapsulate).

Through this IP tunnel interface configuration, traffic can be simultaneously received from many tunnels over a single interface, although transmission cannot be carried out as the interface is static and therefore cannot learn the tunnel addresses.

This type of tunnel is useful in specific cases as for example when you wish to learn the remote networks accessed through IPSEC through RIP.

1.3. The Importance of RIP

One of the most delicate aspects of these types of tunnels is the status control; from an *idle state* the status changes to *connected* with the *client* remote device requesting this. While the tunnel is being used, the status remains connected and once terminated, it should return to the initial idle state for future use.

One of the most critical aspects when establishing dynamic tunnels is the decision whether the tunnel remains in use or not as the remote device may have disconnected or been switched off without prior warning.* Dialogue *is therefore necessary between the routers maintaining the tunnels and RIP protocol is used for this.

If you wish to avoid tunnel reuse and exclusively reserve it to give service to a client device, you can identify it with a unique key.

Summary of the RIP features:

a) Controls disconnection in order to reuse the tunnels.
b) Gives information on the accessible network(s).

Problems that the RIP can present:

When the IP addresses at each end of the tunnel are distinct networks, this can give rise to a router receiving access information on the tunnel destination network through the tunnel interface itself therefore losing access to the remote end. This does not occur in Internet where the address acquired by the client belongs to the ISP device network however it does in all other cases.

The solution for this is to simply add a static route to access the destination provided this is previously known.

In any case, this situation is detected by the routers who in turn report the events and statistics.

2. User Scenarios

It is essential to define the use of the router before configuring the tunnels in it as in order to take maximum advantage of the router and the communications line depends on the configuration corresponding to the needs.

The most important decision is based on configuration of the static routes or whether to leave this to a routing protocol. This would make for easier configuration but would diminish performance, as some part of the bandwidth would have to be used to exchange messages between routers.

Although a general norm does not exist, this can be based on some guidelines that will help you to take the most adequate decision. It is therefore essential to define the scenario where the router is going to operate.

- **Scenario 2/3 (tunnel to interconnect local networks through Internet):** All the non-local addresses are accessible through the tunnel i.e. there are sufficient to configure this as a default route (with the exception of the tunnel remote end which must be configured as static).
- **Scenario 1+2/3 (surfing through Internet and interconnecting local networks through Internet via tunnels).** Non-local addresses are accessible through Internet, however there do exist dynamic or semi-dynamic tunnels through Internet configured in the client router for each remote LAN.

You also need to bear in mind some aspects seen in previous chapters:

- When you use *dynamic* tunnels, you need to configure RIP in order to reuse them.

2.1. Tunnel function without surfing through Internet (Scenarios 2/3)

The client configuration is based on defining the tunnel as the default route towards any destination (except local destinations or ISP). Here you may disable the RIP arriving from the ISP therefore improving the line performance due to the fact that RIP traffic flow is high if the ISP supports a large number of tunnels.

a) *Minimum configuration through RIP*

When employing dynamic tunnels (i.e. reusable) it is essential to use RIP in the *client ® ISP* direction in order to know the client's accessible networks.

Surfing Permitted:	No
Type of tunnel:	Dynamic
Default route:	Tunnel
RIP:	Client ⇒ ISP
Security:	Low
Difficulty level in configuration	Very low
Efficiency	High

b) More complex configuration reducing RIP traffic

Through dedicated tunnels for each remote device requiring access, the client is identified and RIP is not essential in *the client ® ISP* direction. On the other hand, you must maintain the identifiers when configuring the tunnels in ISP and clients.

Surfing Permitted:	No
Type of tunnel:	Semi-dynamic
Default route:	Tunnel
RIP:	No
Security:	High
Difficulty level in configuration	Average
Efficiency	Very high

2.2. Simultaneous Tunnel and Surfing (Scenarios 1 + 2/3)

In the clients, the default route accesses any Internet destination. Therefore any destinations in networks accessible through the tunnel must be known. This is achieved by static configuration in each of them or configured in the ISP which in turn informs the clients through RIP.

a) Maximum load in the network/Minimum configuration

In cases where there are not very many clients, the routing mechanism can be given to the RIP protocol thus avoiding the need to configure static routes in the clients.

Surfing Permitted:	No
Type of tunnel:	Dynamic
Default route:	Internet
RIP:	Clients ⇔ ISP
Security:	Low
Difficulty level of configuration	Low
Efficiency	Low if there are many tunnels

This does not exclude specific cases where it is not convenient to use RIP with a particular client as a unique *key* can be dedicated.

This scenario is very useful when you wish to interconnect just a few remote networks.

b) Minimum load in the network / More complex configuration

When the number of routes known to the ISP device is high (either due to a high number of tunnels or the remote local networks are complex), the RIP traffic can seriously affect the tunnel performance. In this case you need to evaluate the possibility of dispensing with the routing protocol in the *ISP ®*

Clients direction. This means you must statically configure the networks needing to be accessed through the tunnel in the clients.

Surfing Permitted:	Yes
Type of tunnel:	Dynamic
Default route:	Internet
RIP:	Clients ⇒ ISP
Security:	Average
Difficulty level of configuration	Average
Efficiency	High

As in the above cases, you can dispense with RIP in the *Clients®ISP* direction by dedicating a unique *key*.

This scenario is particularly useful when you wish to access remote local networks from the ISP and not the interconnection of remote local networks. E.g. when an entity has an ISP and wishes to access the branches from this.

c) Void overload in the network / More complex configuration/Client control

This scenario is totally based on the use of distinct *keys* so that each tunnel interface is perfectly defined. I.e. you know who the connected client is and the networks that can be accessed through this.

Surfing Permitted:	Yes
Type of tunnel:	Semi-Dynamic
Default route:	Internet
RIP:	No
Security:	High
Difficulty level for configuration	High
Efficiency	Maximum

This reduces overload to zero as the RIP traffic is totally eliminated by specifically configuring all the reachable routes.

Given the greater client control in this scenario, this is useful when the ISP offers network interconnection services to third parties.

IMPORTANT: When the service is offered to third parties, you must avoid duplicating addresses between the client installations (you cannot have the same subnet in distinct clients).

3. Security

When you are trying to connect local networks through a public network, the relative security aspects become very important. You must have authentication mechanisms for the connections and mechanisms to avoid undesired inter-tunnel traffic.

The first authentication mechanism is the use of fixed addresses although this is expensive in Internet. If this option cannot be used you need to resort to another mechanism such as GRE tunnel identifier (*key*) which must be known to both ends. Additionally it is also possible to cipher the content of the GRE packet.

As the inter-tunnel traffic is carried out through the ISP device, you have complete control over it.

Chapter 4
IP tunnel Interface Monitoring (TNIP)

1. IP Tunnel interface monitoring (TNIP)

In order to access the PPP interface monitoring menu, you need to introduce the ***"NETWORK <TNIP interface>"*** command from the general monitoring menu:

```
+NETWORK tnip1
TNIP protocol monitor
TNIP>
```

The available command is as follows:

LIST	Displays the monitoring information.

a) LIST

Displays the information relative to the tunnels.

· *LIST STATE*

Displays the state of the tunnel connection (this only applies to dynamic tunnels).

Example:

```
TNIP>LIST STATE

Int Source IP      Dest. IP        Start Time  Conections  Descon.err  Loop
--- ---------      ----------      ----------  ----------  ----------  ----
 7 200.200.200.1   200.200.200.200 00:29                1           0  0

TNIP>
```

Start Time: start time for the last connection.

Connections: number of connections since the last router startup.

Discon.err: number of disconnections due to not receiving routes through the said tunnel.

Loop: reports if there has been a loop.

