IP Tunnel Interface (TNIP)

Teldat Dm719-I

Legal Notice

Table of Contents

I Related Documents

Teldat Dm710-I PPP Interface

Teldat Dm715-I BRS

Teldat Dm768-I Dynamic Multipoint VPN

Teldat Dm772-I Common Configuration Interfaces

Teldat-Dm775-I VRF Lite Facility

Teldat Dm805-I IPv6 Addressing

Teldat Dm809-I IPv4/IPv6 over IPv6 Tunnel

Teldat Dm810-I IPv6 over IPv4 Tunnel

Chapter 1 IP Tunnel Interface (TNIP)

1.1 Description

1.1.1 Introduction

Tunneling is the name given to the process of encapsulating packets from one protocol within another protocol. Configuring a tunnel involves creating a virtual interface known as a *Tunnel Interface*. Tunnel interfaces are not tied to specific encapsulations or internally set transport protocols. They provide an architecture that is designed to support any standard encapsulation scheme. Since tunnels are point-to-point links, you must configure a separate tunnel for each link.

The main tunneling components are:

- Internal, payload, or transport protocol: this is the protocol you are encapsulating (IPv4, IPv6 or SRT).
- Carrier Protocol: this is the encapsulation protocol.

  - Generic Routing Encapsulation (GRE).
- Transport or delivery protocol: this is the protocol used to carry the encapsulated internal protocol (IPv4 or IPv6).

This manual explains how to tunnel IPv4 traffic over IPv4/IPv6 GRE tunnels, how to tunnel IPv6 traffic over IPv4/IPv6 GRE tunnels, and how to tunnel IPv4 and IPv6 over SRT GRE tunnels.

The following manuals explain how to tunnel IPv6 traffic over Ipv4 tunnels and IPv4/IPv6 traffic over IPv6 tunnels:

- Check Teldat manual *Dm810-I IPv6 over IPv4 Tunnel* to learn how to tunnel IPv6 traffic over IPv4 tunnels.
- Check Teldat manual *Dm809-I IPv4/IPv6 over IPv6 Tunnel* to learn how to tunnel IPv4/IPv6 traffic over IPv6 tunnels.

For convenience, IP will stand for IPv4 from here onwards.

1.1.2 Advantages of tunneling

Encapsulating a protocol within another can be useful in several situations:

- To interconnect multiprotocol local networks over a single protocol backbone.
- To solve the problem of interconnecting networks that have limited hop counts and that would not be able to connect without this procedure.
- To connect discontinuous subnetworks.
- To allow Virtual Private Networks across WANs.

1.1.3 Special considerations

The following considerations/precautions should be borne in mind when configuring tunnels:

- Encapsulation and decapsulation, which take place at the tunnel endpoints, are slow operations.
- Special care must be taken with the configurations and you need to consider security and topology issues. For example, you could configure a tunnel with a source and destination that are not restricted by firewalls.
- You need to consider the tunnel media. You might be tunneling across super fast FDDI networks or slow 9,600 bps baud links. Some internal protocols do not function very well in mixed media networks.
- Multiple point-to-point tunnels could saturate a link with routing information.
- Routing protocols that decide the best route based solely on the hop count prefer the tunnel (even if a better route exists). This is because the tunnel always looks like one hop.
- An even worse problem would be if the routing information from the tunneled network were to get mixed up with information from the transport networks. In

this case, the best route to the tunnel destination is via the tunnel. This type of route is known as a *recursive route* and causes the tunnel interface to shut down temporarily. To avoid this kind of problem, the routing information should be kept separate:

- Using a different AS number or TAG.

- Using a different routing protocol.

- Using static routes for the first hop (but watch out for routing loops).

1.2 Encapsulated frame format

The transport or delivery protocol for IP tunneling is IP (or IPv6). Thus, the encapsulated frame format is as follows:

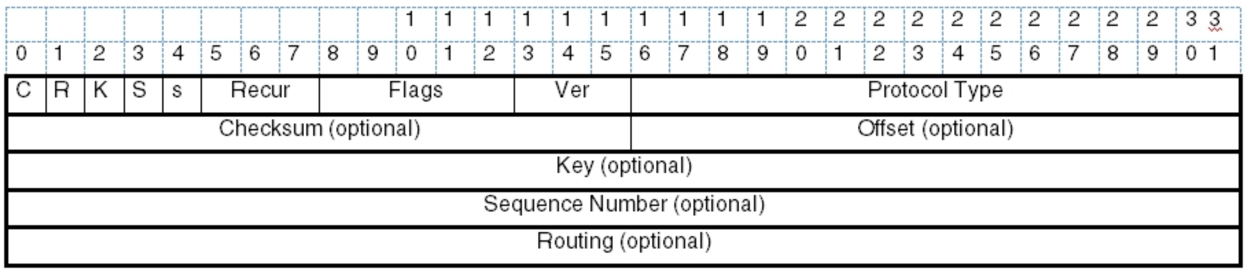
Delivery protocol header: IP (or IPv6)
Encapsulation protocol header
Payload protocol packet

1.2.1 IP over IP/IPv6 GRE tunnels

In this case, the encapsulation protocol is GRE. The payload protocol or protocol being encapsulated is either IP or IPv6. Generic Routing Encapsulation (GRE) is described in RFC 1701 and the encapsulation of IP packets over IP GRE tunnels is defined in RFC 1702.

Delivery protocol header: IP
Encapsulation protocol header: GRE
Payload protocol: IP (or IPv6)

The IP header is beyond the limits of this document. 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 the bit 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)

Contains a three bit unsigned integer showing the number of additional encapsulations allowed. 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 from 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 ensure packets arrive in the correct order.

Routing (variable length)

Does not exist.

When IP is encapsulated in IP using GRE, the ToS and IP security options are copied from the payload protocol header to the delivery protocol header. The TTL, however, is not copied but is set to the default value for IP. This prevents RIP packets sent through the tunnel from expiring before they reach their destination. When IPv6 is encapsulated within IP, the ToS value of the outer protocol header (IP) is taken from the equivalent field of the inner protocol header (IPv6), i.e., from the traffic class field.

1.2.2 IP over SRT GRE tunnels

Once again, the encapsulation protocol is GRE. In this case, the payload protocol or protocol being encapsulated is SRT.

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

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

1.2.3 IPv6 over IP/IPv6 GRE tunnels

The encapsulation protocol is also GRE. In this case, the outer protocol is IPv6 while the inner protocol or protocol being encapsulated is IP or IPv6.

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

When IPv6 is encapsulated in IPv6 using GRE, the traffic class field in the delivery protocol header is copied from the payload protocol header. The hop limit, however, is not copied. It is set to the default hop limit value for IPv6. When IP is encapsulated in IPv6, the traffic class field from the outer protocol header (IPv6) is taken from the equivalent field of the inner protocol header (IP), i.e., from the ToS field.

1.2.4 IPv6 over SRT GRE tunnels

Once again, the encapsulation protocol is GRE. In this case, the inner protocol or protocol being encapsulated is SRT. The GRE header fields are filled out and interpreted in the same way.

The hop limit and traffic class values in the delivery protocol header are the default IPv6 hop limit and traffic class values.

1.3 “Keepalive” maintenance packets

The IP tunnel interface has a *keepalive* mechanism to monitor connectivity with the remote end of the tunnel. This mechanism ensures that the interface is only operational when the two tunnel endpoints are conversing, allowing alternative routes (backup) to be taken without having to use routing protocols like RIP and OSPF.

Connectivity is monitored by sending maintenance packets and checking for a response. The following sections describe the *keepalive* maintenance packets used in IP tunnel interfaces.

The description below is also relevant to IPv6 tunnels (i.e., where the delivery protocol is IPv6).

1.3.1 Keepalive request packet

The packet sent by the device to determine whether the connection with the remote end is still up and running consists 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 inner 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 (except for sequence number), and the inner packet protocol = 0x0000.

1.3.2 Keepalive response packet

When a *keepalive* request packet reaches the tunnel destination endpoint (tunnel destination IP address), it is processed and the inner packet decapsulated by the corresponding device. This inner packet is returned to the tunnel source endpoint via conventional routing.

The *keepalive* request packet is thus a conventional IP packet encapsulated in GRE. It is routed normally by the remote device even though it does not have a keepalive feature.

Keepalive response packets are distinguished by the protocol field of the inner packet (in the GRE header), which is set to 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 (except for sequence number), and the inner packet protocol = 0x0000.

1.4 IPSec tunnel protection

Configuring a tunnel ip interface to terminate IPsec tunnels is possible. Encryption occurs in the tunnel when the traffic is forwarded to the interface. For further information on this, please refer to Teldat manual *Dm739-I IPSec*.

1.5 References

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

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

Chapter 2 IP Tunnel Interface Configuration (TNIP)

2.1 How to create an IP tunnel interface (TNIP)

To create an IP tunnel interface, enter **add device tnip <tunnel identifier>** in the global configuration menu:

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

Then access the configuration by entering **network tnipX,** where **X** represents the tunnel identifier:

```
Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
tnip1 config>
```

The TNIP interface supports IPv4 and IPv6. To enable IPv4 on the TNIP interface, assign an IPv4 address to the selected interface or configure the interface as an unnumbered interface. With IPv6, you must either enable it or configure an IPv6 address as indicated in Teldat manual *Dm805-I IPv6 Addressing*.

