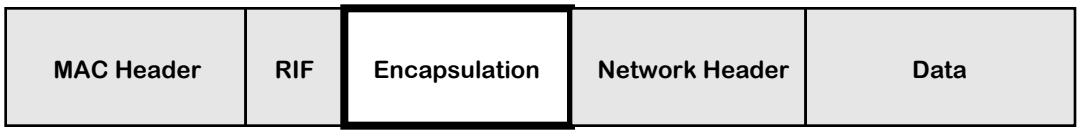
15 Configuring Frame Translations

Encapsulation is the biggest problem for implementing a transformation algorithm in support of any to any switching. All of the media provide a choice of more than one encapsulation and not all encapsulations are available on all media. Additionally, the methodology of these encapsulations vary from protocol to protocol.



An ideal protocol would dictate a single encapsulation which would be the same on all media.

Most protocols make use of more than one encapsulation. For example, IP uses Ethertype most of the time on Ethernet and SNAP (an instance of an 802.2 LLC) on FDDI and Token Ring. In this case, there may be clearly established rules for transforming from one encapsulation to another as media are traversed.

Some protocols may allow more than one encapsulation even on a single media type. Some might use the encapsulation to separate functional parts of the protocols, for example, routing table updating protocols from user data forwarding protocols. Others, like IPX may simply allow the user to arbitrarily choose them.

Some, most notably IPX, may entangle the notion of encapsulation with the notion of the network level broadcast domain to create multiple logical networks over a single physical broadcast domain.

Clearly, then there is no single algorithmic rule by which the any to any transformation function can switch arbitrary protocols. There are two choices available to address this situation.

1. The switch must be configurable, per device, per protocol, per media to select the transformation of encapsulations.
2. The switch performs a single transformation and the user must configure all end-stations and routers to use this single choice made by the switch.

The OmniAccess 408 uses the first approach for IP and IPX as the dominant protocols in the market. It uses the second approach for all other protocols.

Protocols other than IP and IPX

For protocols other than IP or IPX three encapsulations are possible on Ethernet media:

- Ethertype
- IEEE 802.2 LLC
- IEEE 802.2 SNAP (This is an instance of an LLC encapsulation defined by the 802.2 committee to support the transformation of Ethertype Ethernet frames to media which don't support that encapsulation.)

The SNAP Conversion

The intent of the 802.2 committee is that Ethertype frames are transformed to SNAP on crossing from Ethernet media to 802 media and restored to Ethertype in the reverse direction.

The OmniAccess 408 could follow this rule for all protocols including IP; however, this would prevent AppleTalk interworking between Ethernet and FDDI. The OmniAccess 408 explicitly checks for the AppleTalk protocol. If found, the rule is not applied. In addition, the OmniAccess 408 checks for the Banyan Vines protocol and translates according to the media type (see *Banyan Vines* on page 15-8).

As there may be other protocols with this problem, the SNAP-to-Ethertype transformation is configurable for all protocols other than AppleTalk.

Other Conversions

There are no equivalent algorithmic approaches which the transformation function can adopt for dealing with protocols which require Ethertype on Ethernet and some form of LLC encapsulation on FDDI and/or Token Ring. The mapping between Ethertype values and LLC values is arbitrary requiring tables indexed by protocol.

The approach followed in the OmniAccess 408 is therefore to simply pass LLC encodings between Ethernet, FDDI and Token Ring with no changes other than to insert/strip the length field required by IEEE 802.3 on Ethernet.

This leaves protocols which require transformations between Ethertype and LLC encapsulations as unswitchable unless the clients and servers can be configured to use SNAP.

Summary of Non-IPX Encapsulation Transformation Rules

To summarize:

- Ethertype/SNAP transformations are configurable for all protocols except AppleTalk and Banyan Vines. Ethertype frames going to FDDI or Token Ring are translated to SNAP unconditionally. SNAP frames going to Ethernet are translated to Ethertype or left as SNAP as per configuration, unless the protocol is AppleTalk in which case they are left as SNAP.
- LLC frames are passed unchanged in value but with the length field required on Ethernet media stripped/inserted.