2. IP Tunnel interface statistics (TNIP)

On executing the ***"DEVICE <TNIP interface>"*** command from the general monitoring process (+) prompt, all the interface statistics for the corresponding TNIP are displayed:

```
+DEVICE TNIP1

                                    Auto-test   Auto-test     Maintenance
Interface           CSR    Vect       valids    failures        failures
tnip1                 0       0            2           0               0
Imput Stats
-----------
  Frames ok    12980
  Frames error 0
  ---> Invalid encapsulation    0
  ---> Out-of-Order frames      0
  ---> Checsksum errors         0
  ---> Key errors               0
  ---> Unknown payload protocol 0
  ---> Error in cipher          0
  ---> Internal errors          0
Output Stats
------------
  Frames ok    11545
  Frames error 0
  ---> Invalid encapsulation    0
  ---> Unknown payload protocol 0
+
```

Chapter 5
IP tunnel configuration examples

1. IP tunnel over IP

1.1. Steps to follow at each end of the tunnel

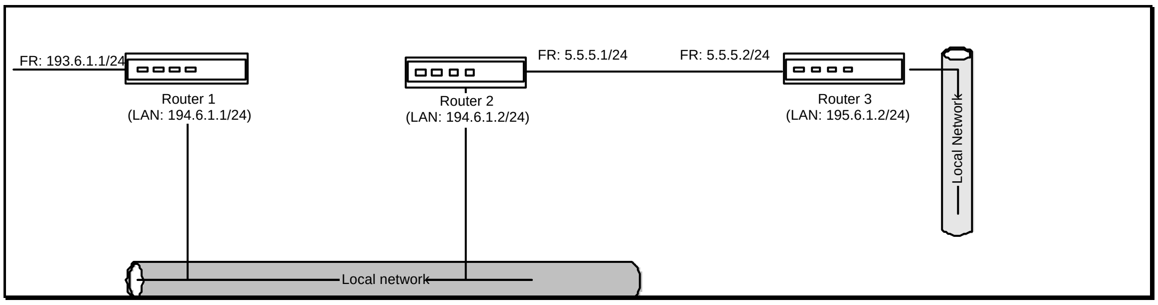
- Create the IP tunnel interface.
- Assign an IP address to the tunnel interface or configure it as unnumbered.
- Configure the tunnel source.
- Configure the tunnel destination. Aggregate the necessary IP route in order to reach this destination.
- Configure the encapsulation protocol which runs in the tunnel (or type of tunnel).
- Enable the desired options.
- Aggregate the IP routes for those networks that need to be accessed through the IP tunnel giving the IP tunnel interface itself as the next hop.
- Enable the tunnel, save and restart.

1.2. Steps to follow for those devices which use the tunnel

- Aggregate the necessary routes so the tunnel source and destination are accessible.

1.3. Example 1.a: IP over IP with GRE

Configuration of a tunnel with Router1 source and Router3 destination so networks 193.6.1.0/24 and 195.6.1.0/24 can communicate.



a) Router1 Configuration

Aggregate the Frame Relay interface and the IP tunnel.

```
*P 4
Config>SET HOSTNAME Router1
Router1 Config>SET DATA-LINK FRAME-RELAY serial0/0
Router1 Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
Router1 Config>
```

Configure the interface addresses.

```
Router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router1 IP config>ADDRESS ethernet0/0 194.6.1.1 255.255.255.0
Router1 IP config>ADDRESS serial0/0 193.6.1.1 255.255.255.0
Router1 IP config>ADDRESS tnip1 unnumbered
Router1 IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     194.6.1.1       255.255.255.0   NETWORK broadcast,  fill 0
 serial0/0       193.6.1.1       255.255.255.0   NETWORK broadcast,  fill 0
 serial0/1                                       IP disabled on this ifc
 serial0/2                                       IP disabled on this ifc
 bri0/0                                          IP disabled on this ifc
 x25-node                                        IP disabled on this ifc
 tnip1           unnumbered      0.0.0.0         NETWORK broadcast,  fill 0
Router1 IP config>EXIT
Router1 Config>
```

Subsequently, configure the IP tunnel.

```
Router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router1 TNIP config>SOURCE 194.6.1.1
Router1 TNIP config>DESTINATION 5.5.5.2
Router1 TNIP config>ENABLE
Router1 TNIP config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 194.6.1.1, destination 5.5.5.2
QoS preclassify: disabled
Router1 TNIP config>ENCAPSULATION

-- GRE Configuration --
Router1 GRE config>CHECKSUM
Router1 GRE config>KEY 1234
Router1 GRE config>LIST
RC4 Cipher.................: disabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [1234]
Drop Out-of-Order Datagrams: disabled
Router1 GRE config>EXIT
Router1 TNIP config>EXIT
Router1 Config>
```

Aggregate the necessary routes.

```
Router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router1 IP config>ROUTE 5.5.5.2 255.255.255.255 194.6.1.2 1
Router1 IP config>ROUTE 195.6.1.0 255.255.255.0 tnip1 1
Router1 IP config>EXIT
Router1 Config>
```

Once all the above configuration steps have been executed, you must save the configuration and restart the device.

b) Router2 Configuration

Aggregate the Frame Relay interface.

```
*P 4
Config>SET HOSTNAME Router2
Router2 Config>SET DATA-LINK FRAME-RELAY serial0/0
Router2 Config>
```

Configure the Frame Relay interface.

```
Router2 Config>NETWORK serial0/0

-- Frame Relay user configuration --
Router2 FR config>NO LMI
Router2 FR config>PVC 16 default
Router2 FR config>PVC 16 cir 64000
Router2 FR config>PVC 16 bc 16000
Router2 FR config>PROTOCOL-ADDRESS 5.5.5.2 16
Router2 FR config>EXIT
Router2 Config>
```

Configure the interface addresses.

```
Router2 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router2 IP config>ADDRESS ethernet0/0 194.6.1.2 255.255.255.0
Router2 IP config>ADDRESS serial0/0 5.5.5.1 255.255.255.0
Router2 IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     194.6.1.2       255.255.255.0   NETWORK broadcast,  fill 0
 serial0/0       5.5.5.1         255.255.255.0   NETWORK broadcast,  fill 0
 serial0/1                                       IP disabled on this ifc
 serial0/2                                       IP disabled on this ifc
 bri0/0                                          IP disabled on this ifc
 x25-node                                        IP disabled on this ifc
Router2 IP config>EXIT
Router2 Config>
```

Once all the above configuration steps have been executed, you must save the configuration and restart the device.

c) Router3 Configuration

Aggregate the Frame Relay interface and the IP tunnel.

```
*P 4
Config>SET HOSTNAME Router3
Router1 Config>SET DATA-LINK FRAME-RELAY serial0/0
Router1 Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
Router1 Config>
```

Configure the interface addresses.

```
Router3 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router3 IP config>ADDRESS ethernet0/0 195.6.1.1 255.255.255.0
Router3 IP config>ADDRESS serial0/0 5.5.5.2 255.255.255.0
Router3 IP config>ADDRESS tnip1 unnumbered
Router3 IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     195.6.1.1       255.255.255.0   NETWORK broadcast,  fill 0
 serial0/0       5.5.5.2         255.255.255.0   NETWORK broadcast,  fill 0
 serial0/1                                       IP disabled on this ifc
 serial0/2                                       IP disabled on this ifc
 bri0/0                                          IP disabled on this ifc
 x25-node                                        IP disabled on this ifc
 tnip1           unnumbered      0.0.0.0         NETWORK broadcast,  fill 0
Router3 IP config>EXIT
Router3 Config>
```

Then configure the Frame Relay interface

```
Router3 Config>NETWORK serial0/0

-- Frame Relay user configuration --
Router3 FR config>NO LMI
Router3 FR config>PVC 16 default
Router3 FR config>PVC 16 cir 64000
Router3 FR config>PVC 16 bc 16000
Router3 FR config>PROTOCOL-ADDRESS 5.5.5.1 16
Router3 FR config>EXIT
Router3 Config>
```

Next, configure the tunnel

```
Router3 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router3 TNIP config>ENABLE
Router3 TNIP config>DESTINATION 194.6.1.1
Router3 TNIP config>SOURCE 5.5.5.2
Router3 TNIP config>ENCAPSULATION

-- GRE Configuration --
Router3 GRE config>CHECKSUM
Router3 GRE config>KEY 1234
Router3 GRE config>EXIT
Router3 TNIP config>EXIT
Router3 Config>
```

Aggregate the necessary routes

```
Router3 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router3 IP config>ROUTE 194.6.1.1 255.255.255.255 5.5.5.1 1
Router3 IP config>ROUTE 193.6.1.0 255.255.255.0 tnip1 1
Router3 IP config>EXIT
Router3 Config>
```

You now need to save the configuration and restart the device.

