

# Configuring Frame Relay

# 11

This chapter describes how to configure a Frame Relay interface on your ERX system.

Topic	Page
Overview	11-1
References	11-3
Before You Configure Frame Relay	11-4
Configuring Frame Relay	11-4
End-to-End Fragmentation and Reassembly	11-10
Monitoring Frame Relay	11-15

## Overview

---

Frame Relay is a public, connection-oriented packet service based on the core aspects of Integrated Services Digital Network (ISDN). Frame Relay eliminates all processing at the network layer and greatly restricts data-link layer processing. Such simplified processing is possible because of the availability of virtually error-free physical connections and the presence of intelligent protocols at the end-user site, which can detect and retransmit faulty or discarded packets.

Frame Relay shifts responsibility for error recovery and flow control to the end user, thereby reducing protocol complexity and allowing high-speed packet delivery with low transit delay.

You can configure Frame Relay on the following modules:

- CT3
- CT1 and CE1
- cOCx/STMx
- T3-Frame and E3-Frame
- OCx/STMx POS

### *Framing*

The system supports the following framing features:

- HDLC for data-link framing
- 2-byte addresses only
- 8188-byte information field size (8192 minus 2 bytes for the address and a 16-bit CRC) or 8186-byte information field size (8192 minus 2 bytes for the address and a 32-bit CRC)

The system does not support:

- Protocol-dependent fragmentation
- Autodetection of Local Management Interface (LMI) protocol type

### *Error Frames*

The system relies on higher-layer protocols to detect and recover from Frame Relay data loss. All Frame Relay error frames are discarded.

### *Unicast and Multicast Addressing*

Most Frame Relay services support both unicast (individual) and multicast (group) addressing. Under the most common implementation of multicasting, the Frame Relay network maps multiple individual addresses to a single multicast DLCI and delivers copies of a single Frame Relay packet to each member of the group.



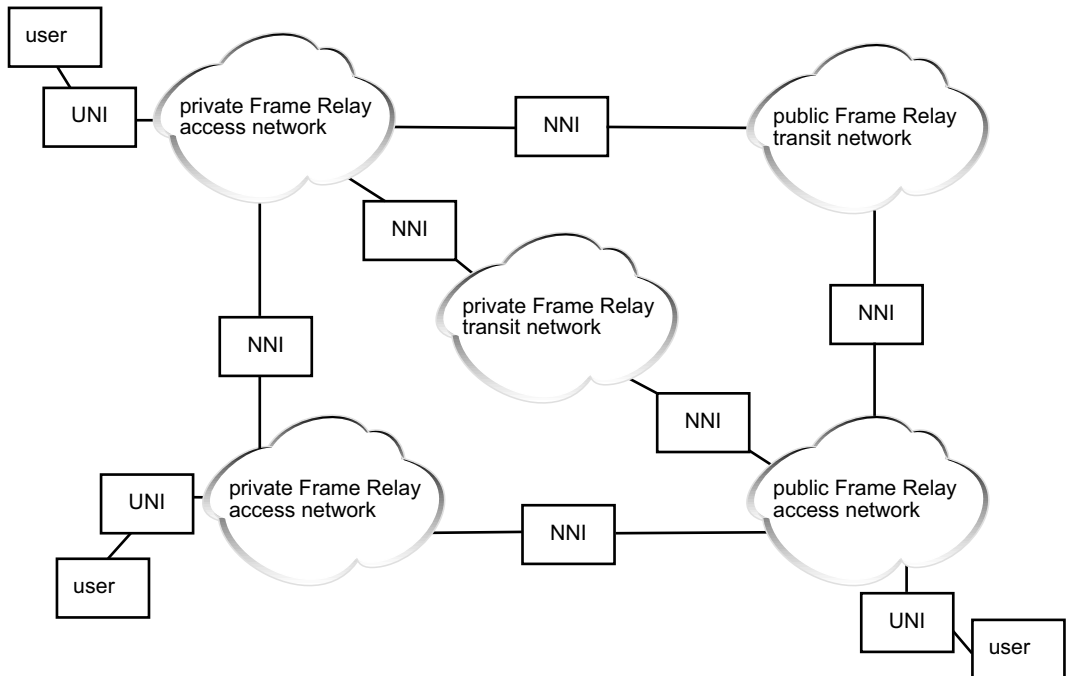
**Note:** *The system supports only unicast addressing.*

### *User-to-Network and Network-to-Network Interfaces*

Frame Relay User-to-Network Interface (UNI) is a protocol that permits users to access private or public Frame Relay networks and to establish a communications path to another user within the same network.

Network-to-Network Interface (NNI) makes connections possible between users connected to different Frame Relay networks. These separate Frame Relay networks can be considered as subnetworks within a complete network service.

Figure 11-1 shows the interconnection of these types of subnetworks and the location of NNI between them.



**Figure 11-1** Interconnection and relationship of NNIs and subnetworks

## References

For more information about Frame Relay, consult the following:

- RFC 2115 – Management Information Base for Frame Relay DTEs Using SMIV2 (September 1997)
- RFC 2863 – The Interfaces Group MIB (June 2000)
- RFC 2427 – Multiprotocol Interconnect over Frame Relay (September 1998)
- Frame Relay Forum – User-to-Network Implementation Agreement (UNI), FRF 1.1 (January 1996)

- Frame Relay Forum – Frame Relay Fragmentation Implementation Agreement, FRF.12 (December 1997)
- CCITT ITU-T Recommendation Q.922, Integrated Services Digital Network (ISDN) Data Link Layer Specification for Frame Mode Bearer Services; Annex A (February 1992)

## Before You Configure Frame Relay

---

Before you attempt to configure a Frame Relay interface, you should configure the physical line interface over which Frame Relay traffic flows.