Example - enabling IP with a known IP address:

```
tnip1 config>IP ADDRESS 5.5.5.1 255.255.0.0
tnip1 config>
```

Example - enabling IP with an unnumbered interface:

```
tnip1 config>IP ADDRESS unnumbered
tnip1 config>
```

Example - enabling the IPv6 protocol:

```
tnip1 config>IPV6 ENABLE
tnip1 config>
```

2.2 IP tunnel interface configuration (TNIP)

This chapter describes the TNIP interface configuration commands. To access the TNIP configuration environment, enter **network <TNIP interface>**.

```
Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
tnip1 config>
```

Some commands are common to all device interfaces. These commands are described in Teldat manual *Dm772-I Common Configuration Interfaces*. Commands for IPv6 over IPv4 tunnels can be found in Teldat manual *Dm810-I IPv6 over IPv4 Tunnel*, while commands for Ipv4/IPv6 over IPv6 tunnels are in Teldat manual *Dm809-I IPv4/IPv6 over IPv6 Tunnel*.

The available commands are as follows:

Command	**Function**
6RD	Configures the 6rd tunnel parameters (please see Teldat manual *Dm810-I IPv6 over IPv4 Tunnel*).
CHECK-SOURCE	Enables the tunnel source availability check.
DESTINATION	Configures the tunnel destination IPv4/IPv6 address or hostname (available from version 11.01.07)
DISABLE	Disables the tunnel interface (obsolete as of versions 11.00.03 and 11.01.00).
ENABLE	Enables the tunnel interface (obsolete as of versions 11.00.03 and 11.01.00).

ENCAPSULATION	Accesses the encapsulation protocol configuration menu.
IPv6	Configures IPv6 protocol parameters (please see Teldat manual *Dm805-I IPv6 Addressing*).
KEEPALIVE	Enables *keepalive* maintenance.
LIST	Displays configured parameters.
MODE	Selects the tunnel interface encapsulation mode (encapsulation protocol).
NHRP	NHRP protocol configuration commands.
NHRP-TOS	Allows NHRP packets to be marked with a ToS value.
NO	Disables or eliminates functionalities.
PATH-MTU-DISCOVERY	Enables *Path MTU Discovery* in the tunnel.
PROTECTION-IPSEC	Enables tunnel protection with IPSec.
QOS-PRE-CLASSIFY	Enables pre-classification for BRS packets.
SOURCE	Configures the tunnel source IPv4/IPv6 address.
VRF-ENCAP	Specifies *VPN Routing/Forwarding* instance parameters.
EXIT	Exits the TNIP configuration menu.

2.2.1 CHECK SOURCE

The **no check-source** command disables the tunnel source availability check. Use **check-source** to restore the check so that tunnels go back to operating normally.

The tunnel source check is executed in accordance with the **source** command:

- SOURCE <a.b.c.d>: this checks that the configured IP address is enabled (local address, management address or address assigned to an enabled interface).
- SOURCE <interface>: this checks that the configured interface is enabled and has been assigned an IP address.

The tunnel cannot be activated if the tunnel source availability check returns a negative result.

2.2.2 DESTINATION

Configures the tunnel destination IPv4/IPv6 address or a hostname.

The destination IPv4/IPv6 address must match the tunnel source IPv4/IPv6 address configured in the router at the opposite end of the tunnel. If the tunnel destination IPv4/IPv6 address does not match the source address at the other end, any packets sent to that router are discarded (as they don't belong to said tunnel).

A path to the destination IPv4/IPv6 address is essential because tunnel packets cannot be routed without one. As a precautionary measure, the route should be static to avoid the sort of recursive problems in the routing table described in chapter 1.

Use **no destination** when you do not want to specify the destination IP address (dynamic or promiscuous tunnel). These types of tunnels are only available when IPv4 is the delivery protocol.

When a router has a routable but dynamic IP address, a DynDNS protocol is commonly used to update the DNS servers with the new IP address. The router must be given a hostname to identify its dynamic IP address. The router at the other end of the tunnel must be configured with the same hostname as destination. In addition, a DNS server will need to be configured to resolve the remote hostname (please see Teldat manual *Dm723-I DNS*). A DNS request is periodically issued to check if a new IP address has been given to the other side of the tunnel.

Syntax:

```
tnip1 config>DESTINATION ?
  <a.b.c.d>     Ipv4 format
  <a::b>        Ipv6 address
  <word>        Host name
tnip1 config>
```

Example 1:

```
tnip1 config>DESTINATION 66.187.232.56
tnip1 config>
```

Example 2:

```
tnip1 config>DESTINATION 2001:DB8:3::2
tnip1 config>
```

Example 3:

```
tnip1 config>DESTINATION peer1.gre.id.teldat.com
tnip1 config>
```

Command history:

Release	Modification
11.01.07	The possibility of setting a *hostname* in the destination command has been added as of version 11.01.07.

2.2.3 DISABLE

Disables the tunnel interface.

Syntax:

```
tnip1 config>DISABLE ?
  <cr>
tnip1 config>
```

Example:

```
tnip1 config>DISABLE
tnip1 config>
```

Command history:

Release	Modification
11.00.03	This command is obsolete. As of version 11.00.03, the behavior has changed and the tunnel interface is enabled by default.
11.01.00	This command is obsolete. As of version 11.01.00, the behavior has changed and the tunnel interface is enabled by default.

2.2.4 ENABLE

Enables the tunnel interface.

Syntax:

```
tnip1 config>ENABLE ?
  <cr>
tnip1 config>
```

Example:

```
tnip1 config>ENABLE
tnip1 config>
```

Command history:

Release	Modification
11.00.03	This command is obsolete. As of version 11.00.03 the behavior has changed and the tunnel interface is enabled by default.

Release	Modification
11.01.00	This command is obsolete. As of version 11.01.00 the behavior has changed and the tunnel interface is enabled by default.

2.2.5 ENCAPSULATION

Accesses the encapsulation protocol configuration. GRE (*Generic Routing Encapsulation*) is the only encapsulation protocol supported at the moment. This submenu is described in section 3.

Syntax:

```
tnip1 config>ENCAPSULATION ?
  <cr>
tnip1 GRE config>
```

Example:

```
tnip1 config>ENCAPSULATION

-- GRE Configuration --
tnip1 GRE config>
```

2.2.6 KEEPALIVE

Enables the IP tunnel *keepalive* mechanism, which sends periodic *keepalive* packets through the tunnel to check its status. If there is no response within the configured time period, the router declares the tunnel interface down until a new connection is established.

Note

Keepalive messages are only sent through dynamic tunnels once the tunnel has been established, that is, once the tunnel IP address has been learned (please see the **list status** monitoring command).

Keepalive messages are never sent through promiscuous tunnels.

The basic format for this command is as follows:

keepalive [<*period*> [<*attempts*> [<*attempts period*>]]]

These parameters are described below:

Parameter	Description
period	The time interval between successive *keepalive* transmissions. It also acts as maximum response time, as only the responses to the last *keepalive* request message are considered. This interval ranges from 1 second to 10 hours. Default is 10 seconds.
attempts	The number of retransmissions without response to determine loss of connectivity. Valid values range from 1 to 255 unanswered transmissions. Default is 3.
attempts period	The time interval between successive *keepalive* transmissions, if no response to the previous *keepalive* transmission is received. This parameter allows you to configure a higher rate of retries to detect loss of connectivity faster.

You might want to further personalize the *keepalive* schema. Using a set of advanced parameters, you can define the appropriate times and criteria for each maintenance phase. The commands used to configure advanced *keepalive* parameters are as follows:

keepalive down timeout <*down_timeout*>

keepalive down period reachable <*down_per_reach*>

keepalive down period unreachable <*down_per_unreach*>

keepalive down stability threshold <*down_threshold*>

keepalive up timeout <*up_timeout*>

keepalive up period reachable <*up_per_reach*>

keepalive up period unreachable <*up_per_unreach*>

keepalive up stability threshold <*up_threshold*>

These parameters are described below:

Parameter	Description
down_timeout	This parameter applies when the interface is *down* and specifies the amount of time the device waits for a response following a *keepalive* transmission.
down_per_reach	This parameter applies when the interface is *down* and specifies the time interval between one *keepalive* transmission and the next if the corresponding response is received on time (see down_timeout parameter). This parameter must be greater than ,or equal to, the **down_timeout** parameter.
down_per_unreach	This parameter applies when the interface is *down* and specifies the time interval between one *keepalive* transmission and the next if the corresponding response was not received on time (see **down_timeout** parameter). This parameter must be greater than the **down_timeout** parameter.
down_threshold	This parameter applies when the interface is *down* and specifies the maximum number of *keepalive* retransmissions with a response to determine that the connection is alive, and therefore the interface enters an *up* state.
up_timeout	This parameter applies when the interface is *up* and specifies the amount of time the device waits for a response after a *keepalive* transmission.
up_per_reach	This parameter applies when the interface is *up* and specifies the time interval between one *keepalive* transmission and the next if the corresponding response is received in time (see **up_timeout** parameter). This parameter must be greater than, or equal to, the **up_timeout** parameter.
up_per_unreach	This parameter applies when the interface is *up* and specifies the time interval between one *keepalive* transmission and the next if the corresponding response was not received on time (see **up_timeout** parameter). This parameter must be greater than, or equal to, the **up_timeout** parameter.
up_threshold	This parameter applies when the interface is *up* and specifies the maximum number of *keepalive* retransmissions without a response to determine that the connection is lost, and therefore the interface enters a *down* state.

You enable *keepalive* packets when you configure one of the parameters. The actual configuration is the result of applying the different parameters or, in their absence, default values, with the limitations inherent in the *keepalive* schema (e.g., period greater than, or equal to, timeout). Enter **list** to see the actual values for the current configuration.