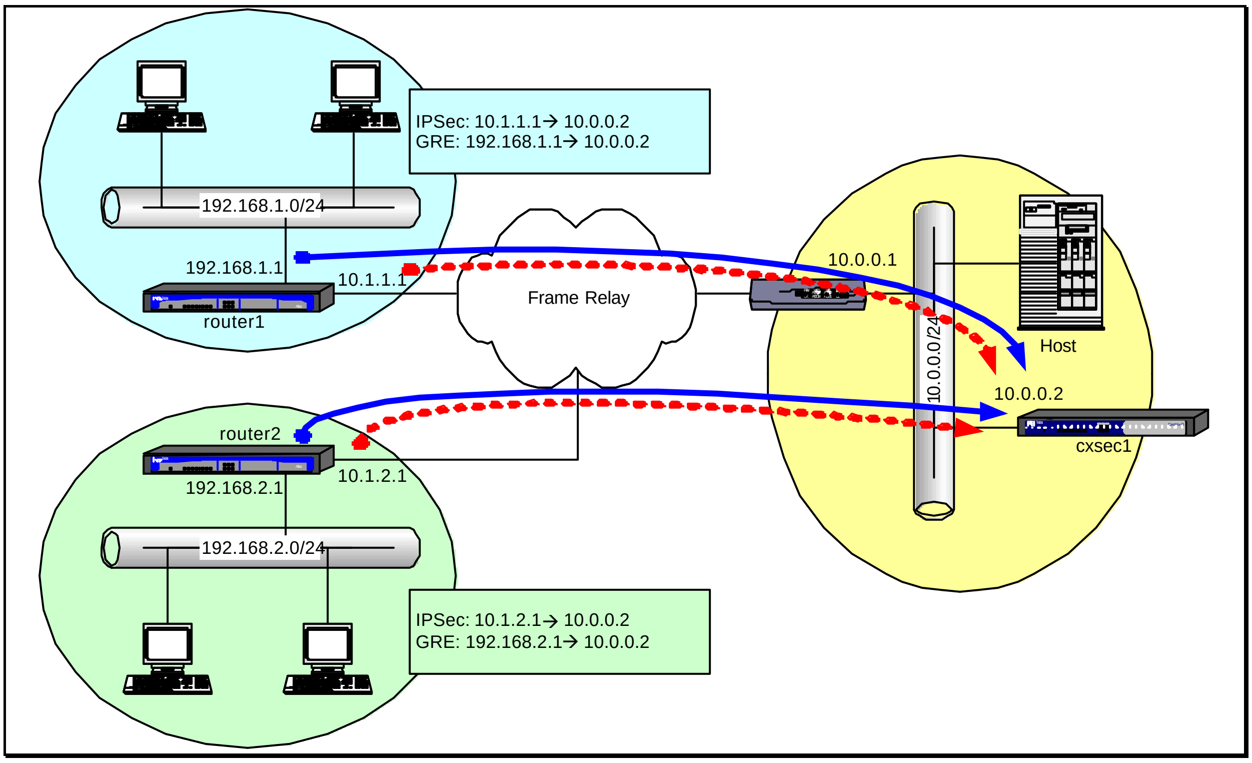
1.4. Example 1.b: promiscuous tunnel

Configuration of a tunnel with source ***ROUTER1*** and destination ***CXSEC1***, to send a branch RIP to the Centrix-Sec.

This scenario arises when you wish to transmit cipher traffic with IPSec and problems arise for sending RIP through the IPSec tunnel. One solution consists of configuring a GRE tunnel where RIP sending is enabled and the encapsulated traffic is sent through the IPSec tunnel so that the device at the other end (possibly a Centrix-Sec) receives both the ciphered data and the accessible networks through the said tunnel.

The problem occurs when the device receiving RIP information provides service to many IPSec tunnels in which case a TNIP interface will be required for each one. In order to avoid this, you can configure a single TNIP interface in promiscuous mode so that the same interface receives the encapsulated RIP traffic from all the tunnels.

The scheme for this example is as follows:



Promiscuous tunnels application example scenario.

a) CXSEC1 Configuration

Add the IP tunnel interface:

```
*PROCESS 4
Config>SET HOSTNAME cxsec1
cxsec1 Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
cxsec1 Config>
```

Configure the interface addresses:

```
cxsec1 Config>PROTOCOL IP

-- Internet protocol user configuration --
cxsec1 IP config>ADDRESS ethernet0/0 10.0.0.2 255.255.255.0
cxsec1 IP config>ADDRESS tnip1 unnumbered 0.0.0.0
cxsec1 IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     10.0.0.2        255.255.255.0    NETWORK broadcast,  fill 0
 serial0/0                                        IP disabled on this ifc
 serial0/1                                        IP disabled on this ifc
 serial0/2                                        IP disabled on this ifc
 bri0/0                                           IP disabled on this ifc
 x25-node                                         IP disabled on this ifc
 tnip1           unnumbered      0.0.0.0          NETWORK broadcast,  fill 0
cxsec1 IP config>EXIT
cxsec1 Config>
```

As this is dealing with a promiscuous tunnel, the source and the destination address are not configured. No additional function is used in this example, so all that is required is to enable the interface:

```
cxsec1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
cxsec1 TNIP config>ENABLE
cxsec1 TNIP config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 0.0.0.0, destination 0.0.0.0
QoS preclassify: disabled
cxsec1 TNIP config>ENCAPSULATION

-- GRE Configuration --
cxsec1 GRE config>LIST
RC4 Cipher.................: disabled
End-to-End Checksumming....: disabled
Tunnel identification key..: disabled
Drop Out-of-Order Datagrams: disabled
cxsec1 GRE config>EXIT
cxsec1 TNIP config>EXIT
cxsec1 Config>
```

Add the necessary routes:

```
cxsec1 Config>PROTOCOL IP

-- Internet protocol user configuration --
cxsec1 IP config>ROUTE 0.0.0.0 0.0.0.0 10.0.0.1 1
cxsec1 IP config>LIST ROUTES

route to 0.0.0.0,0.0.0.0 via 10.0.0.1, cost 1
cxsec1 IP config>EXIT
cxsec1 Config>
```

And configure the RIP protocol so this can receive routing information through the TNIP interface and send it to the local network.

Lastly, configure the IPSec cipher:

```
cxsec1 Config>FEATURE ACCESS-LISTS

-- Access Lists user configuration --
cxsec1 Access Lists config>ACCESS-LIST 100


cxsec1 Extended Access List 100>ENTRY 1 default
cxsec1 Extended Access List 100>ENTRY 1 permit
cxsec1 Extended Access List 100>ENTRY 1 source address 10.0.0.0 255.255.255.0
cxsec1 Extended Access List 100>ENTRY 1 destination address 192.168.0.0 255.255.0.0
cxsec1 Extended Access List 100>EXIT
cxsec1 Access Lists config>EXIT
cxsec1 Config>PROTOCOL IP

-- Internet protocol user configuration --
cxsec1 IP config>IPSEC

-- IPSec user configuration --
cxsec1 IPSec config>ENABLE
cxsec1 IPSec config>ASSIGN-ACCESS-LIST 100
cxsec1 IPSec config>TEMPLATE 1 default
cxsec1 IPSec config>TEMPLATE 1 isakmp des md5
cxsec1 IPSec config>TEMPLATE 1 ike mode aggressive
cxsec1 IPSec config>TEMPLATE 2 default
cxsec1 IPSec config>TEMPLATE 2 dynamic esp des md5
cxsec1 IPSec config>TEMPLATE 2 source-address 10.0.0.2
cxsec1 IPSec config>MAP-TEMPLATE 100 2
cxsec1 IPSec config>KEY preshared hostname router* plain 0x112233445566
```

```
cxsec1 IPSec config>EXIT
cxsec1 IP config>EXIT
cxsec1 Config>
```

Once all the above configuration steps have been executed, you must save the configuration and restart the device.

b) ROUTER1 Configuration

Add the Frame Relay interface and the IP tunnel interface:

```
*PROCESS 4
Config>SET HOSTNAME router1
router1 Config>SET DATA-LINK FRAME-RELAY serial0/0
router1 Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
router1 Config>
```

Configure the interface addresses and the internal address:

```
router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
router1 IP config>ADDRESS ethernet0/0 192.168.1.1 255.255.255.0
router1 IP config>ADDRESS serial0/0 10.1.1.1 255.255.255.252
router1 IP config>ADDRESS tnip1 unnumbered
router1 IP config>INTERNAL-IP-ADDRESS 192.168.1.1
router1 IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     192.168.1.1     255.255.255.0   NETWORK broadcast,  fill 0
 serial0/0       10.1.1.1        255.255.255.252 NETWORK broadcast,  fill 0
 serial0/1                                       IP disabled on this ifc
 serial0/2                                       IP disabled on this ifc
 bri0/0                                          IP disabled on this ifc
 x25-node                                        IP disabled on this ifc
 tnip1           unnumbered      0.0.0.0         NETWORK broadcast,  fill 0
Internal IP address: 192.168.1.1
router1 IP config>EXIT
router1 Config>
```