This process is described in the following chapters:

- *Chapter 1, Configuring Channelized T3 Interfaces*
- *Chapter 2, Configuring T3 and E3 Interfaces*
- *Chapter 3, Configuring CT1 and CE1 Interfaces*

The procedures described in this chapter assume that a physical interface has been configured.

## Configuring Frame Relay

---

Configure a Frame Relay interface by entering Interface Configuration mode. The procedure that follows is an example of a Frame Relay configuration on a serial interface. All tasks are mandatory unless otherwise noted.

- 1 From Configuration mode, enter the physical interface on which you want to configure Frame Relay.

```
host1(config)#interface serial 3/1:2/1
```

- 2 Select Frame Relay as the encapsulation method for the interface.

```
host1(config-if)#encapsulation frame-relay ietf
```

- 3 (Optional) Add a description to the interface.

```
host1(config-if)#frame-relay description abcd1234
```

- 4 Configure the interface as a DTE, DCE, or NNI.

```
host1(config-if)#frame-relay intf-type dte
```

- 5 Configure the LMI type.

```
host1(config-if)#frame-relay lmi-type ansi
```

- 6 (Optional) Configure Frame Relay counters and timers.

```
host1(config-if)#frame-relay lmi-n391dte 20
```

- 7 Configure the cyclic redundancy check (CRC).

```
host1(config-if)#crc 32
```

- 8 Create a subinterface.

```
host1(config-if)#interface serial 3/1:2/1.1
```

- 9 (Optional) Add a description to the subinterface.

```
host1(config-subif)#frame-relay description abcd12345678
```

- 10 Add a circuit to a subinterface.

```
host1(config-subif)#frame-relay interface-dlci 17 ietf
```

- 11 Assign a local IP address to the circuit.

```
host1(config-subif)#ip address 192.32.10.2 255.255.255.0
```

- 12 Use **show** commands to verify that your configuration changes are correct by checking the state of the interfaces.

```
host1#show frame-relay lmi
```

```
host1#show frame-relay map
```

```
host1#show frame-relay pvc
```

- 13 (Optional) Disable the local management interface.

```
host1#no frame-relay keepalive
```

- 14 (Optional) Disable the interface.

```
host1(config-if)#shutdown
```

### **crc**

- Use to set the number of bits used for CRC.
- The CRC is an error-checking technique that uses a calculated numeric value to detect errors in transmitted data.
- 16 and 32 indicate the number of bits per frame that are used to calculate the frame check sequence (FCS).
- A 32-bit CRC transmits longer streams at faster rates and therefore provides better ongoing error detection, such as for an OC12/STM4 POS module.
- The default is 16. You must configure CRC (CRC16 or CRC32) to match the configuration on the other side of the Frame Relay connection.
- Example

```
host1(config-if)#crc 32
```
- Use the **no** version to set the CRC to the default value.

### ***encapsulation ietf***

- Use to specify Frame Relay as the encapsulation method for the interface.
- The system uses IETF format (RFC 2427 encapsulation).
- Example

```
host1(config-if)#encapsulation frame-relay ietf
```
- Use the **no** version to remove Frame Relay configuration from an interface.

### ***frame-relay description***

- Use to assign a text description or alias to a Frame Relay major interface or subinterface.
- Description name can be up to 64 characters.
- Examples

```
host1(config-if)#frame-relay description abcd1234
host1(config-subif)#frame-relay description abcd12345678
```
- Use the **no** version to remove the text description or alias from the Frame Relay major interface or subinterface.

### ***frame-relay interface-dlci ietf***

- Use to configure a Frame Relay permanent virtual circuit (PVC) over a subinterface.
- The **ietf** keyword is mandatory and indicates RFC 2427 encapsulation.
- Define a data-link connection identifier (DLCI) in the range 16–1007.
- To configure a Frame Relay PVC, you must specify a DLCI.
- Frame Relay service is offered in the form of PVCs. A PVC is a data-link connection that is predefined on both ends of the connection. A network operator assigns the endpoints of the circuit. Although the actual path taken through the network may vary from time to time, the beginning and end of the circuit do not change. This type of circuit behaves like a dedicated point-to-point circuit.
- PVCs are identified by DLCIs. A DLCI is a 10-bit channel number that is attached to data frames to tell a Frame Relay network how to route the data. Frame Relay is *statistically multiplexed*, which means that only one frame can be transmitted at a time, but many logical connections can coexist on a single physical line. The DLCI allows the data to be logically tied to one of the connections, so that when the data gets to the network, the network knows where to send it.
- DLCIs on the same physical line must match. However, DLCIs have local significance; that is, if the DLCIs are not on the same physical line, the end devices at two different ends of a connection may use a different DLCI to refer to the same connection.
- The system does not support switched virtual circuits (SVCs). An SVC is an any-to-any connection that can be established or removed as needed. With SVCs, you initiate calls using Frame Relay by requesting a destination address and assigning a DLCI, which is established for the duration of the call.