IPX Encapsulation Transformation Rules

For IPX the encapsulation problems described above are compounded by the introduction of a fourth encapsulation on Ethernet media. Novell introduced a frame format when the IEEE 802.3 standards committee produced its version of Ethernet which was incompatible with Ethernet.

Novell places its network header and data within a raw IEEE 802.3 Ethernet frame with no intervening IEEE 802.2 LLC header. This is in direct contravention of the standards but has become a de facto standard encapsulation.

Novell refers to this encapsulation types as ETHERNET_802.3. It is also widely known as Novell Proprietary, Novell Raw, Raw 802.3, etc. Such frames are identifiable only by the fact that the Novell Network header starts with a two byte field called the *checksum*, which is never used and assumes the value 0xFFFF.

Routers, bridges and switches therefore check for the checksum after an 802.3 length field. In effect, Novell has usurped the value OxFF for the Destination and Source SAP addresses (DSAP/SSAP) of an LLC header.

Thus on Ethernet media there are four encapsulations for IPX

- Ethertype - value 0x8137
- Novell Proprietary
- LLC - SAP value 0xE0
- SNAP - Protocol Identifier 0x0000008137

On Token Ring and FDDI the same LLC and SNAP encapsulations are found as on Ethernet (without the length field.)

This leaves an aggregate of four encapsulations across all media with only two being universal (LLC and SNAP.)

Unfortunately, the SNAP conversion rule isn't applicable and there is no algorithmic determination for the use of particular encapsulations on any media - it's purely the choice of the network administrator. Worse, multiple encapsulations can be found on a single media to create multiple logical networks over a single physical broadcast domain.

The OmniAccess 408 therefore allows configuration of the encapsulation transformations of IPX frames. Before transmission of a frame occurs the switch determines first the current encapsulation of the frame. Then, it consults configuration information to determine which of the permitted encapsulations for the media the frame is to be transmitted on is required. Thus, the administrator can choose not only a single output option but an option per possible received encapsulation.

For example, over FDDI media LLC and SNAP are permissible so the administrator might configure one of the following:

- LLC and SNAP encapsulations received from other FDDI, Token Ring or Ethernet media are translated to SNAP.
- Ethertype and Proprietary encapsulations from Ethernet are translated to LLC.

Essentially, for each encapsulation, transformation to each of the other three encapsulations is available, but may simply be left as is. This choice may be further constrained by the output media type, for example, Ethertype is not a valid option on FDDI or Token Ring.

The Network Header

There are essentially two requirements for the any to any switching transformation function to address the network header fields:

- Network Address to MAC Address Mapping

  In every protocol there is a mechanism for mapping global network wide addresses to the MAC addresses required in the local broadcast domain.

- Frame Size Requirements of the Media

  Different media have different minimum and maximum frame sizes leading to the issues of padding insertion/stripping and fragmentation/reassembly or maximum frame size negotiation protocols at the network level.

Address Mapping

There are almost as many ways to map a global network level address to a local subnetwork MAC address as there are routing protocols. These may or may not be affected by any to any switching.

Some may construct MAC addresses algorithmically, for example, DECNET model. Some may involve table lookups with an additional protocol to build and maintain these tables, for example, the IP/ARP model. Others may involve some form of building the network address around the MAC address as in the IPX model.

In all cases these mechanisms are susceptible, without good design and forethought, to the problem of canonical versus non-canonical representation of addresses in the network header area.

Address Mapping in IP: ARP

To map a 32-bit IP network address into the MAC address of a locally connected station a router uses the Address Resolution Protocol (ARP) to build an ARP Table. The router broadcasts a request containing the IP address in the body of the frame. The station with that IP address responds with its MAC address *in the body* of an ARP reply frame. The router inserts these two addresses in its ARP table and can then use the MAC address received to transmit any frames addressed to that IP address.