Configure the tunnel source and destination addresses and enable the interface:

```
router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
router1 TNIP config>ENABLE
router1 TNIP config>DESTINATION 10.0.0.2
router1 TNIP config>SOURCE 192.168.1.1
router1 TNIP config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 192.168.1.1, destination 10.0.0.2
QoS preclassify: disabled
router1 TNIP config>EXIT
router1 Config>
```

Add the necessary routes:

```
router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
router1 IP config>ROUTE 10.0.0.0 255.255.255.0 10.1.1.2 1
router1 IP config>LIST ROUTES

route to 10.0.0.0,255.255.255.0 via 10.1.1.2, cost 1
router1 IP config>EXIT
router1 Config>
```

And configure the RIP protocol so this sends information on the local network, which is accessible through IPSec, through the tunnel:

```
router1 Config>FEATURE ACCESS-LISTS

-- Access Lists user configuration --
router1 Access Lists config>ACCESS-LIST 1


router1 Standard Access List 1>ENTRY 1 default
router1 Standard Access List 1>ENTRY 1 permit
router1 Standard Access List 1>ENTRY 1 source address 192.168.1.0 255.255.255.0
router1 Standard Access List 1>EXIT
router1 Access Lists config>EXIT
router1 Config>PROTOCOL RIP

-- RIP protocol user configuration --
router1 RIP config>ENABLE
router1 RIP config>COMPATIBILITY 192.168.1.1 send none
router1 RIP config>COMPATIBILITY 192.168.1.1 receive none
router1 RIP config>COMPATIBILITY 10.1.1.1 send none
router1 RIP config>COMPATIBILITY 10.1.1.1 receive none
router1 RIP config>COMPATIBILITY tnip1 receive none
router1 RIP config>SENDING tnip1 distribute-list 1
router1 RIP config>EXIT
router1 Config>
```

Finally, configure the Frame Relay interface and the IPSec cipher:

```
router1 Config>NETWORK serial0/0

-- Frame Relay user configuration --
router1 FR config>PVC 21 default
router1 FR config>PROTOCOL-ADDRESS 10.1.1.2 21
router1 FR config>EXIT
router1 Config>FEATURE ACCESS-LISTS

-- Access Lists user configuration --
router1 Access Lists config>ACCESS-LIST 100


router1 Extended Access List 100>ENTRY 1 default
router1 Extended Access List 100>ENTRY 1 permit
router1 Extended Access List 100>ENTRY 1 source address 192.168.1.0 255.255.255.0
router1 Extended Access List 100>ENTRY 1 destination address 10.0.0.0 255.255.255.0
router1 Extended Access List 100>EXIT
router1 Access Lists config>EXIT
router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
router1 IP config>IPSEC

-- IPSec user configuration --
router1 IPSec config>ENABLE
router1 IPSec config>ASSIGN-ACCESS-LIST 100
router1 IPSec config>TEMPLATE 1 default
router1 IPSec config>TEMPLATE 1 isakmp des md5
```

```
router1 IPSec config>TEMPLATE 1 destination-address 10.0.0.2
router1 IPSec config>TEMPLATE 1 ike mode aggressive
router1 IPSec config>TEMPLATE 1 ike idtype fqdn
router1 IPSec config>TEMPLATE 1 keepalive dpd
router1 IPSec config>TEMPLATE 2 default
router1 IPSec config>TEMPLATE 2 dynamic esp des md5
router1 IPSec config>TEMPLATE 2 source-address 10.1.1.1
router1 IPSec config>TEMPLATE 2 destination-address 10.0.0.2
router1 IPSec config>MAP-TEMPLATE 100 2
router1 IPSec config>KEY preshared ip 10.0.0.2 plain 0x112233445566

router1 IPSec config>EXIT
router1 IP config>EXIT
router1 Config>
```

Once all the above configuration steps have been executed, you must save the configuration and restart the device.

c) *ROUTER2 Configuration*

The configuration of router2 is similar to that of router1. The principal difference lies in the local IP addresses:

```
*PROCESS 4
Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
Config>SET DATA-LINK FRAME-RELAY serial0/0
Config>SET HOSTNAME router2
router2 Config>NETWORK serial0/0

-- Frame Relay user configuration --
router2 FR config>PVC 22 default
router2 FR config>PROTOCOL-ADDRESS 10.1.2.2 22
router2 FR config>EXIT
router2 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
router2 TNIP config>ENABLE
router2 TNIP config>DESTINATION 10.0.0.2
router2 TNIP config>SOURCE 192.168.2.1
router2 TNIP config>EXIT
router2 Config>FEATURE ACCESS-LISTS

-- Access Lists user configuration --
router2 Access Lists config>ACCESS-LIST 1


router2 Standard Access List 1>ENTRY 1 default
router2 Standard Access List 1>ENTRY 1 permit
router2 Standard Access List 1>ENTRY 1 source address 192.168.2.0 255.255.255.0
router2 Standard Access List 1>EXIT
router2 Access Lists config>ACCESS-LIST 100


router2 Extended Access List 100>ENTRY 1 default
router2 Extended Access List 100>ENTRY 1 permit
router2 Extended Access List 100>ENTRY 1 source address 192.168.2.0 255.255.255.0
router2 Extended Access List 100>ENTRY 1 destination address 10.0.0.0 255.255.255.0
router2 Extended Access List 100>EXIT
router2 Access Lists config>EXIT
router2 Config>PROTOCOL IP

-- Internet protocol user configuration --
router2 IP config>INTERNAL-IP-ADDRESS 192.168.2.1
router2 IP config>ADDRESS ethernet0/0 192.168.2.1 255.255.255.0
router2 IP config>ADDRESS serial0/0 10.1.2.1 255.255.255.252
router2 IP config>ADDRESS tnip1 unnumbered 0.0.0.0
router2 IP config>ROUTE 10.0.0.0 255.255.255.0 10.1.2.2 1
```

```
router2 IP config>IPSEC

-- IPSec user configuration --
router2 IPSec config>ENABLE
router2 IPSec config>ASSIGN-ACCESS-LIST 100
router2 IPSec config>TEMPLATE 1 default
router2 IPSec config>TEMPLATE 1 isakmp des md5
router2 IPSec config>TEMPLATE 1 destination-address 10.0.0.2
router2 IPSec config>TEMPLATE 1 ike mode aggressive
router2 IPSec config>TEMPLATE 1 ike idtype fqdn
router2 IPSec config>TEMPLATE 1 keepalive dpd
router2 IPSec config>TEMPLATE 2 default
router2 IPSec config>TEMPLATE 2 dynamic esp des md5
router2 IPSec config>TEMPLATE 2 source-address 10.1.2.1
router2 IPSec config>TEMPLATE 2 destination-address 10.0.0.2
router2 IPSec config>MAP-TEMPLATE 100 2
router2 IPSec config>KEY preshared ip 10.0.0.2 plain 0x112233445566

router2 IPSec config>EXIT
router2 IP config>EXIT
router2 Config>PROTOCOL RIP

-- RIP protocol user configuration --
router2 RIP config>ENABLE
router2 RIP config>COMPATIBILITY 192.168.2.1 send none
router2 RIP config>COMPATIBILITY 192.168.2.1 receive none
router2 RIP config>COMPATIBILITY 10.1.2.1 send none
router2 RIP config>COMPATIBILITY 10.1.2.1 receive none
router2 RIP config>COMPATIBILITY tnip1 receive none
router2 RIP config>SENDING tnip1 distribute-list 1
router2 RIP config>EXIT
router2 Config>
```

Now you need to save the configuration and restart the device:

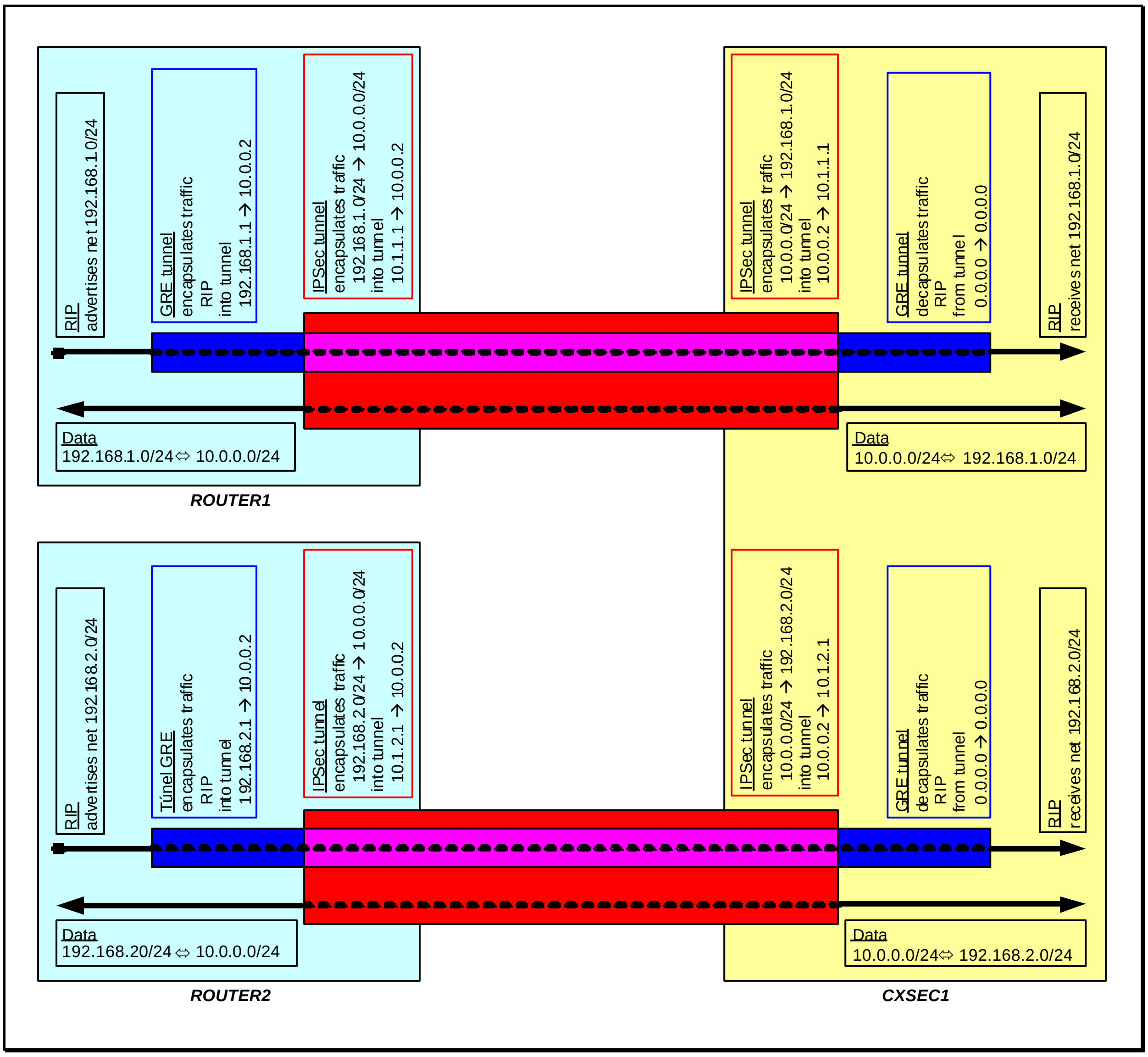
d) Final result

The final result is that both ***ROUTER1*** and ***ROUTER2*** send information on their local networks (192.168.1.0/24 and 192.168.2.0/24 respectively) through RIP, encapsulated in GRE and ciphered with IPSec.

CXSEC1 receives all the RIP information through the same tunnel interface (in promiscuous mode) and notifies its local network 10.0.0.0/24.

In this way, all devices pertaining to network 10.0.0.0/24 know that in order to send traffic to the ***ROUTER1*** and ***ROUTER2*** networks, this must be sent through ***CXSEC1***, which in turn sends this ciphered through the corresponding IPSec tunnel.

The following illustration represents the data and RIP information flow between the ***ROUTER1, ROUTER2*** and ***CXSEC1*** in their various encapsulations:


RIP advertises net 192.168.1.0/24
GRE tunnel encapsulates traffic RIP into tunnel 192.168.1.1 → 10.0.0.2
IPSec tunnel encapsulates traffic 192.168.1.0/24 → 10.0.0.0/24 into tunnel 10.1.1.1 → 10.0.0.2
IPSec tunnel encapsulates traffic 10.0.0.0/24 → 192.168.1.0/24 into tunnel 10.0.0.2 → 10.1.1.1
GRE tunnel decapsulates traffic RIP from tunnel 0.0.0.0 → 0.0.0.0
RIP receives net 192.168.1.0/24
Data 192.168.1.0/24⇔ 10.0.0.0/24
Data 10.0.0.0/24⇔ 192.168.1.0/24
ROUTER1
RIP advertises net 192.168.2.0/24
Túnel GRE encapsulates trafic RIP into tunnel 192.168.2.1 → 10.0.0.2
IPSec tunnel encapsulates trafic 192.168.2.0/24 → 10.0.0.0/24 into tunnel 10.1.2.1 → 10.0.0.2
IPSec tunnel encapsulates traffic 10.0.0.0/24 → 192.168.2.0/24 into tunnel 10.0.0.2 → 10.1.2.1
GRE tunnel decapsulates trafic RIP from tunnel 0.0.0.0 → 0.0.0.0
RIP receives net 192.168.2.0/24
Data 192.168.20/24 ⇔ 10.0.0.0/24
Data 10.0.0.0/24⇔ 192.168.2.0/24
ROUTER2
CXSEC1

Distinct traffic encapsulation in this scenario.

As you can see in the scheme, the RIP traffic from the ***ROUTER1*** TNIP interface (upper black arrow) is one direction only, from ***ROUTER1*** to ***CXSEC1***. This traffic is encapsulated in the GRE tunnel (illustrated in blue) as having source 192.168.2.1 and destination 10.0.0.2, and in turn will be encapsulated in the IPSec tunnel (illustrated in red).

The ***CXSEC1*** device receives traffic through the IPSec tunnel (red) and deciphers and de-encapsulates it obtaining GRE traffic (blue). As this traffic is destined to ***CXSEC1*** (address 10.0.0.2), it processes it in the TNIP interface, in this way obtaining the RIP information sent by ***ROUTER1.***

The rest of the traffic between networks 192.168.1.0/24 and 10.0.0.0/24 is directly encapsulated in the IPSec tunnel in both directions, as can be seen in the illustration (lower arrow).

Thanks to the TNIP interface promiscuous mode, the ***CXSEC1*** device is capable of receiving RIP information from all the branches through the same interface. In this way, you are not limited by the maximum of 15 TNIP interfaces and thus can provide service to an elevated number of branches.