- Example

```
host1(config-subif)#frame-relay interface-dlci 17 ietf
```

- Use the **no** version to remove DLCI/PVC assignment.

### ***frame-relay intf-type***

- Use to configure a Frame Relay interface circuit to operate as data communications equipment (DCE), data terminal equipment (DTE), or Network-to-Network Interface (NNI).
- Frame Relay provides packet-switching data communications between user devices and network equipment across the interface. User devices are referred to as DTE.
- Network equipment that interfaces with a DTE is referred to as a DCE.
- NNI provides a connection between two Frame Relay subnetworks.
- If your system is connected to a Frame Relay switch, configure the interface as a DTE. If your system is connected by a point-to-point line, configure one end as the DTE and the other as the DCE.
- Example

```
host1(config-if)#frame-relay intf-type dte
```

- Use the **no** version to set the default of DTE.

### ***frame-relay keepalive***

- Use to enable the LMI on the interface.
- You can specify the keepalive interval in seconds.
- The value on the DTE should be less than the value set on the DCE.
- The default is 10 seconds.
- Example

```
host1#no frame-relay keepalive
```

- Use the **no** version to disable LMI on the interface.

### ***frame-relay lmi-type***

- Use to configure one of the local management interface types.
- LMI provides configuration and status information relating to the virtual circuits operating over Frame Relay.
- LMI specifies polling mechanisms to receive incremental and full-status updates from the network.
- Systems conform to the following LMI specifications:
  - › **ansi** – ANSI-T1.617 Annex D
  - › **q933a** – ITU-T Q.933 Annex A
  - › **cisco** – original *Group of Four* specification developed by DEC, Northern Telecom, Stratacom, and Cisco
  - › **none** – suppresses LMI
- The default is **cisco**.

- Example
 

```
host1(config-if)#frame-relay lmi-type ansi
```
- Use the **no** version to return to the default LMI type.

***frame-relay lmi-n391dte***  
***frame-relay lmi-n392dce***  
***frame-relay lmi-n392dte***  
***frame-relay lmi-n393dce***  
***frame-relay lmi-n393dte***  
***frame-relay lmi-t391dte***  
***frame-relay lmi-t392dce***

- Use to configure LMI counters and timers.
- LMI counters and timers have configurable ranges that allow you to control the state of the Frame Relay interface. In general, you should accept the default values for the timers and counters, unless you need to modify them according to a special arrangement with your customers.
- Some commands have DTE and DCE versions. Use the **dte** version of the command if the interface is operating as a DTE. Use the **dce** version of the command if the interface is operating as a DCE. Use both versions of the command if the interface is operating as an NNI.
- Use the **frame-relay lmi-n391dte** command to set the N391 full-status polling counter. When you set this counter to a number, *n*, the *n*th request is a full-status request. The range is 1–255 event messages. The default is 6 event messages.
- Use the **frame-relay lmi-n392dte** or **frame-relay lmi-n392dce** command to set the N392 error threshold counter, which specifies the number of errors within N393 events that will place the interface in an operationally down state. The range is 1–10. The default for the DTE version is 3. The default for the DCE version is 2.
- Use the **frame-relay lmi-n393dte** or **frame-relay lmi-n393dce** command to set the N393 monitored events counter to specify the diagnostic window used to verify link integrity. Detection of N392 errors within the window of N393 samples places the interface in an operationally down state. The range is 1–10 events. The default for the DTE version of the command is 4 events. The default for the DCE version is 2 events.
- Use the **frame-relay lmi-t391dte** command to set the T391 link integrity polling timer interval between status inquiries issued by the DTE. The network checks that the DTE polls within the verification interval, T392. The range is 5–30 seconds. The default is 10 seconds.
- Use the **frame-relay lmi-t392dce** command to set the T392 polling verification timer that specifies the maximum interval (in seconds) between the receipt of status inquiries from the DTE equipment before it considers it as an error event. The range is 5–30 seconds. The default is 15 seconds.
- Example
 

```
host1(config-if)#frame-relay lmi-n391dte 20
```
- Use the **no** version to remove the current setting and set the default.

### ***interface pos***

- Use to configure a POS interface in *slot/port* format:
  - › A *slot* refers to a system chassis slot.
  - › A *port* refers to a line module port.
- Example

```
host1(config)#interface pos 0/1
```
- Use the **no** version to remove the POS interface.

### ***interface serial***

- Use to configure a serial interface in the appropriate format by selecting a previously configured physical interface on which you want to configure Frame Relay. For example, for a CT3 interface use *slot/port:channel/subchannel*.
- Use to configure a Frame Relay subinterface in the appropriate format by selecting a previously configured physical interface. For example, for a T3-Frame interface use *slot/port.subinterface*, for a CT1/CE1 interface use *slot/port.channel-group.subinterface*.



**Note:** Before you configure Frame Relay, see the appropriate chapter in this guide for details on configuring physical interfaces.

- › slot – system chassis slot
  - › port – CT3, T3, E3, CT1, or CE1 module I/O port
  - › channel – T1 (DS1) channel
  - › subchannel – set of DS0 timeslots. Refer to the *Fractional T1* section in *Chapter 1, Configuring Channelized T3 Interfaces*.
  - › subinterface – user-assigned nonnegative number that identifies a Frame Relay subinterface
  - › channel-group – CT1 or CE1 channel group number in the range 1–24 (CT1 and J1) and 1–31 (CE1)
- Use the **no** version to remove the subinterface or the serial interface.