Enter **no keepalive** to disable *keepalive* maintenance.

Syntax:

```
tnip1 config>keepalive ?
  <1s..10h>    Keepalive period (default 10 seconds)
  down         Configuration while interface is down
  up           Configuration while interface is up
  <cr>
tnip1 config>keepalive 30 ?
  <1..255>    Keepalive retries (default 3 retries)
  <cr>
tnip1 config>keepalive 30 5 ?
  <1s..10h>    Keepalive period on retries
  <cr>
tnip1 config>keepalive down ?
  period       Period of time for keepalive transmissions
  stability    Stability definition
  timeout      Timeout for keepalive response
tnip1 config>keepalive down period ?
```

```
  reachable      Period for reachable keepalive
  unreachable    Period for unreachable keepalive
tnip1 config>keepalive down period reachable ?
  <1s..10h>    Time value
tnip1 config>keepalive down period unreachable ?
  <2s..10h>    Time value
tnip1 config>keepalive down stability ?
  threshold    Threshold for keepalive success
tnip1 config>keepalive down stability threshold ?
  <1..255>    Number of consecutive reachable keepalives
tnip1 config>keepalive down timeout ?
  <1s..10h>    Time to wait for keepalive response
tnip1 config>keepalive up ?
  period       Period of time for keepalive transmissions
  stability    Stability definition
  timeout      Timeout for keepalive response
tnip1 config>keepalive up period ?
  reachable      Period after reachable keepalive
  unreachable    Period after unreachable keepalive
tnip1 config>keepalive up period reachable ?
  <1s..10h>    Time value
tnip1 config>keepalive up period unreachable ?
  <1s..10h>    Time value
tnip1 config>keepalive up stability ?
  threshold    Threshold for keepalive failure
tnip1 config>keepalive up stability threshold ?
  <1..255>    Number of consecutive unreachable keepalives
tnip1 config>keepalive up timeout ?
  <1s..10h>    Time to wait for keepalive response
tnip1 config>
```

Example:

```
tnip1 config>KEEPALIVE 30 5
tnip1 config>
```

2.2.7 LIST

Displays the tunnel configuration.

Example:

```
tnip1 config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 212.95.195.132, destination 66.187.232.56
QoS preclassify: disabled
Keepalive enabled
  Down interface keepalive configuration:
    Timeout:                             10s
    Period for unreachable destinations: 11s
    Period for reachable destinations:   10s
    Stability threshold:                 2
  Up interface keepalive configuration:
    Timeout:                             10s
    Period for unreachable destinations: 10s
    Period for reachable destinations:   10s
    Stability threshold:                 3
NHRP type of service: 25
tnip1 config>
```

Tunnel mode:	Indicates the encapsulation type and state (enabled/disabled).
Tunnel source / destination:	Tunnel source / destination IPv4/IPv6 addresses/hostname (hostname only for destination and available from version 11.01.07)
QoS preclassify:	Shows whether BRS pre-classification is enabled.
Keepalive:	Displays the *keepalive* maintenance configuration.

NHRP type of service:	Displays the chosen service type for NHRP packets.

2.2.8 MODE

Selects the encapsulation mode. Supports GRE (*Generic Routing Encapsulation*), mGre (*Multipoint GRE*), IPSec (described in Teldat manual *Dm739-I IPSec*), IPv6 over IPv4 (manuals, 6to4 and 6rd) and IPv4/IPv6 over IPv6. Modes relating to IPv6 over IPv4 tunnels and IPv4/IPv6 over IPv6 tunnels are described in the Teldat *Dm810-I IPv6 over IPv4 Tunnel* and Teldat *Dm809-I IPv4/IPv6 over IPv6 Tunnel* manuals respectively.

Syntax:

```
tnip1 config>MODE ?
  gre       Generic Routing Encapsulation Protocol
  ipsec     IPSec tunnel encapsulation
  ipv6      IPv4 or IPv6 over IPv6 tunnel
  ipv6ip    IPv6 over IPv4 tunnel
tnip1 config>MODE GRE ?
  ip            Over IP
  ipv6          Over IPv6
  multipoint    Over IP (multipoint)
  <cr>
tnip1 config>MODE GRE IP?
  transparent    Transparent Ether Bridging
  <cr>
tnip1 config>
```

When TNIP is on a Bridge and **MODE GRE IP TRANSPARENT** is configured, the protocol field of the GRE header will include the Transparent Ether Bridging protocol value [0x6558].

Example 1:

```
tnip1 config>MODE GRE IP
tnip1 config>
```

Example 2:

```
tnip1 config>MODE GRE MULTIPOINT
tnip1 config>
```

Example 3:

```
tnip1 config>MODE GRE IPV6
tnip1 config>
```

Command history:

Release	Modification
11.01.04	The IPSEC mode option was introduced as of version 11.01.04.
11.01.08	The GRE IP TRANSPARENT mode option was introduced as of version 11.01.08.

2.2.9 NHRP

Configures the NHRP protocol. For further information, please see the *Configuring the NHRP protocol* section found in Teldat manual *Dm768-I Dynamic Multipoint VPN*.

2.2.10 NHRP-TOS

Marks NHRP packets with a Type of Service (ToS) value. By setting a ToS value you can, for example, filter NHRP packets to prevent an NHRP registration request from initiating a call in the UMTS link.

Syntax:

```
tnip1 config>NHRP-TOS ?
  <0..255>    IPv4 type of service value
tnip1 config>
```

Example:

```
tnip1 config>NHRP-TOS 25
```

```
tnip1 config>
```

2.2.11 PATH-MTU-DISCOVERY

Activates the feature to determine the appropriate MTU size (to prevent packet fragmentation). This feature is responsible for discovering the lowest MTU value along the path from one end of the tunnel to the other, thereby preventing fragmentation from taking place. Any TCP/IP packets sent by the device have the DF (*don't fragment*) bit set when this is enabled.

Syntax:

```
tnip1 config>PATH-MTU-DISCOVERY ?
  <cr>
tnip1 config>
```

Example:

```
tnip1 config>PATH-MTU-DISCOVERY
tnip1 config>
```

2.2.12 PROTECTION-IPSEC

Enables IPSec protection for tunnel IP interfaces. The command is available when the ipsec ip, gre ip or gre multipoint modes are configured. For further information, please refer to Teldat manual *Dm739-I IPSec*.

Syntax:

```
tnip1 config>PROTECTION-IPSEC
tnip1 config>
```

Command history:

Release	Modification
11.01.04	The "*PROTECTION-IPSEC*" command was introduced as of version 11.01.04.

2.2.13 QOS-PRE-CLASSIFY

Enables BRS packet pre-classification. When this option is enabled, packets reaching the tunnel are classified through BRS (please see Teldat manual *Dm715-I BRS*) before being encapsulated by the tunnel. This allows you to distinguish between the various kinds of IP traffic sent through the tunnel. If this option is disabled, packets are sorted after they have been encapsulated. Hence, all traffic processed by the tunnel receives the same IP header (from the tunnel) and is classified in the same BRS class.

Enter **no qos-pre-classify** to disable this parameter.

Syntax:

```
tnip1 config>QOS-PRE-CLASSIFY ?
  <cr>
tnip1 config>
```

Example:

```
tnip1 config>QOS-PRE-CLASSIFY
tnip1 config>
```

2.2.14 SOURCE

Configures the tunnel's source IPv4/IPv6 address.

The source IPv4/IPv6 address and the IPv4/IPv6 address of one of the interfaces configured in the router (Ethernet, PPP, Loopback, etc.,), except that of the tunnel itself, must match. The source IPv4/IPv6 address should also match the destination IPv4/IPv6 address configured in the device at the other end of the tunnel.

If the source IPv4/IPv6 address for the tunnel packets does not match any of the router's interfaces, the router does not accept them and tries to route them to an alternative device.

The link is not possible if the packet source IPv4/IPv6 address does not match the destination address configured in the device at the other end.

The tunnel source for tunnels where the delivery protocol is IPv4 can be:

- A numbered IPv4 address.
- An interface that uses the address configured in the interface as the source address. In the case of PPP interfaces that receive dynamically assigned IP addresses and have an *unnumbered* address (please see Teldat manual *Dm710-I PPP Interface*), the source address is taken when it is assigned by the IPCP protocol.
- Not configured. If the IP address is filled with 0.0.0.0, or left empty, then it is a dynamic tunnel (see chapter 3, *Dynamic Tunnels*). This is only valid for PPP interfaces.

The tunnel source for tunnels where the delivery protocol is IPv6 can be:

- A numbered IPv6 address.
- An interface that uses one IPv6 address configured and active in the interface as the source address. When the source interface has several IPv6 addresses, the tunnel source is chosen based on the destination address of the tunnel.

Syntax:

```
tnip1 config>SOURCE ?
  <a.b.c.d>      Tunnel source address
  <a::b>         Ipv6 address
  <interface>    Tunnel source interface
tnip1 config>
```

Example 1:

```
tnip1 config>SOURCE 212.95.195.132
tnip1 config>
```

Example 2:

```
tnip1 config>SOURCE ppp1
tnip1 config>
```

Example 3:

```
tnip1 config>SOURCE 2001:db8:3::1
tnip1 config>
```

Command history:

Release	Modification
11.01.04	As of version 11.01.04, an interface can be configured as source on tunnels where the delivery protocol is IPv6 (IPv6 and GRE IPv6 mode tunnels).