2. IP tunnel over SRT

2.1. Steps to follow at each end of the tunnel

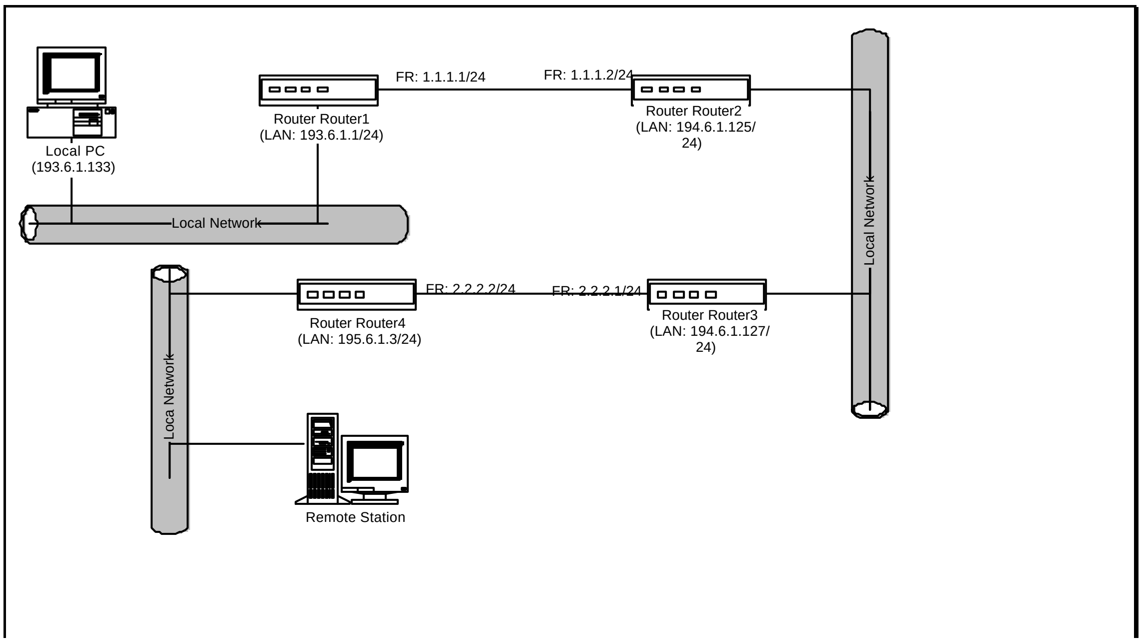
- Create the IP tunnel interface.
- Enable the bridge.
- Aggregate a port in the bridge for the tunnel interface.
- Configure the tunnel source.
- Configure tunnel destination. Aggregate the necessary IP route in order to reach the destination.
- Configure the encapsulation protocol which will run in the tunnel (or type of tunnel).
- Enable required options.
- Enable tunnel, save and restart.

2.2. Steps to follow for those devices which use the tunnel

- Aggregate the necessary routes so the tunnel source and destination are accessible.

2.3. Example: IP over SRT with GRE

Configure a tunnel with Router1 source and Router4 destination so that the networks 193.6.1.0/24 and 195.6.1.0/24 can communicate through NetBEUI traffic. To achieve this, you need to establish an IP tunnel over SRT between them both.



a) Router1 Configuration

Similarly to the previous example, the FR interface and the IP tunnel interface (TNIP) must be added and the IP addresses for the interfaces must be configured.

```
*P 4
Config>SET HOSTNAME Router1
Router1 Config>SET DATA-LINK FRAME-RELAY serial0/0
Router1 Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
Router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router1 IP config>ADDRESS ethernet0/0 193.6.1.133 255.255.255.0
Router1 IP config>ADDRESS serial0/0 1.1.1.1 255.255.255.0
Router1 IP config>ADDRESS tnip1 unnumbered
Router1 IP config>EXIT
Router1 Config>
```

Following that, configure the tunnel

```
Router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router1 TNIP config>ENABLE
Router1 TNIP config>DESTINATION 2.2.2.2
Router1 TNIP config>SOURCE 1.1.1.1
Router1 TNIP config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 1.1.1.1, destination 2.2.2.2
QoS preclassify: disabled
Router1 TNIP config>ENCAPSULATION

-- GRE Configuration --
Router1 GRE config>CHECKSUM
Router1 GRE config>KEY 1234
Router1 GRE config>SEQUENCE-DATAGRAM
Router1 GRE config>LIST
RC4 Cipher.................: disabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [1234]
Drop Out-of-Order Datagrams: enabled
Router1 GRE config>EXIT
Router1 TNIP config>EXIT
Router1 Config>
```

Next, configure the Frame Relay interface

```
Router1 Config>NETWORK serial0/0

-- Frame Relay user configuration --
Router1 FR config>PVC 16 default
Router1 FR config>PVC 16 CIR 64000
Router1 FR config>PROTOCOL-ADDRESS 1.1.1.2 16
Router1 FR config>NO LMI
Router1 FR config>EXIT
Router1 Config>
```

The necessary routes are included

```
Router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router1 IP config>ROUTE 2.2.2.0 255.255.255.0 1.1.1.2 1
Router1 IP config>EXIT
Router1 Config>
```

Lastly, configure the bridge.

```
Router1 Config>PROTOCOL ASRT

-- ASRT Bridge user configuration --
Router1 ASRT config>BRIDGE
Router1 ASRT config>PORT ethernet0/0 1
Router1 ASRT config>PORT tnip1 2
Router1 ASRT config>LIST BRIDGE

                 Source Routing Transparent Bridge Configuration
              ====================================================

Bridge:    Enabled                               Bridge behavior: STB
                   +------------------------------------------+
-------------------|       SOURCE ROUTING INFORMATION         |----------------
                   +------------------------------------------+
Bridge Number:          00                       Segments:          0
Max ARE Hop Cnt:        00                       Max STE Hop cnt:  00
1:N SRB:                Not Active               Internal Segment:  0x000
LF-bit interpret:       Extended
                   +------------------------------------------+
-------------------|             SR-TB INFORMATION            |----------------
                   +------------------------------------------+
SR-TB Conversion:        Disabled
TB-Virtual Segment:      0x000                   MTU of TB-Domain:  0
                   +------------------------------------------+
-------------------|    SPANNING TREE PROTOCOL INFORMATION    |----------------
                   +------------------------------------------+
Bridge Address:         Default                Bridge Priority:   32768/0x8000
STP Participation:       Disabled
                   +------------------------------------------+
-------------------|        TRANSLATION INFORMATION           |----------------
                   +------------------------------------------+
FA<=>GA Conversion:      Enabled                 UB-Encapsulation:  Disabled
DLS for the bridge:      Disabled
                   +-----------------------------------------+
-------------------|             PORT INFORMATION            |-----------------
                   +-----------------------------------------+
Number of ports added: 2
Port:   1    Interface:     ethernet0/0 Behavior:    STB Only   STP: Enabled

Port:   2    Interface:           tnip1 Behavior:    STB Only   STP: Enabled


Router1 ASRT config>EXIT
Router1 Config>
```

Now you need to save the configuration and restart the device.

b) Router2 and Router 3 Configuration

These are assumed to be correctly configured in order to provide IP connectivity.

c) Router4 Configuration

The Frame Relay interface and the IP tunnel interface (TNIP) are added in a similar way to the Router1 example given above.

```
*P 4
Config>SET HOSTNAME Router4
Router4 Config>ADD DEVICE tnip 1
Added TNIP interface tnip1
Router4 Config>SET DATA-LINK FRAME-RELAY serial0/0
Router4 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router4 IP config>ADDRESS ethernet0/0 195.6.1.3 255.255.255.0
Router4 IP config>ADDRESS serial0/0 2.2.2.2 255.255.255.0
Router4 IP config>ADDRESS tnip1 unnumbered
```

```
Router4 IP config>LIST ADDRESSES
IP addresses for each interface:
 ethernet0/0     195.6.1.3       255.255.255.0   NETWORK broadcast,  fill 0
 serial0/0       2.2.2.2         255.255.255.0   NETWORK broadcast,  fill 0
 serial0/1                                       IP disabled on this ifc
 serial0/2                                       IP disabled on this ifc
 bri0/0                                          IP disabled on this ifc
 x25-node                                        IP disabled on this ifc
 tnip1           unnumbered      0.0.0.0         NETWORK broadcast,  fill 0
Router4 IP config>EXIT
Router4 Config>
```

Following that, configure the tunnel

```
Router4 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router4 TNIP config>ENABLE
Router4 TNIP config>DESTINATION 1.1.1.1
Router4 TNIP config>SOURCE 2.2.2.2
Router4 TNIP config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 2.2.2.2, destination 1.1.1.1
QoS preclassify: disabled
Router4 TNIP config>ENCAPSULATION

-- GRE Configuration --
Router4 GRE config>CHECKSUM
Router4 GRE config>KEY 1234
Router4 GRE config>SEQUENCE-DATAGRAM
Router4 GRE config>LIST
RC4 Cipher.................: disabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [1234]
Drop Out-of-Order Datagrams: enabled
Router4 GRE config>EXIT
Router4 TNIP config>EXIT
Router4 Config>
```

Then configure the Frame Relay interface

```
Router4 Config>NETWORK serial0/0

-- Frame Relay user configuration --
Router4 FR config>PVC 16 default
Router4 FR config>PVC 16 CIR 64000
Router4 FR config>PROTOCOL-ADDRESS 2.2.2.1 16
Router4 FR config>NO LMI
Router4 FR config>EXIT
Router4 Config>
```

Include the necessary routes

```
Router4 Config>PROTOCOL IP

-- Internet protocol user configuration --
Router4 IP config>ROUTE 1.1.1.0 255.255.255.0 2.2.2.1 1
Router4 IP config>EXIT
Router4 Config>
```

Lastly configure the bridge.

```
Router4 Config>PROTOCOL ASRT

-- ASRT Bridge user configuration --
Router4 ASRT config>BRIDGE
Router4 ASRT config>PORT ethernet0/0 1
```

```
Router4 ASRT config>PORT tnip1 2
Router4 ASRT config>NO STP
Router4 ASRT config>EXIT
Router4 Config>
```

Finally, save the configuration and restart the device.