### ***ip address***

- Use to assign an IP address and subnet mask to a subinterface.
- Use the **no** version to remove an IP address or to disable IP processing.

### ***sleep***

- Use to disable a Frame-Relay interface.
- Use the **no** version to restart a disabled interface.

## End-to-End Fragmentation and Reassembly

---

The fragmentation and reassembly feature reduces excessive delays of Frame Relay packets by breaking them up into smaller fragments and interleaving them with real-time frames. By doing this, real-time and non-real-time data frames can be carried together on lower-speed links without causing excessive delays to the real-time traffic. On receiving the smaller fragments by the peer interface, the fragments are reassembled into their original packet. For example, short delay-sensitive packets, such as packetized voice, can race ahead of larger delay-insensitive packets, such as common data packets.

The ERX system support end-to-end fragmentation according to the FRF.12 Implementation Agreement standard. Unlike UNI and NNI fragmentation, end-to-end supports fragmentation only at the endpoints. End-to-end fragmentation and reassembly are supported only on non-multilink Frame Relay interfaces on cOC12/STM4 and CT3 12 FO modules.

You configure end-to-end fragmentation at the Frame Relay subinterface level. Fragmentation is applied to all PVCs associated with the subinterface. In the majority of cases, fragmentation and reassembly are used together. Fragmentation and reassembly, however, can be configured separately for each map class.

For additional information, see:

- FRF.12 Version 1.0 – Frame Relay Fragmentation Implementation Agreement, The Frame Relay Forum; Malis, A. (December 1997)

### *Frame Fragmentation*

When you enable fragmentation, you can specify a maximum payload size of the resulting fragments. If the maximum payload size is not specified, the default value of 52 bytes is used. When enabled, fragmentation begins when the portion of the packet that has not been transmitted in previous fragments exceeds the configured maximum payload size. The fragmentation process continues until the entire packet has been transmitted. Frames that do not exceed the configured maximum payload size are not fragmented.

If you disable fragmentation, all packets transmitted by the Frame Relay subinterface are transmitted intact.

## Frame Reassembly

When reassembly is disabled and a data frame is received, a few scenarios may occur:

- If the frame is not a fragment, it is forwarded normally.
- If the frame is a fragment:
  - > If the upper interface is IP (that is, the interface above the Frame Relay subinterface), then the fragment is immediately discarded.
  - > If the upper interface is a connection-based forwarding (CBF) interface and there is an established CBF connection with another peer interface, that fragment is forwarded to the peer interface and no reassembly occurs.

If you enable reassembly, then received fragments undergo the reassembly process. Packets that are not fragments are forwarded as normal.

## Map Class

Within Frame Relay, a map class acts as a container or context for fragmentation and reassembly parameters. Within the map class context, you can explicitly enable fragmentation and reassembly.

Once you define a map class, you can apply it to an unlimited number of subinterfaces. This allows you to change fragmentation and reassembly parameters one time and have the changes immediately reflected in all subinterfaces configured to use that map class.

## Configuring End-to-End Fragmentation

You configure end-to-end fragmentation and reassembly on a subinterface in much the same way you configure a standard Frame Relay interface. In this example, end-to-end fragmentation and reassembly is configured on a single subinterface with a 100-byte fragment size (maximum payload size). All tasks are mandatory unless otherwise noted.



**Note:** The procedure described in this section assumes that a physical interface has been configured. See *Before You Configure Frame Relay*, earlier in this chapter.

- 1 Create a map class that you can apply to subinterfaces.  

```
host1(config)#map-class frame-relay testmap
```
- 2 Specify fragmentation and reassembly for the map class. Optionally, you can specify the maximum payload size of a fragment.  

```
host1(config-map-class)#frame-relay fragment 100
```
- 3 Enter the physical interface on which you want to configure Frame Relay end-to-end fragmentation and reassembly.  

```
host1(config-map-class)#interface serial 5/0:4/1
```
- 4 Select Frame Relay as the encapsulation method for the interface.  

```
host1(config-if)#encapsulation frame-relay ietf
```
- 5 Create a subinterface.  

```
host1(config-if)#interface serial 5/0:4/1.1
```
- 6 Add a circuit to a subinterface.  

```
host1(config-subif)#frame-relay interface-dlci 16 ietf
```
- 7 Assign a local IP address to the circuit.  

```
host1((config-subif)#ip address 42.42.42.41 255.255.255.0
```
- 8 Associate a map class with a subinterface.  

```
host1(config-subif)#frame-relay class testmap
```

### ***encapsulation ietf***

- Use to specify Frame Relay as the encapsulation method for the interface.
- The system uses IETF format (RFC 2427 encapsulation).
- Example  

```
host1(config-if)#encapsulation frame-relay ietf
```
- Use the **no** version to remove Frame Relay configuration from an interface.

### ***frame-relay class***

- Use to associate a map class with a subinterface.
- Example  

```
host1(config-subif)#frame-relay class testmap
```
- Use the **no** version to remove the association between the subinterface and the specified map class from the subinterface.

