

## Chapter 23

# Configuring Frame Relay

The Frame Relay protocol allows network designers to reduce costs by using shared facilities that are managed by a Frame Relay service provider. Users pay fixed charges for the local connections from each site in the Frame Relay network to the first point of presence (POP) in which the provider maintains a Frame Relay switch. The portion of the network between the endpoint switches is shared by all the customers of the service provider, and individual data-link connection identifiers (DLCIs) are assigned to ensure each customer receives only their own traffic.

Users contract with their providers for a specific minimum portion of the shared bandwidth Committed Information Rate (CIR) and for a maximum allowable peak rate, Burst Information Rate (BIR). Depending on the terms of the contract, traffic exceeding the CIR can be marked as eligible for discard, in the event of network congestion, or a best effort term can apply up to the BIR rate.

Frame Relay does not require private and permanently connected wide area network facilities, unlike some older WAN protocols.

Frame Relay was developed as a replacement for the older and much slower X.25 protocol. It scales to much higher data rates because it does not require explicit acknowledgment of each frame of data.

You can configure the Frame Relay protocol on SONET/SDH, E1/E3, and T1/T3 physical routing platform interfaces, and on the channelized DS3, channelized OC12, channelized T3 intelligent queuing (IQ), channelized OC12 IQ, and channelized E1 IQ interfaces.

This chapter discusses configuration of the following Frame Relay properties:

Configuring Frame Relay Interface Encapsulation on page 398

Configuring Frame Relay Control Bit Translation on page 402

Configuring the Media MTU on page 403

Setting the Protocol MTU on page 404

Configuring Frame Relay Keepalives on page 404

Configuring Inverse Frame Relay ARP on page 406

Configuring the Router as a DCE on page 406

Configuring Frame Relay DLCIs on page 407

## Configuring Frame Relay Interface Encapsulation

---

Point-to-Point Protocol (PPP) encapsulation is the default encapsulation type for physical interfaces. You need not configure encapsulation for any physical interfaces that support PPP encapsulation. If you do not configure encapsulation, PPP is used by default. For physical interfaces that do not support PPP encapsulation, you must configure an encapsulation to use for packets transmitted on the interface. You can optionally configure an encapsulation on a logical interface, which is the encapsulation used within certain packet types.

For more information, see the following sections:

Configuring the Frame Relay Encapsulation on a Physical Interface on page 398

Configuring the Frame Relay Encapsulation on a Logical Interface on page 401

### ***Configuring the Frame Relay Encapsulation on a Physical Interface***

For Frame Relay interfaces, you configure Frame Relay encapsulation on the physical interface. This encapsulation is defined in RFC 1490, *Multiprotocol Interconnect over Frame Relay*. SONET/SDH and T3 interfaces can use Frame Relay encapsulation.

To configure Frame Relay encapsulation on a physical interface, include the encapsulation statement at the [edit interfaces *interface-name*] hierarchy level:

```
[edit interfaces interface-name]  
  encapsulation type;
```

When you configure a multipoint encapsulation (such as Frame Relay), the physical interface can have multiple logical units, and the units can be either point-to-point or multipoint.

The encapsulation type can be one of the following:

**Flexible Frame Relay (flexible-frame-relay)**—IQ interfaces can use flexible Frame Relay encapsulation. You use flexible Frame Relay encapsulation when you want to configure multiple per-unit Frame Relay encapsulations. This encapsulation type allows you to configure any combination of TCC, CCC, and standard Frame Relay encapsulations on a single physical port. Also, each logical interface can have any DLCI value from 1 through 1022.

**Frame Relay (frame-relay)**—Defined in RFC 1490. E1, E3, link services, SONET/SDH, T1, T3, and voice services interfaces can use Frame Relay encapsulation. Five related versions are supported:

**Circuit cross-connect (CCC) version (frame-relay-ccc)**—The same as standard Frame Relay for DLCIs 0 through 511. DLCIs 512 through 1022 are dedicated to CCC. This numbering restriction does not apply to IQ interfaces. The logical interface must also have frame-relay-ccc encapsulation. When you use this encapsulation type, you can configure the ccc family only.