Chapter 6
IP Tunnel interface Events (TNIP)

1. IP Tunnel interface events monitoring (TNIP)

This permits real time monitoring of the events which occur over one or various TNIP interfaces when the event logging system is enabled for this protocol.

This is enabled from the configuration menu as shown below:

```
*P 4
Config>EVENT

-- ELS Config --
ELS config>ENABLE TRACE SUBSYSTEM TNIP ALL
ELS config>EXIT
Config>save
Save configuration [n]? yes

Saving configuration...OK on Flash as XXXXX.CFG
Config>                                              press <ctrl-p>
*RESTART
Are you sure to restart the system(Yes/No)? yes
```

In the same way, these can be enabled from the monitoring menu at any time without needing to store this in the configuration as shown below:

```
*P 3
Console Operator
+EVENT

-- ELS Monitor --
ELS>ENABLE TRACE SUBSYSTEM TNIP ALL
ELS>EXIT
+
```

The available events list for the TNIP protocol is as follows:

TNIP.001

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.001 Pack rec *str_enc_protocol*,ext prt 0x*num_external_protocol*, (*source_ip_address->destination_ip_address*), int *interface*

Long Syntax:

TNIP.001 Packet received with encapsulation *str_enc_protocol*, external protocol 0x*num_external_protocol*, (source *source_ip_address* y destination *destination_ip_address*), on interface *interface*

Description:

An encapsulated packet for an external protocol was received on an interface to decapsulate.

TNIP.002

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.002 Pack rec *str_enc_protocol*, ext prt 0x*num_esternal_protocol*, (*source_ip_address->destination_ip_address*), no tunnel

Long Syntax:

TNIP.002 Received packet with encapsulation *str_enc_protocol*, external protocol 0x*num_esternal_protocol*, (source *source_ip_address* y destination *destination_ip_address*) There is no tunnel for it.

Description:

A packet for an external protocol and with a certain encapsulation was received. There is no tunnel between the encapsulated packet source and destination address.

Cause:

An external device is sending packets to the router but these packets are discarded because there is no tunnel interface between its source and destination address

Action:

Change tunnels configuration in order to accept those packets. Identify the external device and configure it to not send those packets.

TNIP.003

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.003 Incom pack disc no act int, int *interface*

Long Syntax:

TNIP.003 Incoming packet discarded. The interface is down. Interface *interface*

Description:

The tunnel interface is down so the interface entry function has discarded the packet

TNIP.004

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.004 Incom pack disc no conf int, int *interface*

Long Syntax:

TNIP.004 Incoming packet discarded. The interface has no configuration. Interface *interface*

Description:

The tunnel interface has no configuration so the packet is discarded.

TNIP.005

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.005 Incom pack disc encapsul prot reject, int *interface*

Long Syntax:

TNIP.005 Incoming packet has been discarded because the interface configuration has a different encapsulation protocol, interface *interface*

Description:

The interface entry function has discarded the packet because the interface configuration has a different encapsulation protocol

Cause:

Bad interface configuration in one tunnel end.

Action:

Set the same encapsulation protocol in both tunnel ends.

TNIP.006

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.006 Err GRE header flag routing act, int *interface*

Long Syntax:

TNIP.006 Error in GRE header. Flags have the routing option, interface *interface*

Description:

The packet has active the routing option so the GRE desencapsulation function has discarded it

Cause:

The other end tunnel is sending GRE packets with routing field. This packet type is rejected

Action:

Configure the other tunnel end without GRE routing.

TNIP.007

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.007 Err GRE header chksm 0x*checksum* (exp 0x*expected_checksum*), int *interface*

Long Syntax:

TNIP.007 Error in GRE header, checksum 0x*checksum* (des 0x*expected_checksum*), interface *interface*

Description:

This message is generated when a packet has an invalid checksum. The received checksum, together with the correct checksum, are displayed.

Cause:

Most likely, this is a damaged packet. It may be that another node is building an incorrect header.

Action:

If the problem persists, examine a line trace to determine where the packet is being damaged.

TNIP.008

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.008 Dsc GRE pack key *key* (exp *expected_key*), int *interface*

Long Syntax:

TNIP.008 Packet GRE discarded, key *key* (expected *expected_key*), interface *interface*

Description:

This message is generated when the key packet is invalid. The received key, together with the correct key, is displayed.

Cause:

It may be that another node has a configuration with an invalid key.

Action:

Set the key configuration properly

TNIP.009

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.009 DSC GRE pack with no key (exp *expected_key*), int *interface*

Long Syntax:

TNIP.009 Packet GRE with no key discarded (expected *expected_key*), interface *interface*

Description:

This message is generated when a packet has no key and the tunnel is configured to check identification key.

Cause:

Check configuration.

Action:

Set the key configuration properly

TNIP.010

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.010 Dsc GRE pack with key (exp *key*), int *interface*

Long Syntax:

TNIP.010 Packet GRE with key discarded, key *key* (not expected key), interface *interface*

Description:

This message is generated when a packet has key and the tunnel is configured not to check identification key.

Cause:

Check configuration.

Action:

Set the key configuration properly

TNIP.011

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.011 Dsc GRE pack seq *seq_num* less exp *expected_seq_num*, int *interface*

Long Syntax:

TNIP.011 Packet GRE discarded, sequence number *seq_num* less than expected *expected_seq_num*, interface *interface*

Description:

This message is generated when a packet has a wrong sequence number. The received sequence number, together with the correct sequence number, is displayed.

Cause:

The tunnel ends are not synchronized.

Action:

Wait until ends are synchronized.

TNIP.012

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.012 Err GRE header flag recurs act, int *interface*

Long Syntax:

TNIP.012 Error in GRE header. Recursion option active, interface *interface*

Description:

The packet has the recursion option active so the desencaptulation GRE function has discarded it.

Multiple encapsulation is not accepted.

Cause:

The other tunnel end is sending GRE packets with multiple encapsulation.

Action:

Change configuration to avoid GRE packets going through the tunnel.

TNIP.013

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.013 Err GRE header vrsn *version* incorrect, int *interface*

Long Syntax:

TNIP.013 Error in GRE header, encapsulation version *version* invalid, interface *interface*

Description:

The packet has unknown version so the desencaptulation GRE function has discarded it.

Cause:

The other tunnel end is sending GRE packets with a later version or is making bad the header.

Action:

Change GRE version.

TNIP.014

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.014 Desencap pack GRE , inter prt 0x*protocol_num*, (*source_ip_address*->*destination_ip_address*), int *interface* seq *num_seq*

Long Syntax:

TNIP.014 Desencapsulated GRE packet with successfully, internal protocol 0x*protocol_num* (source *source_ip_address* and destination *destination_ip_address*), interface *interface* sequence number *num_seq*

Description:

Successful packet GRE decapsulation. The internal protocol of the GRE packet (the source and the destination) together with the interface are displayed

TNIP.015

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.015 Desencap pack GRE, prt inter 0x*protocol_num* unknown, int *interface*

Long Syntax:

TNIP.015 Error in desencapsulated GRE packet, unknown internal protocol 0x*protocol_num*, interface *interface*

Description:

Unknown protocol of the decapsulated GRE packet.

TNIP.016

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.016 Dsc out pack int no act, int *interface*

Long Syntax:

TNIP.016 The interface is down so the outgoing packet has been discarded, interface *interface*

Description:

The interface is down so the outgoing packet has been discarded.

TNIP.017

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.017 Dsc out pack int no conf, int *interface*

Long Syntax:

TNIP.017 The interface is no configured so the outgoing packet has been discarded, interface *interface*

Description:

The interface is no configured so the outgoing packet has been discarded.

TNIP.018

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.018 Dsc out pack invalid inter prt 0x*protocol_num*, int *interface*

Long Syntax:

TNIP.018 The internal protocol 0x*protocol_num* has been rejected so the outgoing packet has been discarded, interface *interface*

Description:

The internal protocol (payload protocol) has been rejected so the outgoing packet has been discarded.