2.2.15 VRF-ENCAP

Associates the IP tunnel destination address with a VRF instance. The path to said destination is then consulted in the associated VRF routing table. The tunnel source and destination must be in the same VRF. For more information, please see the Teldat- **Dm775-I VRF** Lite Facility manual.

Syntax:

```
tnip1 config>VRF-ENCAP ?
  <1..32 chars>    VPN Routing/Forwarding instance name
tnip1 config>
```

Example:

```
tnip1 config>VRF-ENCAP thisIsAnExample
tnip1 config>
```

Command history:

Release	Modification
11.01.09	This command was introduced in tunnels where IPv6 is the transport protocol as of version 11.01.09.

2.3 Configuring GRE

This section describes the commands available on the router to configure Generic Routing Encapsulation (GRE). To access the GRE configuration environment, enter **encapsulation** in the tunnel interface configuration menu (with GRE encapsulation mode configured in the interface).

```
Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
tnip1 config>ENCAPSULATION

-- GRE Configuration --
tnip1 GRE config>
```

The available commands are as follows:

Command	**Function**
CHECKSUM	Enables the point-to-point 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 any datagrams that are received out of their correct order (obsolete as of version 11.01.00).

2.3.1 CHECKSUM

Enables checksums on a GRE tunnel. By default, the integrity of the data is not guaranteed by the tunnel. When you turn on checksums, the router sends GRE packets with a checksum field. The router verifies the checksum of every GRE packet it receives and drops any packets that don't match (even if the checksum option is disabled).

Enter **no checksum** to disable checksums.

Syntax:

```
tnip1 GRE config>CHECKSUM ?
  <cr>
tnip1 GRE config>
```

Example:

```
tnip1 GRE config>CHECKSUM
tnip1 GRE config>
```

2.3.2 CIPHER

Activates the RC4 cipher for those packets encapsulated in the GRE tunnel. Default is disabled.

Note

The response packets are not ciphered like the *keepalive* packets because they are not encapsulated in the tunnel.

Enter **no cipher** to disable RC3 cipher.

Syntax:

```
tnip1 GRE config>CIPHER ?
  <cr>
tnip1 GRE config>
```

Example:

```
tnip1 GRE config>CIPHER
tnip1 GRE config>
```

2.3.3 CIPHER-KEY

Configures the *cipher key* tunnel interface. This key admits a maximum of 32 alphanumerical characters. From versions 10.08.34.05.15, 11.00.05.10.05 and 11.01.04 onwards, the key appears encrypted by default whenever the configuration is shown.

To reset the default ciphered key in the GRE tunnels, enter **no cipher-key.**

Syntax:

```
tnip1 GRE config>CIPHER-KEY ?
  ciphered         Enter a ciphered key
  <1..32 chars>    Text
tnip1 GRE config>
```

Example:

```
tnip1 GRE config>CIPHER-KEY example
tnip1 GRE config>show config
          cipher-key ciphered 0x92B067B7AEBE3742
```

2.3.3.1 CIPHERED

Enables the setting of a ciphered key. When the configuration is shown, the key appears encrypted (i.e., just as it was entered).

Example:

```
tnip1 GRE config>cipher-key ciphered 0x4D0279AF2D5AB11D
tnip1 GRE config>show config
          cipher-key ciphered 0x4D0279AF2D5AB11D
```

Command history:

Release	Modification
11.01.04	The ciphered option was introduced as of version 11.01.04.

2.3.4 KEY

Enables an ID key for a tunnel interface. When this option is enabled, the device asks for the identifier key of the tunnel in question. The tunnel identifier **must be the same at both ends** of the tunnel. The identifier is a whole number between 0 and 4294967295 (32 bits). Default is disabled.

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

Syntax:

```
tnip1 GRE config>KEY ?
  plain              ID plain key for the tunnel interface
  ciphered           ID ciphered key for the tunnel interface
  <0..4294967295>    Value in the specified range
tnip1 GRE config>
```

Example:

```
tnip1 GRE config>KEY 5
tnip1 GRE config>
```

2.3.4.1 PLAIN

Enables the setting of a plain key. When the configuration is shown, the key appears encrypted. Note the difference with the **key <0..4294967295>** command, where the entered key does not appear ciphered.

Example:

```
tnip1 GRE config>key plain 5
tnip1 GRE config>show config

         key ciphered 0x0E4DCE3F8B680B09
```

Command history:

Release	Modification
11.01.04	The plain option was introduced as of version 11.01.04.

2.3.4.2 CIPHERED

Enables the setting of a ciphered key. When the configuration is shown, the key appears encrypted (i.e., just as it was entered).

Example:

```
tnip1 GRE config>key ciphered  0x0E4DCE3F8B680B09
tnip1 GRE config>show config

         key ciphered 0x0E4DCE3F8B680B09
```

Command history:

Release	Modification
11.01.04	The ciphered option was introduced as of version 11.01.04.

2.3.5 LIST

Displays the GRE protocol configuration.

Example:

```
tnip1 GRE config>LIST
RC4 Cipher.................: enabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [5]
Drop Out-of-Order Datagrams: disabled
tnip1 GRE config>
```

RC4 Cipher:	Shows whether the RC4 cipher is enabled.
End-to-End Checksumming:	Shows whether the point-to-point checksum is enabled.
Tunnel identification key:	Tunnel identifier (if this is enabled).
Drop Out-of-Order Datagrams:	Drops datagrams that are not received in order.

Command history:

Release	Modification
11.01.00	The command output has changed and information on Drop Out-of-Order Datagrams will not be shown as of version 11.01.00 (as the **sequence-datagrams** command is obsolete).

2.3.6 SEQUENCE-DATAGRAMS

Enables the option to ensure that incoming datagrams are received in order. When this option is enabled, the router checks the sequence number included in the GRE header and drops any packets arriving out of order. This option is disabled in the GRE tunnel by default.

Enter **no sequence-datagrams** to disable the sequence number option.

Syntax:

```
tnip1 GRE config>SEQUENCE-DATAGRAMS ?
  <cr>
```

```
tnip1 GRE config>
```

Example:

```
tnip1 GRE config>SEQUENCE-DATAGRAMS
tnip1 GRE config>
```

Command history:

Release	Modification
11.01.00	The "*SEQUENCE-DATAGRAMS*" command is obsolete as of version 11.01.00.

Chapter 3 Dynamic Tunnels (Internet)

3.1 Description

Tunneling over public networks and the Internet provides a cheap and efficient means of interconnecting diverse local networks. The tunneling technique described in the above chapters could be used for this purpose. However, a number of problems (like the ones below) could arise. These will be dealt with in this chapter.

(1) A tunnel can only be established between two points if both of them know the end IP address. This means that access to the tunnel is only possible through authenticated access and static IP addresses (which are a limited and expensive resource on the Internet).

(2) Connecting *n* number of local networks would require you to configure and establish *n-1* number of tunnels in each.

The solution to these problems is to have a single central device (with a static/known IP address) supporting *n* number of tunnels and handling inter-tunnel traffic. This would automatically resolve the second problem and the first would disappear if the device was given the ability to tailor tunnel configuration and support connections from devices with changing IP addresses.

Note

Dynamic tunnels benefit from the dynamic tunnel reconfiguration that takes place with each new connection.

Note

From this point onwards, the central device will be referred to as *ISP router* (since it is usually found in an Internet Server Provider). Routers that connect to this device will be referred to as *client routers.*

3.1.1 Scenarios/problems

- **Scenario 1:** Local network devices obtain network services (html, ftp, etc.,) over the Internet via a router.
- **Scenario 2**: Remote local networks and the local ISP network are connected via routers using tunnels.
- **Scenario 3:** Remote local networks are connected to each other and the local ISP network via routers using tunnels.

If you want scenarios two and three, the client router configuration is relatively simple since we can predict that any non-local IP addresses will be accessible through the tunnel. If, however, you want scenario one at the same time, then the client routers have to be able to recognize whether they can reach a specific destination address through the tunnel or outside of it (Internet address), thus forcing them to learn the networks accessible via the tunnel. The way around this is to configure all possible routes on all clients, or, alternatively, to configure the routes in the ISP router, which then informs the clients by means of a routing protocol (RIP).

The ISP router must also know which networks are accessible via the client routers.

3.1.2 Tunnel types

The following tunnel classification looks at the different behaviors in each case:

- **Static**: where the source and destination addresses are fixed. We have discussed these above.
- **Dynamic**: when there is an unknown tunnel address (source or destination) prior to connection and the device to be connected is unknown.
- **Semi-dynamic**: This is a special kind of dynamic tunnel where, despite not knowing the tunnel address (source or destination), we do know which device to connect thanks to a unique tunnel identifier (GRE *key* field).
- **Promiscuous**: These are special static tunnels with unknown source and destination addresses. Running in default tunnel interface mode, they receive the traffic that isn't destined for any other tunnel interface but they don't allow

traffic to be transmitted (encapsulated).

(Dynamic and semi-dynamic tunneling are normal over the Internet when the remote device acquires an unassigned address).

3.1.2.1 Dynamic tunnels

Dynamic tunnels are recommended over the others because they are the easiest to configure and the most flexible.

Clients are given *n* number of tunnels by the ISP router, which they use when they connect. When a client stops sending information, a tunnel becomes available for a new connection.

This configuration is possible thanks to RIP, which allows client routers to share information with the ISP router on networks that can be reached through them (and vice-versa).

3.1.2.2 Semi-dynamic tunnels

Semi-dynamic tunnels, which are an extension of dynamic tunnels, discriminate against the remote routers allowed to connect in each ISP tunnel. This requires configuring a unique tunnel identifier that matches the one configured in the remote device (GRE *key* field). It is really like configuring a unique dynamic tunnel for each remote device.