**Translational cross-connect (TCC) version (frame-relay-tcc)**—Similar to Frame Relay CCC and has the same configuration restrictions, but used for circuits with different media on either side of the connection.

**Extended CCC version (extended-frame-relay-ccc)**—This encapsulation type allows you to dedicate DLCIs 1 through 1022 to CCC. The logical interface must have frame-relay-ccc encapsulation. When you use this encapsulation type, you can configure the ccc family only.

**Extended TCC version (extended-frame-relay-tcc)**—Similar to extended Frame Relay CCC, this encapsulation type allows you to dedicate DLCIs 1 through 1022 to TCC, which is used for circuits with different media on either side of the connection.

**Port CCC version (frame-relay-port-ccc)**—Defined in the Internet Engineering Task Force (IETF) document, *Frame Relay Encapsulation over Pseudo-Wires* (expired December 2002). This encapsulation type allows you to transparently carry all the DLCIs between two customer edge (CE) routers without explicitly configuring each DLCI on the two provider edge (PE) routers with Frame Relay transport. The connection between the two CE routers can be either user-to-network interface (UNI) or network-to-network interface (NNI); this is completely transparent to the PE routers. The logical interface does not require an encapsulation statement. When you use this encapsulation type, you can configure the ccc family only.

Support for extended Frame Relay and flexible Frame Relay differs by PIC type, as shown in Table 34.

**Table 34: PIC Support for Enhanced Frame Relay Encapsulation Types**

PIC Type	Extended Frame Relay CCC	Extended Frame Relay TCC	Flexible Frame Relay
<b>Intelligent Queuing</b>			
1-port Channelized CHOC12 IQ	Yes	Yes	Yes
4-port Channelized DS3 IQ	Yes	Yes	Yes
10-port Channelized E1 IQ	Yes	Yes	Yes
4-port E3 IQ	Yes	Yes	Yes
1-port Channelized STM1 IQ	Yes	Yes	Yes
<b>SONET/SDH</b>			
1-port OC12	Yes	Yes	No
2-port OC3	Yes	Yes	No
1-port OC48	Yes	Yes	No
1-port OC192	Yes	Yes	No
1-port STM16 SDH, SMSR	Yes	Yes	No
<b>Others</b>			
4-port E1	No	No	No
4-port T1	No	No	No
4-port T3	No	No	No
10-port Channelized E1	No	No	No
2-port Channelized DS3	No	No	No
1-port Channelized OC12, SMIR	No	No	No
4-port Channelized DS3	No	No	No
1-port Channelized STM1, SMIR	No	No	No
2-port Serial	No	No	No

**Example: Configuring the Encapsulation on a Physical Interface**

Configure Frame Relay encapsulation on a SONET/SDH interface. The second and third family statements allow Intermediate System-to-Intermediate System (IS-IS) and Multiprotocol Label Switching (MPLS) to run on the interface.

```
[edit interfaces]
so-7/0/0 {
  encapsulation frame-relay;
  unit 0 {
    point-to-point;
    family inet {
      address 192.168.1.113/32 {
        destination 192.168.1.114;
      }
    }
    family iso;
    family mpls;
  }
}
```

**Configuring the Frame Relay Encapsulation on a Logical Interface**

Generally, you configure an interface's encapsulation at the [edit interfaces *interface-name*] hierarchy level. However, for Frame Relay encapsulation, you can also configure the encapsulation type that is used inside the Frame Relay packet itself. To do this, include the encapsulation statement, specifying the `frame-relay-ccc` or `frame-relay-tcc` option:

```
encapsulation (frame-relay-ccc | frame-relay-tcc);
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number]
```

## Configuring Frame Relay Control Bit Translation

---

On interfaces with Frame Relay CCC encapsulation, you can configure Frame Relay control bit translation, as defined in the IETF documents:

Internet draft draft-martini-frame-encap-mpls-00.txt, *Frame Relay Encapsulation over Pseudo-Wires* (expired December 2002)

Internet draft draft-martini-l2circuit-encap-mpls-07.txt, *Encapsulation Methods for Transport of Layer 2 Frames Over IP and MPLS Networks* (expires December 2004)

To support Frame Relay services over IP and MPLS backbones using Layer 2 VPNs and Layer 2 circuits, you can configure translation of the Frame Relay control bits. When you configure translation of Frame Relay control bits, the bits are mapped into the Layer 2 circuit control word and preserved across the IP or MPLS backbone.