TNIP.019

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.019 Encap GRE pack, inter prt 0x*protocol_num*, (*source_ip_address*->*destination_ip_address*), int *interface*

Long Syntax:

TNIP.019 Successful encapsulated GRE packet, internal protocol 0x*protocol_num*, (source *source_ip_address* destination *destination_ip_address*), interface *interface*

Description:

Successful encapsulated GRE packet. The protocol, source and destination of the encapsulated packet are displayed.

TNIP.020

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.020 Snd pack *str_enc_protocol*, ext prt 0x*protocol_num*, (*source_ip_address*->*destination_ip_address*), int *interface* seq *num_sec*

Long Syntax:

TNIP.020 Sending packet *str_enc_protocol*, ext prt 0x*protocol_num*, (source *source_ip_address* destination *destination_ip_address*), interface *interface* sequence number *num_sec*

Description:

A packet is sent. The encapsulation protocol, the external protocol (delivery protocol), source, destination and the interface that encapsulated the packet are displayed.

TNIP.021

Level: Common operation trace, C-TRACE

Short Syntax:

TNIP.021 Set up tn int *interface* (*source_ip_address*->*destination_ip_address*)

Long Syntax:

TNIP.021 Interface *interface* sets up a tunnel with source address *source_ip_address* and destination *destination_ip_address*

Description:

A free dynamic tunnel has taken to set up the new tunnel.

TNIP.022

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.022 Err tn rel, int *interface*

Long Syntax:

TNIP.022 Error tunnel release, interface *interface*

Description:

A dynamic tunnel is released because there are no static nor RIP routes or because keepalive failed.

TNIP.023

Level: Common operation trace, C-TRACE

Short Syntax:

TNIP.023 Rele tn int *interface* Rx idem IP *ip_address*

Long Syntax:

TNIP.023 Release tunnel interface *interface* set up another tunnel with the same address *ip_address*

Description:

A new tunnel is set up. It has the same IP address than a previous tunnel. At this moment the release conditions are attained.

TNIP.024

Level: Unusual operation trace, U-TRACE

Short Syntax:

TNIP.024 Err loop in tunnel, int *interface*

Long Syntax:

TNIP.024 Error loop in tunnel, interface *interface*

Description:

A packet that will cause an infinite loop has been detected.

TNIP.025

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.025 Pack GRE ciph, inter prt 0x*protocol_num*, (*source_ip_address*->*destination_ip_address*), int *interface*

Long Syntax:

TNIP.025 Successful ciphered GRE packet, internal protocol 0x*protocol_num*, (source *source_ip_address* and destination *destination_ip_address*), interface *interface*

Description:

The payload of a GRE packet has been ciphered successfully. The protocol, source and destination of the encapsulated packet are displayed.

TNIP.026

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.026 Pack GRE desciph, inter prt 0x*num_protocolo*, (*source_ip_address->destination_ip_address*), int *interface*

Long Syntax:

TNIP.026 Successful desciphered GRE packet, internal protocol 0x*num_protocolo* (source *source_ip_address* and destination *destination_ip_address*), interface *interface*

Description:

The payload of a GRE packet has been desciphered successfully. The internal protocol, source and destination of the GRE packet and the interface are displayed.

TNIP.027

Level: Unusual external error, UE-ERROR

Short Syntax:

TNIP.027 Err desciph GRE, int *interface*

Long Syntax:

TNIP.027 Error desciphering GRE, interface *interface*

Description:

A desciphered packet has a invalid cipher checksum or the protocol is not ciphered.

Cause:

Most likely, the cipher key is wrong or the packet is not ciphered.

Action:

If the problem persists, check cipher configuration in both tunnel ends.

TNIP.028

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.028 Desencap pack GRE , inter prt 0x*protocol_num*, int *interface* seq *num_seq*

Long Syntax:

TNIP.028 Desencapsulated GRE packet with successfully, internal protocol 0x*protocol_num* , interface *interface* sequence number *num_seq*

Description:

Successful packet GRE decapsulation. The internal protocol of the GRE packet together with the interface are displayed

TNIP.029

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.029 Pack GRE ciph, inter prt 0x*protocol_num*, int *interface*

Long Syntax:

TNIP.029 Successful ciphered GRE packet, internal protocol 0x*protocol_num*, interface *interface*

Description:

The payload of a GRE packet has been ciphered successfully. The protocol of the encapsulated packet is displayed.

TNIP.030

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.030 Encap GRE pack, inter prt 0x*protocol_num*, int *interface*

Long Syntax:

TNIP.030 Successful encapsulated GRE packet, internal protocol 0x*protocol_num*, interface *interface*

Description:

Successful encapsulated GRE packet. The protocol of the encapsulated packet is displayed.

TNIP.031

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.031 Desencap pack GRE , inter prt 0x*protocol_num*, (*source_mac_address*->*destination_mac_address*), int *interface* seq *num_seq*

Long Syntax:

TNIP.031 GRE packet successfully decapsulated, internal protocol 0x*protocol_num* (source *source_mac_address*nd destination *destination_mac_address*nterface *interface* sequence number *num_seq*

Description:

Successful packet GRE decapsulation. The internal protocol of the GRE packet (the mac source and the mac destination) together with the interface are displayed

TNIP.032

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.032 Pack GRE ciph, inter prt 0x*protocol_num*, (*source_mac_address*->*destination_mac_address*), int *interface*

Long Syntax:

TNIP.032 Successful ciphered GRE packet, internal protocol 0x*protocol_num*, (source *source_mac_address* and destination *destination_mac_address*), interface *interface*

Description:

The payload of a GRE packet has been ciphered successfully. The protocol, mac source and destination of the encapsulated packet are displayed.

TNIP.033

Level: Per packet trace, P-TRACE

Short Syntax:

TNIP.033 Encap GRE pack, inter prt 0x*protocol_num*, (*source_mac_address*-

>*destination_mac_address*), int *interface*

Long Syntax:

TNIP.033 Successful encapsulated GRE packet, internal protocol 0x*protocol_num*, (source *source_mac_address* destination *destination_mac_address*), interface *interface*

Description:

Successful encapsulated GRE packet. The protocol, mac source and destination of the encapsulated packet are displayed.

TNIP.034

Level: Unusual internal error, UI-ERROR

Short Syntax:

TNIP.034 No pkt fr kal int *interface*

Long Syntax:

TNIP.034 No packet for keepalive interface *interface*

Description:

A buffer could not be allocated when needed for keepalive request.

Cause:

This may be caused by temporary traffic loads. Many other causes are possible.

Action:

If the message persists, contact customer service.

TNIP.035

Level: Common operation trace, C-TRACE

Short Syntax:

TNIP.035 Snd kal int *interface*

Long Syntax:

TNIP.035 Sending keepalive packet on interface *interface*

Description:

A keepalive is going to be sent on specified interface to test tunnel endpoint reachability.

Cause:

Keepalive feature is enabled, so periodic keepalive packets are sent to test tunnel endpoint reachability.

TNIP.036

Level: Unusual operation trace, U-TRACE

Short Syntax:

TNIP.036 Bad kal id *received_id* (exp *expected_id*) int *interface*

Long Syntax:

TNIP.036 Unexpected keepalive identification number *received_id* (expected *expected_id*) received on interface *interface*

Description:

A keepalive was received on specified interface with an unexpected identification number. The received and expected identification numbers are shown, as well as incoming interface name.

Cause:

The most probable cause is that configured keepalive period is shorter than network delay.

Action:

If the message persists, increment keepalive period according to network delay.

TNIP.037

Level: Common operation trace, C-TRACE

Short Syntax:

TNIP.037 Rcv kal int *interface*

Long Syntax:

TNIP.037 Received keepalive on interface *interface*

Description:

A keepalive response was received on specified interface.

Cause:

As a result of a previously sent keepalive request packet, a matching response is received. This is the normal behavior when keepalive is enabled on an interface.

TNIP.038

Level: Unusual operation trace, U-TRACE

Short Syntax:

TNIP.038 Dsc kal int *interface*

Long Syntax:

TNIP.038 Discarded keepalive on interface *interface*

Description:

An unexpected keepalive response was received on specified interface.

Cause:

No keepalive response was expected, possibly because previous keepalive timed out and interface went down.

Action:

If the message persists, check tunnel interface status, keepalive configuration and network congestion state.