### ***frame-relay fragment***

- Use to configure fragmentation and reassembly for the map class.
- Specify the keyword **fragmentation-only** to specify only fragmentation; that is, reassembly will not be performed.
- Specify the keyword **reassemble-only** to specify only reassembly; that is, fragmentation will not be performed.
- Specify the maximum payload size of a fragment by using a value from 16–8188 bytes. If a value is not specified, the default value of 52 bytes is used.
- The value for the maximum payload size of a fragment should be less than or equal to the MTU size, otherwise fragmentation never occurs.
- The maximum payload size should be larger than any voice packet so that voice frames are not fragmented.
- Examples

```
host1(config-map-class)#frame-relay fragment 100
host1(config-map-class)#frame-relay fragmentation-only
```
- Use the **no** version to stop fragmentation and reassembly on the subinterface.

### ***frame-relay interface-dlci ietf***

- Use to configure a Frame Relay permanent virtual circuit (PVC) over a subinterface.
- The **ietf** keyword is mandatory and indicates RFC 2427 encapsulation.
- Define a data-link connection identifier (DLCI) in the range 16–1007.
- To configure a Frame Relay PVC, you must specify a DLCI.
- Frame Relay service is offered in the form of PVCs. A PVC is a data-link connection that is predefined on both ends of the connection. A network operator assigns the endpoints of the circuit. Although the actual path taken through the network may vary from time to time, the beginning and end of the circuit do not change. This type of circuit behaves like a dedicated point-to-point circuit.
- PVCs are identified by DLCIs. A DLCI is a 10-bit channel number that is attached to data frames to tell a Frame Relay network how to route the data. Frame Relay is *statistically multiplexed*, which means that only one frame can be transmitted at a time, but many logical connections can coexist on a single physical line. The DLCI allows the data to be logically tied to one of the connections, so that when the data gets to the network, the network knows where to send it.
- DLCIs on the same physical line must match. However, DLCIs have local significance; that is, if the DLCIs are not on the same physical line, the end devices at two different ends of a connection may use a different DLCI to refer to the same connection.
- The system does not support switched virtual circuits (SVCs). An SVC is an any-to-any connection that can be established or removed as needed. With SVCs, you initiate calls using Frame Relay by requesting a destination address and assigning a DLCI, which is established for the duration of the call.

- Example

```
host1(config-subif)#frame-relay interface-dlci 16 ietf
```
- Use the **no** version to remove DLCI/PVC assignment.

### ***interface serial***

- Use to configure a serial interface in the appropriate format by selecting a previously configured physical interface on which you want to configure Frame Relay. For example, for a CT3 interface use *slot/port:channel/subchannel*.
- Use to configure a Frame Relay subinterface in the appropriate format by selecting a previously configured physical interface. For example, for a T3-Frame interface use *slot/port.subinterface*, for a CT1/CE1 interface use *slot/port.channel-group.subinterface*.



**Note:** Before you configure Frame Relay, see the appropriate chapter in this guide for details on configuring physical interfaces.

- › slot – system chassis slot
  - › port – CT3, T3, E3, CT1, or CE1 module I/O port
  - › channel – T1 (DS1) channel
  - › subchannel – set of DS0 timeslots. Refer to the *Fractional T1* section in *Chapter 1, Configuring Channelized T3 Interfaces*.
  - › subinterface – user-assigned nonnegative number that identifies a Frame Relay subinterface
  - › channel-group – CT1 or CE1 channel group number in the range 1–24 (CT1 and J1) and 1–31 (CE1)
- Use the **no** version to remove the subinterface or the serial interface.

### ***ip address***

- Use to assign an IP address and subnet mask to a subinterface.
- Use the **no** version to remove an IP address or to disable IP processing.

### ***map-class frame-relay***

- Use to create a map class.
- Example

```
host1(config)#map-class frame-relay testmap
```
- Use the **no** version to remove a map class.

## Monitoring Frame Relay

---

You can monitor Frame Relay interfaces using the **show frame-relay** commands:

- **show frame-relay interface**
- **show frame-relay lmi**
- **show frame-relay map**
- **show frame-relay pvc**
- **show frame-relay subinterface**
- **show frame-relay summary**

You can set a statistics baseline for Frame Relay interfaces, subinterfaces, and/or circuits using the **baseline frame-relay** command.

You can use the output-filtering feature of the **show** command to include or exclude lines of output based on a text string you specify. Refer to *show Commands* in *ERX System Basics Configuration Guide, Chapter 2, Command Line Interface*, for details.

If you do not specify an interface type for the appropriate **show** command, the output indicates whether a serial or POS interface is being displayed.