The JUNOS software allows you to translate the following Frame Relay control bits:

Discard eligibility (DE)—A header bit used to identify lower priority traffic that can be dropped during periods of congestion.

Forward explicit congestion notification (FECN)—A header bit transmitted by the source routing platform requesting that the destination routing platform slow down its requests for data.

Backward explicit congestion notification (BECN)—A header bit transmitted by the destination routing platform requesting that the source routing platform send data more slowly.

By default, translation of Frame Relay control bits is disabled. If you enable Frame Relay control bit translation, the bits are translated in both directions (CE to PE and PE to CE):

From CE to PE—At ingress, the DE, FECN, and BECN header bits from the incoming Frame Relay header are mapped to the control word.

From PE to CE—At egress, the DE, FECN, and BECN header bits from the control word are mapped to the outgoing Frame Relay header.

The Frame Relay control bits do not map to MPLS EXP labels, and do not affect class-of-service (CoS) behavior inside the provider network.

You enable or explicitly disable translation of Frame Relay control bits by including the `translate-discard-eligible` and `translate-fecn-and-becn` statements:

```
(translate-discard-eligible | no-translate-discard-eligible);
(translate-fecn-and-becn | no-translate-fecn-and-becn);
```

You can include these statements at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number family ccc]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number family ccc]
```

If you enable or disable Frame Relay control bit translation on one CE-facing interface, you must configure the same Frame Relay control bit translation settings on the other CE-facing interface.

If you change the Frame Relay control bit translation settings, the circuit goes down and comes back up, which might result in traffic loss for a few seconds.

If you enable Frame Relay control bit translation, the number of supportable Layer 2 virtual private networks (VPNs) and Layer 2 circuits is reduced to one eighth of what the routing platform can support without Frame Relay control bit translation enabled.

For ATM2 IQ interfaces, the control word contains a field to carry ATM cell loss priority (CLP) information by default. For more information, see “Configuring Layer 2 Circuit Transport Mode” on page 194.

For more information about Layer 2 circuits, see the *JUNOS VPNs Configuration Guide* and the *JUNOS Routing Protocols Configuration Guide*. For a comprehensive example, see the *JUNOS Feature Guide*.

## Configuring the Media MTU

---

For Frame Relay interfaces, the default media maximum transmission unit (MTU) is 4482 bytes. (For a complete list of MTU values, see “Configuring the Media MTU” on page 67.)

To modify the default media MTU size for a physical interface, include the `mtu` statement at the [edit interfaces *interface-name*] hierarchy level:

```
[edit interfaces interface-name]  
  mtu bytes;
```

If you change the size of the media MTU, you must ensure that the size is equal to or greater than the sum of the protocol MTU and the encapsulation overhead. You can include the `mtu` statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number family family]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit  
  logical-unit-number family family]
```

For more information, see “Setting the Protocol MTU” on page 404.

## Setting the Protocol MTU

---

For each interface, you can configure an interface-specific MTU by including the `mtu` statement at the [edit interfaces interface *interface-name*] hierarchy level. If you need to modify this MTU for a particular protocol family, include the `mtu` statement:

```
mtu mtu;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number family family]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number family family]
```

For Frame Relay encapsulation, the default protocol MTU is 4470 bytes.

If you increase the size of the protocol MTU, you must ensure that the size of the media MTU is equal to or greater than the sum of the protocol MTU and the encapsulation overhead. (You configure the media MTU by including the `mtu` statement at the [edit interfaces *interface-name*] hierarchy level, as discussed in “Configuring the Media MTU” on page 403.)