These tunnels add two functions:

- Remote device identification.
- Routes can be added because the client connecting to each tunnel is already known. Hence, you don't have to enable RIP.

This is a more complex tunnel configuration because identifiers in the tunnel and in the remote devices must match. For this reason, we only recommended using this option when access control is essential and you can do without RIP and security.

3.1.2.3 Promiscuous tunnels

Promiscuous tunnels are special static tunnels with an unknown tunnel source or destination address.

As long as the tunnel identifier configuration (GRE *key* field) and the received packet match, promiscuous tunnels receive any traffic not destined to any of the other tunnel interfaces. They do not, however, allow any traffic to be transmitted (encapsulated).

This IP tunnel interface makes it possible for traffic to be received simultaneously from multiple tunnels on a single interface. However, the static nature of the interface makes transmission impossible because the interface cannot learn the tunnel addresses.

This kind of tunnel is useful in very special cases, for example, when using RIP to learn the remote networks accessed through IPSEC.

3.1.3 The importance of RIP

One of the trickiest aspects of these types of tunnels is to control the tunnel interface state: they go from *idle* to *connected* (connected to the client device) when asked to do so by the client device. They only remain in a *connected* state while they are being used. When they are no longer being used, they must return to the initial *idle* state pending future use.

Note

Given that a remote router could be switched off or disconnected without warning, one of the most critical aspects when setting up dynamic tunnels is the decision of whether to leave the tunnel up or not. Dialogue between the routers maintaining the tunnels is therefore necessary, and the RIP protocol is used for this purpose.

Note

You can identify a tunnel with a unique key to avoid reusing it or to reserve the tunnel exclusively for a particular client device.

Summary of RIP features:

a) Controls disconnection to allow the tunnel to be reused.

b) Provides information on accessible networks.

Problems presented by RIP:

When tunnel endpoint IP addresses belong to different networks, a router can receive access information on the tunnel destination network through the tunnel interface itself, causing it to lose access to the remote end. This won't ever happen on the Internet when the client-acquired address belongs to the ISP router network, but it will in all other cases.

You can fix this by adding a static route to the destination network, but it must be known a priori.

In any case, the routers are able to detect when this happens and report it through events and statistics.

3.2 User scenarios

You need to configure how the router will be used before you configure any tunnels in the router. A configuration that best fits your needs will ensure you get the most out of your router and communication line.

The most important decision is whether to configure static routes or delegate the issue to a routing protocol. While delegation would facilitate the configuration, it would also decrease performance because part of the line bandwidth would be used for exchanging messages between routers.

There isn't a general rule here, but you can base your decision on certain guidelines. To do so, you will need to define the router operating scenario.

- **Scenarios 2/3 (tunnel to interconnect local networks via the Internet):** All non-local addresses are accessible through the tunnel, so configuring this as the default route will suffice (except that the address of the remote end of the tunnel must be static).
- **Scenarios 1+2/3 (surf the Internet and interconnect local networks over the Internet via tunnels)**. Non-local addresses can be accessed over the Internet, but there are dynamic and semi-dynamic tunnels over the Internet configured in the client router for each remote LAN.

We should also take into account certain aspects seen in previous chapters:

- Since *dynamic* tunnels are reused, it is important to configure RIP when using them.

3.2.1 Tunnel function without surfing the Internet (scenarios 2/3)

Client configuration defines the tunnel as the default route (except for ISP and local destinations). This means we can disable RIP coming from the ISP, thus improving line performance as RIP traffic in this direction is high when the ISP supports a large number of tunnels.

3.2.1.1 Minimal configuration through RIP

This is achieved with dynamic (and therefore reusable) tunnels. RIP in the *client -> ISP* direction is essential to know which client networks are available.

Surfing permitted:	No
Type of tunnel:	Dynamic
Default route:	Tunnel
RIP:	Client -> ISP
Security:	Low
Configuration difficulty level	Very low
Efficiency	High

3.2.1.2 More complex configuration reducing RIP traffic

The client is identified via dedicated tunnels for each remote device requiring access, and RIP is not necessary in the *client -> ISP* direction. However, the identifiers must be maintained when configuring tunnels in ISP and clients.

Surfing permitted:	No
Type of tunnel:	Semi-dynamic
Default route:	Tunnel
RIP:	No
Security:	High
Configuration difficulty level	Average
Efficiency	Very high

3.2.2 Simultaneous tunnel and surfing (scenarios 1 + 2/3)

The default route in clients is the wide area network (the Internet), so any network destinations accessible through the tunnel must be learned. You can achieve this by means of static configuration in each client (or only in the ISP, which then informs the clients using RIP).

3.2.2.1 Maximum load in the network/minimum configuration

When there are few clients, the routing mechanism can be given to the RIP protocol. This means you can avoid having to configure static routes for clients.

Surfing permitted:	No
Type of tunnel:	Dynamic
Default route:	Internet
RIP:	Clients <-> ISP
Security:	Low
Configuration difficulty level	Low
Efficiency	Low if there are many tunnels

Sometimes, using RIP with a particular client is not a desirable option (as the client can be given a unique *key*).

Note

This scenario is very useful when you are looking to interconnect few remote networks.

3.2.2.2 Minimum load in the network/more complex configuration

RIP traffic can seriously affect tunnel performance when you start having an excessively large number of routes learned by the ISP device (either because of the large number of tunnels or because of the complexity of the remote local networks). In this case, you should consider dispensing with the routing protocol in the *ISP -> Client* direction. You would then have to statically configure the clients with the networks that are accessible through the tunnel.

Surfing permitted:	Yes
Type of tunnel:	Dynamic
Default route:	Internet
RIP:	Clients -> ISP
Security:	Average
Configuration difficulty level	Average

Efficiency	High

As in the previous cases, you can remove the RIP protocol in the *Client* -> *ISP* direction by setting a unique *key*.

Note

This scenario is particularly useful when you are looking to access remote local networks from the ISP, rather than interconnecting remote local networks (for example, when an entity wants to access its branch offices from its own ISP).

3.2.2.3 Void overload in the network/more complex configuration/client control

This scenario is based entirely on the use of distinct *keys*, so each tunnel interface is well-defined (i.e., the connecting client and the networks accessible through the interface are known).

Surfing permitted:	Yes
Tunnel type:	Semi-dynamic
Default route:	Internet
RIP:	No
Security:	High
Configuration difficulty level	High
Efficiency	Maximum

Overload is reduced to zero, since RIP traffic is totally eliminated by specifically configuring all the reachable routes.

Note

This scenario allows for greater client control, which is useful when the ISP offers network interconnection services to third parties.

Important

Duplicating addresses between client installations should be avoided when the service is offered to third parties (different clients cannot have the same subnet).

3.3 Security

Security aspects become of foremost importance when trying to connect local networks through a public network. Authentication mechanisms and mechanisms that prevent unwanted inter-tunnel traffic are required.

The first authentication mechanism uses fixed addresses, though this is expensive on the Internet. If this is not an option for you, then you will have to resort to other mechanisms (such as a GRE tunnel identifier, which is a unique key ID that must be known at both ends of the tunnel). With this option, the content of the GRE packet can also be encrypted.

Since it is routed through the ISP device, you have complete control over inter-tunnel traffic.

Chapter 4 IP Tunnel Interface Monitoring (TNIP)

4.1 IP tunnel interface monitoring (TNIP)

Access the TNIP interface monitoring menu by entering **network<TNIP interface>** on the general monitoring menu:

```
+NETWORK tnip1
-- TNIP protocol monitor --
tnip1+
```

The available commands are as follows:

Command	**Function**
? (HELP)	Lists the available commands or options.
NHRP	NHRP protocol monitoring (please see Teldat manual *Dm768-I Dynamic Multipoint VPN*.
LIST	Displays monitoring information.
EXIT	Exits the TNIP interface monitoring menu.

4.1.1 ? (HELP)

Lists the valid commands at the router configuration level. You can also use it to list the options available for a specific command.

Syntax:

```
tnip1+NHRP ?
  list    Display the monitoring information
  nhrp    Access NHRP protocol monitoring commands
  exit
tnip1+
```

4.1.2 LIST

Displays information on the tunnels.

Syntax:

```
tnip1+list ?
  state    Display the state of the tunnel connection
tnip1+
```

LIST STATE

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

Example:

```
tnip1+list state

Source IP       Dest. IP        STime Conn  DescR L IpMtu Head
---------       ----------      ----- ----- ----- - ----- ----
10.1.3.2        0.0.0.0               0     0     0 2108   28

tnip1+
```

STime: last connection start time.

Conn: number of connections since the last router startup.

DescR: number of disconnections as a result of not receiving routes through the tunnel.

L: reports if a loop has occurred.

IpMtu: IP MTU without considering the GRE header.

Head: size of the GRE header. *IpMtu* + *Head* must be less than the physical interface MTU.

4.2 IP tunnel interface statistics (TNIP)

Running the **device <TNIP interface>** command from the general monitoring process prompt (+), provides all the interface statistics for the corresponding TNIP:

```
+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
+
```

Command history:

Release	Modification
11.01.00	The command output has changed and Out-of-Order frames will not be shown as of version 11.01.00. This is due to the **sequence-datagrams** command being obsolete.

Chapter 5 IP Tunnel Configuration Examples

5.1 IP tunnel over IP

5.1.1 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. Add the IP route needed to reach the destination.
- Configure the tunnel encapsulation protocol (or tunnel type).
- Enable the desired options.
- Add 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.