### **baseline frame-relay interface**

- Use to set a statistics baseline at the Frame Relay layer for HSSI, multilink Frame Relay, POS, serial or GRE tunnel interfaces, subinterfaces, and/or circuits.
- The system implements the baseline by reading and storing the statistics at the time the baseline is set and then subtracting this baseline whenever baseline-relative statistics are retrieved.
- Specify an interface or subinterface using the interface type and specifier. For more information, see *Interface Types and Specifiers* in *ERX Command Reference Guide, About This Guide*.
- Specify a circuit using the interface type and specifier and the **dlci** keyword and the *dlci* number.
- You cannot set a baseline for groups of interfaces, subinterfaces, or circuits. You must set baselines individually.
- When baselining is requested, the time since the last baseline was set is displayed in *hours:minutes:seconds* or *days/hours* format. If a baseline has not been set, the message “No baseline has been set” is displayed instead.
- The regular interface statistics and LMI statistics for interfaces are subject to the same baseline timestamp. You cannot set separate baselines.
- Use the optional **delta** keyword with Frame Relay **show** commands to specify that baselined statistics are to be shown.
- Example

```

host1#baseline frame-relay interface serial 2/0:1/1
host1#show frame-relay interface delta
Frame relay interface 2/0:1/1, status is lowerLayerDown
Number of interface down transitions is 0
Time since last status change 21:06:34
Number of configured circuits: 0
Time since last baseline 00:00:05
    In bytes: 0                Out bytes: 0
    In frames: 0              Out frames: 0
    In errors: 0              Out errors: 0
    In discards: 0           Out discards: 0
    In unknown protos: 0

```

### **show frame-relay interface**

- Use to display statistics for the Frame Relay layer in a HSSI, multilink Frame Relay, POS, serial, or GRE tunnel interface.
- Specify an interface using the interface type and specifier. For more information, see *Interface Types and Specifiers* in *ERX Command Reference Guide, About This Guide*.
- Use the **brief** keyword to display the operational status of all configured interfaces.
- Use the optional **delta** keyword to specify that baselined statistics are to be shown.
- Field descriptions
  - › status – one of the following states:
    - Up – traffic can flow on the interface
    - Offline – traffic cannot flow because hardware is unavailable
    - Down – traffic cannot flow because of a problem in the interface at the current protocol layer
    - LowerLayerDown – traffic cannot flow because of a problem in an interface at a lower protocol layer
    - AdministrativelyDown – traffic cannot flow because of manual administrative intervention
  - › In bytes – number of inbound bytes received on the interface
  - › Out bytes – number of outbound bytes transmitted on the interface
  - › In frames – number of inbound frames received on the interface
  - › Out frames – number of outbound frames transmitted on the interface
  - › In errors – number of inbound errors received on the interface
  - › Out errors – number of outbound errors transmitted on the interface
  - › In discards – number of inbound packets discarded
  - › Out discards – number of outbound packets discarded
  - › In unknown protos – number of packets received on the interface with unknown protocols
  - › Time since last status change – time since the last status change on the interface

- Example

```
host1#show frame-relay interface
Frame relay interface 3/2:1/1, status is up
Time since last status change 01:21:10
  In bytes: 19712           Out bytes: 60918
  In frames: 1232          Out frames: 1232
  In errors: 0             Out errors: 0
  In discards: 0           Out discards: 0
  In unknown protos: 0
Frame relay interface 3/2:2/1, status is up
Time since last status change 03:06:18
  In bytes: 19728           Out bytes: 60702
  In frames: 1233          Out frames: 1233
  In errors: 0             Out errors: 0
  In discards: 0           Out discards: 0
  In unknown protos: 0
Frame relay interface 3/2:3/1, status is up
Time since last status change 01:20:38
  In bytes: 19696           Out bytes: 60744
  In frames: 1231          Out frames: 1231
  In errors: 0             Out errors: 0
```

### ***show frame-relay lmi***

- Use to display configuration and state information and statistics about the LMI for a HSSI, multilink Frame Relay, POS, serial, or GRE tunnel interface.
- Specify an interface using the interface type and specifier. For more information, see *Interface Types and Specifiers* in *ERX Command Reference Guide, About This Guide*.
- Use the optional **delta** keyword to specify that baselined statistics are to be shown.
- Field descriptions
  - › This command displays both DTE and DCE fields for NNI.
  - › For the DTE:
    - Enquiries sent – total number of LMI status enquiries sent by the DTE on this interface
    - Full enquiries sent – total number of LMI full-status enquiries sent by the DTE on this interface
    - Enquiry responses received – total number of LMI full- and regular-status responses received by the DTE on this interface
    - Full enquiry responses received – total number of LMI full-status responses received by the DTE on this interface
    - Async updates received – total number of LMI asynchronous updates received by the DTE on this interface
    - Unknown messages received – total number of unknown LMI messages received on this interface

- Loss of sequencing detected – total number of times a loss of sequencing in received LMI messages was detected by the DTE on this interface
  - No response timeouts – total number of times a timeout occurred without receiving a response to an LMI request by the DTE on this interface
  - Last sequence number sent – last sequence number sent on this interface
  - Last sequence number received – last sequence number received on this interface
- › For the DCE:
- Enquiries received – total number of LMI status enquiries received by the DCE on this interface
  - Enquiry responses sent – total number of LMI status responses sent by the DCE on this interface
  - Full enquiry responses sent – total number of LMI full-status responses sent by the DCE on this interface
  - Async updates sent – total number of LMI asynchronous updates sent by the DCE on this interface
  - Unknown messages received – total number of unknown LMI messages received on this interface
  - Loss of sequencing detected – total number of times a loss of sequencing in received LMI messages was detected by the DCE on this interface
  - No response timeouts – total number of times a timeout occurred without receiving a status inquiry from the DTE on this interface
  - Last sequence number sent – last sequence number sent on this interface
  - Last sequence number received – last sequence number received on this interface
- Example

```

host1#show frame-relay lmi
LMI information for frame relay NNI interface 3/2:1/1
DTE Parameter N391 is 6, N392 is 3, N393 is 4, T391 is 10
DCE Parameter N392 is 2, N393 is 2, T392 is 15
Configured LMI type is ANSI, status is up
Time since last status change 01:21:14
  Enquiries received: 1232
  Full enquiries received: 207
  Enquiry responses sent: 1232
  Full enquiry responses sent: 207
  Async updates sent: 0
  Unknown messages received: 0
  Loss of sequencing detected: 2
  No response timeouts: 0
  Last sequence number sent: 0
  Last sequence number received: 0
  Unknown messages received: 0
  Loss of sequencing detected: 2