When the family is `mpls`, the default protocol MTU is 1488 bytes. MPLS packets are 1500 bytes and have 4 to 12 bytes of overhead.

## Configuring Frame Relay Keepalives

---

By default, physical interfaces configured with Cisco High-level Data Link Control (HDLC) or Point-to-Point Protocol (PPP) encapsulation send keepalive packets at 10-second intervals. The Frame Relay term for keepalives is Local Management Interface (LMI) packets; note that the JUNOS software supports both ANSI T1.617 Annex D LMIs and ITU Q933 Annex A LMIs.

To disable the sending of keepalives on a physical interface, include the `no-keepalives` statement at the [edit interfaces interface *interface-name*] hierarchy level:

```
[edit interfaces interface-name]
no-keepalives;
```

For back-to-back Frame Relay connections, either disable the sending of keepalives on both sides of the connection, or configure one side of the connection as a data terminal equipment (DTE) (the default JUNOS configuration) and the other as a data circuit-terminating equipment (DCE).

If keepalives are enabled, the number of possible DLCI configurations on a multipoint or multicast connection is limited by the MTU size selected for the interface. To calculate the available DLCIs, use the formula  $(MTU - 12) / 5$ . To increase the number of possible DLCIs, disable keepalives.

## Configuring Tunable Keepalives for Frame Relay LMI

On interfaces configured with Frame Relay connections, you can tune the keepalive settings by using the `lmi` statement. A Frame Relay interface can be either DCE or DTE (the default JUNOS configuration). DTE acts as a master, requesting status from the DCE part of the link.

By default, the JUNOS software uses ANSI T1.617 Annex D LMIs. To change to ITU Q933 Annex A LMIs, include the `lmi-type itu` statement at the `[edit interfaces interface-name lmi]` hierarchy level:

```
[edit interfaces interface-name lmi]
lmi-type itu;
```

To configure Frame Relay keepalive parameters, include the `lmi` statement at the `[edit interfaces interface-name]` hierarchy level:

```
[edit interfaces interface-name]
lmi {
  lmi-type (ansi | itu);
  n391dte number;
  n392dce number;
  n392dte number;
  n393dce number;
  n393dte number;
  t391dte seconds;
  t392dce seconds;
}
```

You can include the following statements:

**n391dte**—DTE full status polling interval. The DTE sends a status inquiry to the DCE at the interval specified by `t391dte`. `n391dte` specifies the frequency at which these inquiries expect a full status report; for example, a `n391dte` value of 10 would specify a full status report in response to every tenth inquiry. The intermediate inquiries ask for a keepalive exchange only. The range is from 1 through 255, with a default value of 6.

**n392dce**—DCE error threshold. The number of errors required to bring down the link, within the event-count specified by `n393dce`. The range is from 1 through 10, with a default value of 3.

**n392dte**—DTE error threshold. The number of errors required to bring down the link, within the event-count specified by `n393dte`. The range is from 1 through 10, with a default value of 3.

**n393dce**—DCE monitored event-count. The range is from 1 through 10, with a default value of 4.

**n393dte**—DTE monitored event-count. The range is from 1 through 10, with a default value of 4.

t391dte—DTE keepalive timer. Period at which the DTE sends out a keepalive response request to the DCE and updates status depending on the DTE error threshold value. The range is from 5 through 30 seconds, with a default value of 10 seconds.

t392dce—DCE keepalive timer. Period at which the DCE checks for keepalive responses from the DTE and updates status depending on the DCE error threshold value. The range is from 5 through 30 seconds, with a default value of 15 seconds.

## Configuring Inverse Frame Relay ARP

---

Frame Relay interfaces support inverse Frame Relay ARP, as described in RFC 2390, *Inverse Address Resolution Protocol*. When inverse Frame Relay ARP is enabled, the routing platform responds to received inverse Frame Relay ARP requests by providing IP address information to the requesting routing platform on the other end of the Frame permanent virtual circuit (PVC).

The routing platform does not initiate inverse Frame Relay ARP requests.