5.1.2 Steps to follow for devices using the tunnel

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

5.1.3 Example 1.a: IP over IP with GRE tunneling

Tunnel configuration with *Router1* as source and *Router3* as destination, to allow networks 193.6.1.0/24 and 195.6.1.0/24 to communicate.

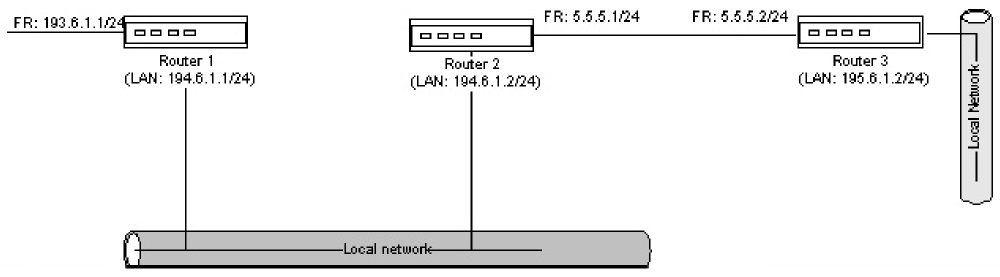


Fig. 2: **IP over IP with GRE**

5.1.3.1 Router1 configuration

Add the frame relay interface and 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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
Router1 ethernet0/0 config>IP ADDRESS 194.6.1.1 255.255.255.0
Router1 IP config>EXIT
Router1 Config>NETWORK ser0/0

-- Frame Relay user configuration --
Router1 serial0/0 config>IP ADDRESS 193.6.1.1 255.255.255.0
```

```
Router1 ser0/0 config>EXIT
Router1 config>NETWORK TNIP1

-- IP Tunnel Net Configuration --
Router1 tnip1 config>IP ADDRESS tnip1 unnumbered
Router1 tnip1 config>EXIT
Router1 config>PROTOCOL IP

-- Internet protocol user configuration --
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>
```

Then configure the IP tunnel:

```
Router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router1 tnip1 config>SOURCE 194.6.1.1
Router1 tnip1 config>DESTINATION 5.5.5.2
Router1 tnip1 config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 194.6.1.1, destination 5.5.5.2
QoS preclassify: disabled
Keepalive disabled
NHRP type of service: 0
Router1 tnip1 config>ENCAPSULATION

-- GRE Configuration --
Router1 tnip1 GRE config>CHECKSUM
Router1 tnip1 GRE config>KEY 1234
Router1 tnip1 GRE config>LIST
RC4 Cipher.................: disabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [1234]
Router1 tnip1 GRE config>EXIT
Router1 tnip1 config>EXIT
Router1 Config>
```

Add 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>
```

5.1.3.2 Router2 configuration

Add 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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
Router2 ethernet0/0 config>IP ADDRESS 194.6.1.2 255.255.255.0
Router2 ethernet0/0 config>EXIT
Router2 config>NETWORK serial0/0

-- Frame Relay user configuration --
Router2 serial0/0 FR config>IP ADDRESS 5.5.5.1 255.255.255.0
Router2 serial0/0 FR config>EXIT
Router2 Config>PROTOCOL IP

-- Internet protocol user configuration --
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>
```

5.1.3.3 Router3 configuration

Add the frame relay interface and 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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
Router3 ethernet0/0 config>IP ADDRESS 195.6.1.1 255.255.255.0
Router3 ethernet0/0 config>EXIT
Router3 config>NETWORK serial0/0

-- Frame Relay user configuration --
Router3 serial0/0 FR config>IP ADDRESS 5.5.5.2 255.255.255.0
Router3 serial0/0 FR config>EXIT
Router3 serial0/0 FR config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router3 tnip1 config>IP ADDRESS unnumbered
Router3 tnip1 config>EXIT
Router3 Config>PROTOCOL IP

-- Internet protocol user configuration --
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 tnip1 config>DESTINATION 194.6.1.1
Router3 tnip1 config>SOURCE 5.5.5.2
Router3 tnip1 config>ENCAPSULATION

-- GRE Configuration --
Router3 tnip1 GRE config>CHECKSUM
Router3 tnip1 GRE config>KEY 1234
Router3 tnip1 GRE config>EXIT
Router3 tnip1 config>EXIT
Router3 Config>
```

Add 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>
```

5.1.4 Example 1.b: Promiscuous tunnel

Tunnel configuration with *Router1* as source and *cxsec1* as destination, to allow a branch office RIP to be sent to the Centrix-Sec.

This scenario arises when we are looking to send IPSec encrypted traffic but are faced with the difficulty of sending RIP through an IPSec tunnel. One solution would be to configure a GRE tunnel enabled with RIP and send the encapsulated traffic through the IPSec tunnel. This would allow the device at the other end of the tunnel (possibly a Centrix-Sec) to receive both the encrypted data and the accessible networks through said tunnel.

The problem arises when the device receiving the RIP information serves multiple IPSec tunnels, in which case a TNIP interface would be required for each one. The way around this problem is to 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:

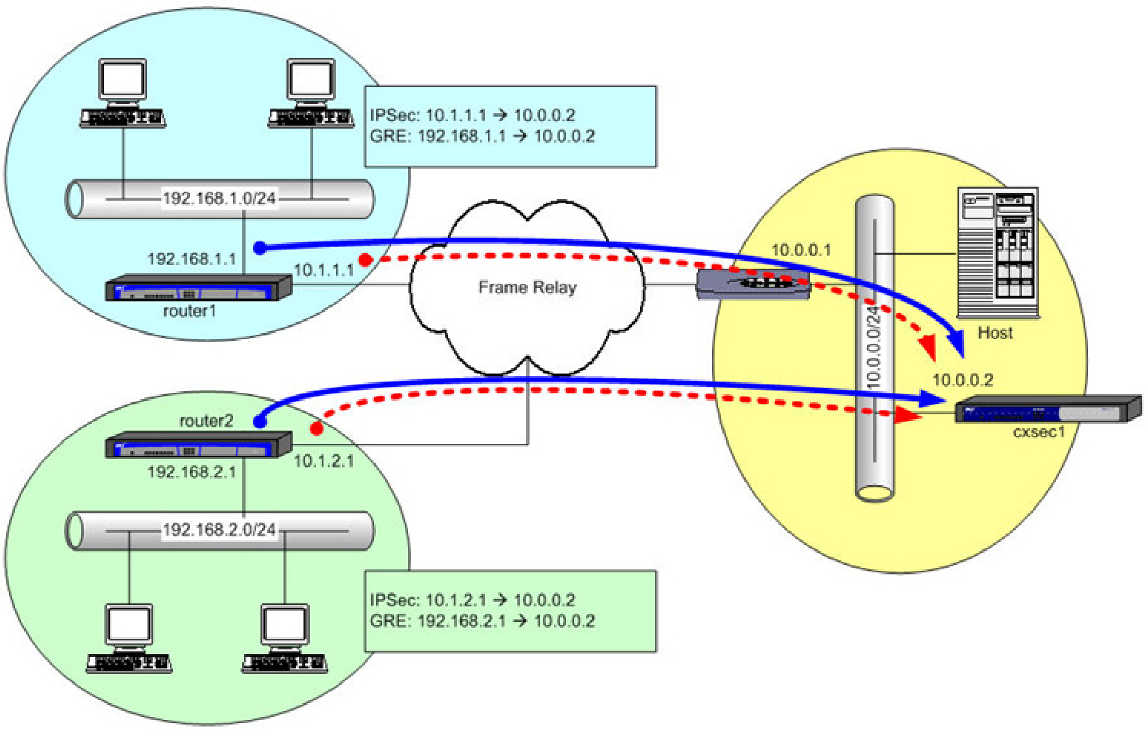


Fig. 3: **Promiscuous tunnel application example scenario**

5.1.4.1 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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
cxsec1 ethernet0/0 config>IP ADDRESS 10.0.0.2 255.255.255.0
cxsec1 ethernet0/0 config>EXIT
cxsec1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
cxsec1 tnip1 config>IP ADDRESS unnumbered 0.0.0.0
cxsec1 tnip1 config>EXIT
cxsec1 Config>PROTOCOL IP

-- Internet protocol user configuration --
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>
```

The source and destination addresses are not configured because this configuration is for a promiscuous tunnel. No other function is used in this example:

```
cxsec1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
cxsec1 tnip1 config>LIST
Tunnel mode: GRE (enabled)
Tunnel source unspecified, destination unspecified
QoS preclassify: disabled
Keepalive disabled
NHRP type of service: 0
cxsec1 tnip1 config>ENCAPSULATION

-- GRE Configuration --
cxsec1 tnip1 GRE config>LIST
RC4 Cipher.................: disabled
End-to-End Checksumming....: disabled
Tunnel identification key..: disabled
cxsec1 tnip1 GRE config>EXIT
cxsec1 tnip1 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>
```

Configure the RIP protocol to receive routing information via the TNIP interface and send it to the local network.