```

```
LMI information for frame relay DCE interface 3/2:2/1
Parameter N392 is 2, N393 is 2, T392 is 15
Configured LMI type is ANSI, status is up
Time since last status change 03:06:22
  Enquiries received: 1233
  Full enquiries received: 207
  Enquiry responses sent: 1233
  Full enquiry responses sent: 207
  Async updates sent: 0
  Last sequence number sent: 0
  Last sequence number received: 0
```

### ***show frame-relay map***

- Use to display the current Frame Relay map entries and information about Frame Relay connections.
- Field descriptions
  - › Frame relay sub-interface – interface number and one of the following states:
    - Up – traffic can flow on the interface
    - Offline – traffic cannot flow because hardware is unavailable
    - Down – traffic cannot flow because of a problem in the interface at the current protocol layer
    - LowerLayerDown – traffic cannot flow because of a problem in an interface at a lower protocol layer
    - AdministrativelyDown – traffic cannot flow because of manual administrative intervention
  - › DLCI – provides decimal value, hexadecimal value, and its value as it appears on the wire
- Example

```
host1#show frame-relay map
Frame relay sub-interface 3/2:1/1.1 (up): DLCI 101(0x65,0x58)
Frame relay sub-interface 3/2:1/1.2 (up): DLCI 102(0x66,0x78)
Frame relay sub-interface 3/2:1/1.3 (up): DLCI 103(0x67,0x78)
Frame relay sub-interface 3/2:1/1.4 (up): DLCI 104(0x68,0x98)
Frame relay sub-interface 3/2:1/1.5 (up): DLCI 105(0x69,0x98)
Frame relay sub-interface 3/2:1/1.6 (up): DLCI 106(0x6a,0xb8)
Frame relay sub-interface 3/2:1/1.7 (up): DLCI 107(0x6b,0xb8)
Frame relay sub-interface 3/2:1/1.8 (up): DLCI 108(0x6c,0xd8)
Frame relay sub-interface 3/2:1/1.9 (up): DLCI 109(0x6d,0xd8)
Frame relay sub-interface 3/2:1/1.10 (up): DLCI 110(0x6e,0xf8)
Frame relay sub-interface 3/2:1/1.11 (up): DLCI 111(0x6f,0xf8)
Frame relay sub-interface 3/2:1/1.12 (up): DLCI 112(0x70,0x1c)
Frame relay sub-interface 3/2:1/1.17 (up): DLCI 117(0x75,0x5c)
Frame relay sub-interface 3/2:1/1.18 (up): DLCI 118(0x76,0x7c)
Frame relay sub-interface 3/2:1/1.19 (up): DLCI 119(0x77,0x7c)
Frame relay sub-interface 3/2:1/1.20 (up): DLCI 120(0x78,0x9c)
Frame relay sub-interface 3/2:1/1.21 (up): DLCI 121(0x79,0x9c)
```

```
Frame relay sub-interface 3/2:1/1.22 (up): DLCI 122(0x7a,0xbc)
Frame relay sub-interface 3/2:1/1.23 (up): DLCI 123(0x7b,0xbc)
Frame relay sub-interface 3/2:1/1.24 (up): DLCI 124(0x7c,0xdc)
```

### **show frame-relay pvc**

- Use to display statistics about PVCs for Frame Relay layer on a HSSI, multilink Frame Relay, POS, serial, or GRE tunnel interface or a specific PVC.
- Specify an interface using the interface type and specifier. For more information, see *Interface Types and Specifiers* in *ERX Command Reference Guide, About This Guide*.
- Specify a virtual circuit using the DLCI number.
- Use the **brief** keyword to display abbreviated PVC information.
- Use the optional **delta** keyword to specify that baselined statistics are to be shown.
- Field descriptions
  - › DLCI – DLCI number
  - › interface – identifies an interface in *slot/port:channel/subchannel* format or subinterface in *slot/port:channel/subchannel.subinterface* format
  - › PVC status – status of the circuit; valid states are *active* and *inactive*.
  - › Number of circuit status inactive transitions – number of times a circuit came down because of error conditions
  - › Time since creation – time since PVC was created
  - › last status change – time since PVC status last changed
  - › In pkts – number of incoming packets received on the circuit
  - › Out pkts – number of outgoing packets transmitted on the circuit
  - › In bytes – number of input bytes received on the circuit
  - › Out bytes – number of output bytes received on the circuit
  - › In FECN pkts – number of packets received with the forward explicit congestion notification (FECN) bit set. The FECN bit is set by a network to notify the user that congestions may be experienced by data traffic in the direction of the frame carrying the FECN bit. The FECN bit is set by the network (not by the transmitting user), and there is no obligation for end systems to take any action regarding the FECN bit.
  - › Out FECN pkts – number of packets transmitted with the FECN bit set
  - › In BECN pkts – number of packets received with the backward explicit congestion notification (BECN) bit set. The BECN bit is set by a network to notify the user that congestions may be experienced by data traffic in the opposite direction of the frame carrying the BECN bit. The BECN bit is set by the network, and there is no obligation for end systems to take any action regarding the BECN bit.
  - › Out BECN pkts – number of packets transmitted with the BECN bit set
  - › In DE pkts – number of packets received with the discard eligibility (DE) bit set. When the DE bit is set, it indicates that the frame should be discarded in preference to other frames without the DE bit set. The DE bit may be set by the network or the user. Once it is set, it cannot be reset by the user.
  - › Out DE pkts – number of packets transmitted with the DE bit set