By default, inverse Frame Relay ARP is disabled. To configure a routing platform to respond to inverse Frame Relay ARP requests, include the `inverse-arp` statement:

```
inverse-arp;
```

For a list of hierarchy levels at which you can include this statement, see `inverse-arp` on page 676.

You must configure Frame Relay encapsulation on the logical interface to support inverse ARP. For more information, see “Configuring Frame Relay Interface Encapsulation” on page 398.

## Configuring the Router as a DCE

---

By default, when you configure an interface with Frame Relay encapsulation, the routing platform is assumed to be DTE. That is, the routing platform is assumed to be at a terminal point on the network. To configure the routing platform to be DCE, include the `dce` statement at the `[edit interfaces interface-name]` hierarchy level:

```
[edit interfaces interface-name]  
dce;
```

When you configure the routing platform to be a DCE, keepalives are disabled by default.

For back-to-back Frame Relay connections, either disable the sending of keepalives on both sides of the connection, or configure one side of the connection as a DTE (the default JUNOS configuration) and the other as a DCE.

## Configuring Frame Relay DLCIs

---

When you are using Frame Relay encapsulation on an interface, each logical interface corresponds to one or more permanent virtual circuits (PVCs) or switched virtual circuits (SVCs). For each PVC or SVC, you must configure one data-link connection identifier (DLCI).

A Frame Relay interface can be a point-to-point interface or a point-to-multipoint (also called a multipoint nonbroadcast multiaccess [NBMA]) connection.

To configure Frame Relay DLCIs, you can do the following:

Configuring a Point-to-Point Frame Relay Connection on page 407

Configuring a Point-to-Multipoint Frame Relay Connection on page 408

Configuring a Multicast-Capable Frame Relay Connection on page 408

### **Configuring a Point-to-Point Frame Relay Connection**

To configure a point-to-point Frame Relay connection, include the `dlci` statement:

```
dlci dlci-identifier;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number]
```

The DLCI identifier is a value from 16 through 1022. Numbers 1 through 15 are reserved for future use. A point-to-point interface can have one DLCI.



**NOTE:** For information about Frame Relay DLCI limitations for channelized interfaces, see “Data-Link Connection Identifiers on Channelized Interfaces” on page 249.

You configure the routing platform to use DLCI sparse mode by including the `sparse-dlcis` statement at the `[edit chassis fpc slot-number pic pic-number]` hierarchy level. For more information about DLCI sparse mode, see the *JUNOS System Basics Configuration Guide*.

For more information about Frame Relay DLCIs, see “Configuring a Point-to-Point Frame Relay Connection” on page 407.

---

When you are configuring point-to-point connections, the MTU sizes on both sides of the connection must be the same.

## Configuring a Point-to-Multipoint Frame Relay Connection

To configure a point-to-multipoint Frame Relay connection (also called a multipoint NBMA connection), include the multipoint-destination statement:

```
multipoint-destination address dlcidlcid-identifier;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number address address]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number address address]
```

*address* is the interface's address.

For each destination, include one multipoint-destination statement. *address* is the address of the remote side of the connection, and *dlcid-identifier* is the DLCI identifier for the connection.

When you are configuring point-to-multipoint connections, all interfaces in the subnet must use the same MTU size.

If keepalives are enabled, causing the interface to send LMI messages during idle times, the number of possible DLCI configurations is limited by the MTU selected for the interface. For more information, see "Configuring Frame Relay Keepalives" on page 404.

## Configuring a Multicast-Capable Frame Relay Connection

By default, Frame Relay connections assume unicast traffic. If your Frame Relay switch performs multicast replication, you can configure the connection to support multicast traffic by including the multicast-dlci statement:

```
multicast-dlci dlcidlcid-identifier;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number]
```

The DLCI identifier is a value from 16 through 1022 that defines the Frame Relay DLCI over which the switch expects to receive multicast packets for replication.

You can configure multicast support only on point-to-multipoint Frame Relay connections.

If keepalives are enabled, causing the interface to send LMI messages during idle times, the number of possible DLCI configurations is limited by the MTU selected for the interface. For more information, see "Configuring Frame Relay Keepalives" on page 404.