Finally, configure the IPSec encryption:

```
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 isakmp des md5
cxsec1 IPSec config>TEMPLATE 1 ike mode aggressive
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>
```

5.1.4.2 Router1 configuration

Add the frame relay and IP tunnel interfaces:

```
*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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
router1 ethernet0/0 config>IP ADDRESS 192.168.1.1 255.255.255.0
router1 ethernet0/0 config>EXIT
router1 config>NETWORK serial0/0

-- Frame Relay user configuration --
router1 serial0/0 FR config>IP ADDRESS 10.1.1.1 255.255.255.252
router1 serial0/0 FR config>EXIT
router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
router1 tnip1 config>IP ADDRESS unnumbered
router1 tnip1 config>EXIT
router1 Config>PROTOCOL IP

-- Internet protocol user configuration --
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:

```
router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
router1 tnip1 config>DESTINATION 10.0.0.2
router1 tnip1 config>SOURCE 192.168.1.1
router1 tnip1 config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 192.168.1.1, destination 10.0.0.2
QoS preclassify: disabled
Keepalive disabled
NHRP type of service: 0
router1 tnip1 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 to send local network information, accessible via 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 IPSec encryption:

```
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 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 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>
```

5.1.4.3 Router2 configuration

The configuration of *Router2* is similar to that of *Router1*. The main differences are 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 tnip1 config>DESTINATION 10.0.0.2
router2 tnip1 config>SOURCE 192.168.2.1
router2 tnip1 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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
router2 ethernet0/0 config>IP ADDRESS 192.168.2.1 255.255.255.0
router2 ethernet0/0 config>EXIT
router2 config>NETWORK serial0/0

-- Frame Relay user configuration --
router2 serial0/0 FR config>IP ADDRESS 10.1.2.1 255.255.255.252
router2 serial0/0 FR config>EXIT
router2 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
router2 tnip1 config>IP ADDRESS unnumbered 0.0.0.0
router2 tnip1 config>EXIT
router2 Config>PROTOCOL IP

-- Internet protocol user configuration --
router2 IP config>INTERNAL-IP-ADDRESS 192.168.2.1
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 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 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>
```

5.1.4.4 Final result

The final result is that both *router1* and *router2* send information from their local networks (192.168.1.0/24 and 192.168.2.0/24 respectively) using RIP, GRE encapsulation and IPSec encryption.

Cxsec1 receives all RIP information through the same tunnel interface (in promiscuous mode) and notifies its local network 10.0.0.0/24.

This way, all devices on the 10.0.0.0/24 network know they must go through *cxsec1* in order to send traffic to the *router1* and *router2* networks.*Cxsec1,* in turn, is responsible for encrypting said traffic and sending it through the corresponding IPSec tunnel.

The following diagram represents the flow of data and RIP information between *router1*, *router2* and *cxsec1* in their various encapsulations:

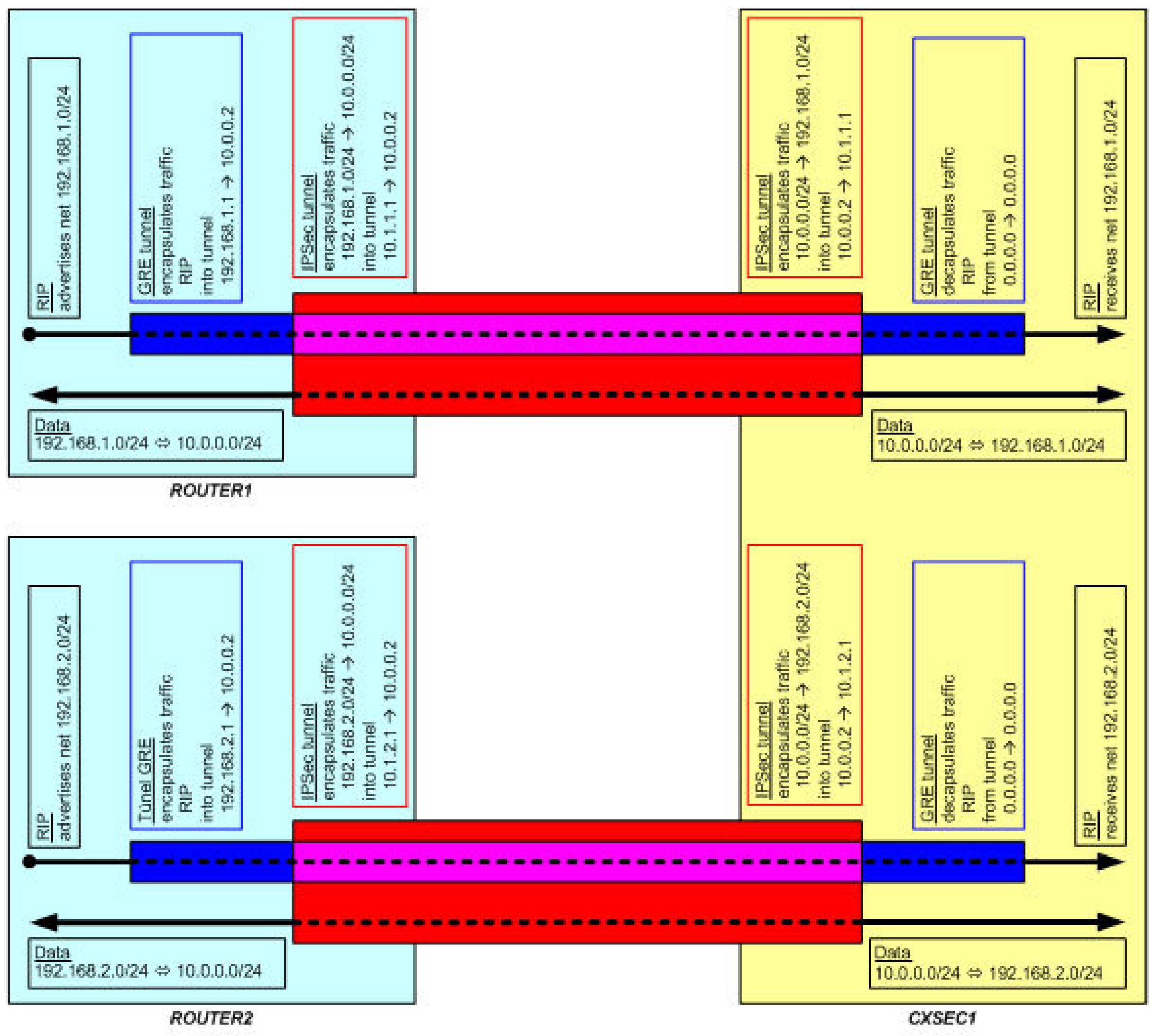


Fig. 4: **Encapsulation for the various traffic types in this scenario**

As you can see in the diagram, RIP traffic from *router1's* TNIP interface (upper black arrow) is only one-way, from *router1* to *cxsec1*. This traffic is encapsulated in the GRE tunnel (illustrated in blue), which, having source 192.168.2.1 and destination 10.0.0.2, will be encapsulated in turn in the IPSec tunnel (illustrated in red).

The *cxsec1* device obtains GRE traffic (blue) by deciphering and decapsulating the traffic it receives through the IPSec tunnel (red). As this traffic is going to *cxsec1* (address 10.0.0.2), it processes it in the TNIP interface, thus 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 shown in the diagram (bottom arrow).

Thanks to the TNIP interface operating in promiscuous mode, the *cxsec1* device can receive RIP information from all branch offices through one interface. This means you are not limited to a maximum of fifteen TNIP interfaces and can serve a high number of said offices.

5.2 IP tunnel over SRT

5.2.1 Steps to follow at each end of the tunnel

- Create the IP tunnel interface.
- Enable the bridge.
- Add a port to the bridge for the tunnel interface.
- Configure the tunnel source.
- Configure the tunnel destination. Add the IP route needed to reach the destination.
- Configure the tunnel encapsulation protocol (or tunnel type).
- Enable required options.

5.2.2 Steps to follow for devices using the tunnel

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

5.2.3 Example: IP over SRT with GRE

Tunnel configuration with *router1* as source and *router4* as destination, to allow networks 193.6.1.0/24 and 195.6.1.0/24 to communicate through NetBEUI traffic. To achieve this, create an IP tunnel over SRT between the two.

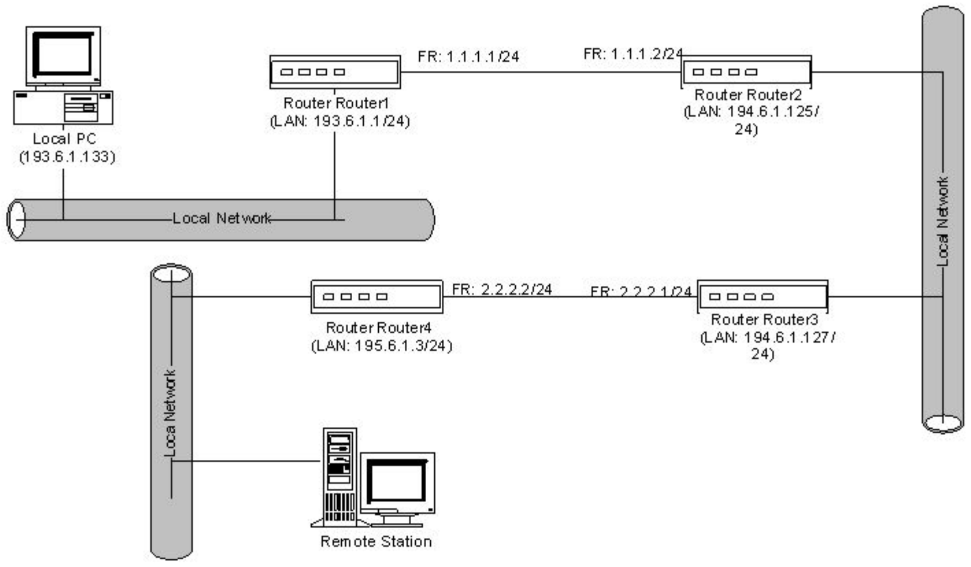


Fig. 5: **IP over SRT with GRE**

5.2.3.1 Router1 configuration

As with the above example, add the frame relay and IP tunnel (TNIP) interfaces and configure the interface IP addresses:

```
*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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
Router1 ethernet0/0 config>IP ADDRESS 193.6.1.133 255.255.255.0
Router1 ethernet0/0 config>EXIT
Router1 Config>NETWORK serial0/0
```

```

-- Frame Relay user configuration --
Router1 serial0/0 FR config>IP ADDRESS 1.1.1.1 255.255.255.0
Router1 serial0/0 FR config>EXIT
Router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router1 tnip1 config>IP ADDRESS unnumbered
Router1 tnip1 config>EXIT
Router1 Config>
```

Then configure the tunnel:

```
Router1 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router1 tnip1 config>DESTINATION 2.2.2.2
Router1 tnip1 config>SOURCE 1.1.1.1
Router1 tnip1 config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 1.1.1.1, destination 2.2.2.2
QoS preclassify: disabled
Keepalive disabled
NHRP type of service: 0
Router1 tnip1 config>ENCAPSULATION

-- GRE Configuration --
Router1 tnip1 config>CHECKSUM
Router1 tnip1 config>KEY 1234
Router1 tnip1 config>LIST
RC4 Cipher.................: disabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [1234]
Router1 tnip1 GRE config>EXIT
Router1 tnip1 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>
```

Add the necessary routes:

```
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>
```

Finally, 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>
```

5.2.3.2 Router2 and router3 configuration

We will assume that these devices have been correctly configured to provide IP connectivity.

5.2.3.3 Router4 configuration

The frame relay and the IP tunnel (TNIP) interfaces are added in a similar way to that shown in the above example for *router1*:

```
*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>NETWORK ethernet0/0

-- Ethernet Interface User Configuration --
Router4 ethernet0/0 config>IP ADDRESS 195.6.1.3 255.255.255.0
Router4 ethernet0/0 config>EXIT
Router4 Config>NETWORK serial0/0

-- Frame Relay user configuration --
Router4 serial0/0 FR config>IP ADDRESS 2.2.2.2 255.255.255.0
Router4 serial0/0 FR config>EXIT
Router4 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router4 tnip1 config>IP ADDRESS unnumbered
Router4 tnip1 config>EXIT
Router4 Config>PROTOCOL IP
```

```
-- Internet protocol user configuration --
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>
```

Now configure the tunnel:

```
Router4 Config>NETWORK tnip1

-- IP Tunnel Net Configuration --
Router4 tnip1 config>DESTINATION 1.1.1.1
Router4 tnip1 config>SOURCE 2.2.2.2
Router4 tnip1 config>LIST
Tunnel mode: GRE (enabled)
Tunnel source 2.2.2.2, destination 1.1.1.1
QoS preclassify: disabled
Keepalive disabled
NHRP type of service: 0
Router4 tnip1 config>ENCAPSULATION

-- GRE Configuration --
Router4 tnip1 GRE config>CHECKSUM
Router4 tnip1 GRE config>KEY 1234
Router4 tnip1 GRE config>LIST
RC4 Cipher................: disabled
End-to-End Checksumming....: enabled
Tunnel identification key..: enabled [1234]
Router4 tnip1 GRE config>EXIT
Router4 tnip1 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>
```

Finally, 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>
```

5.3 IPV6 tunnel over IPV6

5.3.1 Steps to follow at the end of the tunnel

- Create the IP tunnel.
- Configure the tunnel encapsulation protocol (or tunnel type).
- Enable IPv6 on the tunnel interface (or configure an IPv6 address).
- Configure the tunnel source.
- Configure the tunnel destination. Add the IPv6 route needed to reach the destination.
- Enable the required options.
- Add the IPv6 routes from those networks, which need to be accessible through the IP tunnel, placing the IP tunnel's interface as the next hop.

5.3.2 Steps to follow for devices using the tunnel

- Add the routes needed so that the tunnel source and destination are accessible.

5.3.3 Example 3: IPv6 over IPv6 with GRE

Tunnel configuration with *router1* as source and *router2* as destination, to allow networks 2001:db8:1::/64 and 2001:db8:2::/64 to communicate.

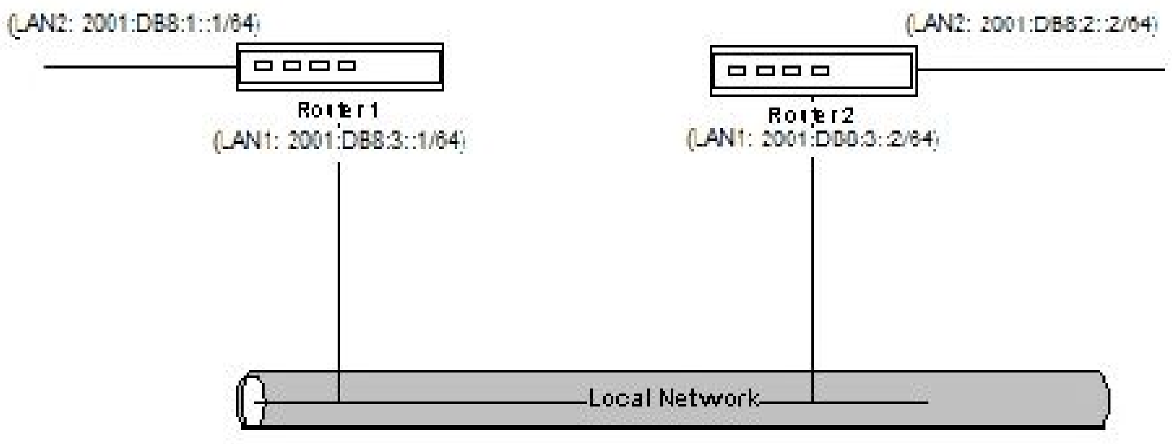


Fig. 6: **IPv6 over IPv6 with GRE**

5.3.3.1 Configuring router1

Add the IP tunnel interface:

```
*P 4
Config>set hostname Router1
Router1 Config>add device tnip 1
Added TNIP interface tnip1
Router1 Config>
```

Configure the interface addresses:

```
Router1 Config>network ethernet0/0

-- Ethernet Interface User Configuration --
Router1 ethernet0/0 config>ipv6 address 2001:db8:3::1/64
Router1 IP config>exit
Router1 Config>network ethernet0/1
```

```
-- Ethernet Interface User Configuration --
Router1 ethernet0/1 config>ipv6 address 2001:db8:1::1/64
Router1 IP config>exit
Router1 config>
```

Next, configure the IP tunnel:

```
Router1 config>network tnip1

-- IP Tunnel Net Configuration --
Router1 tnip1 config>mode gre ipv6
Router1 tnip1 config>ipv6 enable
Router1 tnip1 config>source 2001:db8:3::1
Router1 tnip1 config>destination 2001:db8:3::2
Router1 tnip1 config>list
Tunnel mode: GRE over ipv6 (enabled)
Tunnel source 2001:db8:3::1, destination 2001:db8:3::2
QoS preclassify: disabled
Keepalive disabled
NHRP type of service: 0
Router1 tnip1 config>encapsulation

-- GRE Configuration --
Router1 tnip1 GRE config>key 1234
Router1 tnip1 GRE config>list
RC4 Cipher................: disabled
End-to-End Checksumming....: disabled
Tunnel identification key..: enabled [1234]
Router1 tnip1 GRE config>exit
Router1 tnip1 config>exit
Router1 Config>
```

Add the necessary routes:

```
Router1 Config>protocol ipv6

-- IPv6 user configuration --
Router1 IPv6 config> route 2001:db8:2::/64 interface tnip1
Router1 IPv6 config>exit
Router1 Config>
```

5.3.3.2 Configuring router2

Add the IP tunnel interface:

```
*P 4
Config>set hostname Router2
Router2 Config>add device tnip 1
Added TNIP interface tnip1
Router2 Config>
```

Configure the interface addresses:

```
Router2 Config>network ethernet0/0

-- Ethernet Interface User Configuration --
Router2 ethernet0/0 config>ipv6 address 2001:db8:3::2/64
Router2 IP config>exit
Router2 Config>network ethernet0/1

-- Ethernet Interface User Configuration --
Router2 ethernet0/1 config>ipv6 address 2001:db8:2::2/64
Router2 IP config>exit
Router2 config>
```

Next, configure the IP tunnel:

```
Router2 config>network tnip1

-- IP Tunnel Net Configuration --
Router2 tnip1 config>mode gre ipv6
Router2 tnip1 config>ipv6 enable
Router2 tnip1 config>source 2001:db8:3::2
Router2 tnip1 config>destination 2001:db8:3::1
Router2 tnip1 config>list
Tunnel mode: GRE over ipv6 (enabled)
Tunnel source 2001:db8:3::2, destination 2001:db8:3::1
QoS preclassify: disabled
Keepalive disabled
NHRP type of service: 0
Router2 tnip1 config>encapsulation

-- GRE Configuration --
Router2 tnip1 GRE config>key 1234
Router2 tnip1 GRE config>list
RC4 Cipher.................: disabled
End-to-End Checksumming....: disabled
Tunnel identification key..: enabled [1234]
Router2 tnip1 GRE config>exit
Router2 tnip1 config>exit
Router2 Config>
```

Add the necessary routes:

```
Router2 Config>protocol ipv6

-- IPv6 user configuration --
Router2 IPv6 config> route 2001:db8:1::/64 interface tnip1
Router2 IPv6 config>exit
Router2 Config>
```