› Dropped packets – number of dropped packets

- Example

```
host1#show frame-relay pvc
```

```
PVC information for frame relay NNI interface 3/2:1/1
```

```
DLCI 101 in sub-interface 3/2:1/1.1, status is active
```

```
Number of circuit status inactive transitions is 0
```

```
Time since creation 03:27:29, last status change 01:21:29
```

```
In pkts: 0           Out pkts: 0
In bytes: 0          Out bytes: 0
In FECN pkts: 0     Out FECN pkts: 0
In BECN pkts: 0     Out BECN pkts: 0
In DE pkts: 0       Out DE pkts: 0
Dropped pkts: 0
```

```
DLCI 102 in sub-interface 3/2:1/1.2, status is active
```

```
Number of circuit status inactive transitions is 0
```

```
Time since creation 03:27:28, last status change 01:21:29
```

```
In pkts: 0           Out pkts: 0
In bytes: 0          Out bytes: 0
In FECN pkts: 0     Out FECN pkts: 0
In BECN pkts: 0     Out BECN pkts: 0
In DE pkts: 0       Out DE pkts: 0
Dropped pkts: 0
```

```
DLCI 103 in sub-interface 3/2:1/1.3, status is active
```

```
Number of circuit status inactive transitions is 0
```

```
Time since creation 03:27:28, last status change 01:21:29
```

```
In pkts: 0           Out pkts: 0
In bytes: 0          Out bytes: 0
In FECN pkts: 0     Out FECN pkts: 0
In BECN pkts: 0     Out BECN pkts: 0
In DE pkts: 0       Out DE pkts: 0
Dropped pkts: 0
```

### ***show frame-relay subinterface***

- Use to display the state of the Frame Relay subinterface.
- The subinterface can be in one of the following states:
  - › Up – traffic can flow on the interface
  - › Offline – traffic cannot flow because hardware is unavailable
  - › Down – traffic cannot flow because of a problem in the interface at the current protocol layer
  - › LowerLayerDown – traffic cannot flow because of a problem in an interface at a lower protocol layer
  - › AdministrativelyDown – traffic cannot flow because of manual administrative intervention

- Use the optional **delta** keyword to specify that baselined statistics are to be shown.
- The **brief** keyword displays only the operational status of all configured subinterfaces.
- Field descriptions
  - › sub-interface – identifies the subinterface in *slot/port/channel/subchannel.subinterface* format
  - › status – status of the subinterface
  - › Time since last status change – time since the last status change on the subinterface
  - › In bytes – number of inbound bytes received on the subinterface
  - › Out bytes – number of outbound bytes transmitted on the subinterface
  - › In frames – number of inbound frames received on the interface
  - › Out frames – number of outbound frames transmitted on the interface
  - › In errors – number of inbound errors received on the subinterface
  - › Out errors – number of outbound errors transmitted on the subinterface
  - › In discards – number of inbound packets discarded
  - › Out discards – number of outbound packets discarded
  - › In unknown protos – number of packets received on the subinterface with unknown protocols
- Example

```

host1#show frame-relay subinterface
Frame relay sub-interface 3/2:1/1.1, status is up
Time since last status change 01:21:26
  In bytes: 0                Out bytes: 0
  In frames: 0              Out frames: 0
  In errors: 0              Out errors: 0
  In discards: 0           Out discards: 0
  In unknown protos: 0
Frame relay sub-interface 3/2:1/1.2, status is up
Time since last status change 01:21:26
  In bytes: 0                Out bytes: 0
  In frames: 0              Out frames: 0
  In errors: 0              Out errors: 0
  In discards: 0           Out discards: 0
  In unknown protos: 0

Frame relay sub-interface 3/2:1/1.3, status is up
Time since last status change 01:21:26
  In bytes: 0                Out bytes: 0
  In frames: 0              Out frames: 0
  In errors: 0              Out errors: 0
  In discards: 0           Out discards: 0
  In unknown protos: 0

```

***show frame-relay summary***

- Use to scan all defined Frame Relay interfaces and circuits; reports aggregate status as one of the following:
  - › Up – traffic can flow on the interface
  - › Down – traffic cannot flow because of a problem in the network
  - › Unavailable – traffic cannot flow because hardware is unavailable
- Example

```
host1#show frame-relay summary  
28 interface(s) defined, 28 up, 0 down  
840 sub-interface(s) defined, 840 up, 0 down  
840 circuit(s) defined, 840 up, 0 down
```