Since a router can have interfaces to Ethernet ports (canonical MAC addresses) and FDDI and Token Ring (non-canonical MAC addresses) it is crucial that the router keeps track of what media type it receives on each port.

If IP ARP were defined such that all MAC addresses, *when conveyed in the body of an ARP*, were in canonical format, switching would be easy. A router, when taking an address from the ARP table and using it as the destination MAC address on an Ethernet port would use the address as is. If sending to FDDI or Token Ring it would bit swap the address to non-canonical format as required by the media.

Given this model of implementation a station responding with an ARP on Ethernet which was switched to FDDI would result in the same representation of the MAC address in the ARP table of the router. The router would then use the bit swapped form in the MAC address of subsequent frames to the FDDI ring and the switch would bit swap these MAC header address as it transformed the frame onto Ethernet, resulting in the correct representation to be received by the original station.

Unfortunately, this model has only been defined in IP for Ethernet and FDDI. Token Ring stations place MAC addresses into the body of ARP frames in their native, non-canonical format and routers use addresses from the ARP table as is when sending to Token Ring ports.

To achieve any to any switching with IP it is therefore necessary for the OmniAccess 408 to be sensitive to ARP frames and to bit swap the MAC addresses *in the body of the ARP* when switching a frame between Token Ring and FDDI or Ethernet.

Because IP is well designed, the issue of address mapping being confined to the ARP protocol, this is sufficient to isolate the problem allowing all subsequent IP frames to be switched any to any.

Address Mapping in IPX

A network address in IPX consists of three parts:

1. Network Number -- a globally unique identifier of a particular broadcast domain.

   Strictly, because of the formation of logical networks using encapsulations, this is not equivalent to a physical broadcast domain but the distinction can be put aside for the purposes of this particular discussion.

2. Node Address -- the MAC address of a station on that domain.

3. Socket Number -- the task (process) within that station which should process the message.

Just as in IP, routers move a frame along hop by hop on the basis of the network number portion of the destination address. To do this, IP needs the MAC address of the next hop router. This address is obtained from the RIP table that is built up from the RIP updates sent out by all routers. When a router receives a RIP update frame it uses the source node address in the frame as the MAC address for the next hop router.

Although there is not an explicit ARP like protocol for mapping addresses in IPX, this same function is achieved by the use of source node addresses in RIP frames.

In IPX, as in IP, the canonical versus non-canonical representation of addresses in ARPs still applies. In switching, this needs to be considered for the source node address in IPX frames.

In IPX Ethernet and FDDI observe a convention of using MAC addresses in the IPX header in canonical format.

Frame Size Requirements

The frame size requirement for the different media cause two problem areas which have to be addressed by the any to any switching transformation function.

- Ethernet has a minimum frame size requirement. This requires that padding is inserted on frames switched to it which are below the minimum size and stripped from frames switched from it.
- All media have different maximum frame size requirements. This gives rise to the problems of fragmenting large frames and/or negotiating maximum frame sizes.

Insertion of Frame Padding

Ethernet has a minimum frame size of 64 bytes. For frames smaller than 64 bytes it is a simple task for the OmniAccess 408 to perform padding.

Stripping of Padding for all IEEE 802.3 Frames.

Ethernet frames in IEEE 802.3 format can be stripped of padding because of the presence of the length field. This includes all LLC and hence SNAP encapsulated protocols as well as Novell Proprietary format.

No stripping of non-IPX Ethertype Frames

Padding can only be detected for Ethertype encapsulated frames if the protocol is known and the protocol has some length information which can allow the valid data size to be inferred. This is protocol specific and is currently only performed for IPX frames. Thus, the OmniAccess 408 *does not* strip padding from non-IPX Ethertype encapsulated frames *including IP*.

IPX Specific Stripping

For IPX the OmniAccess 408 performs pad stripping for all frame types including Ethertype. This is possible because all IPX frames have a common header that includes the data length, allowingand the frame size to be inferred.

In fact, for IPX, the length in the IPX header is used to strip padding in all frame encapsulations including the 802.3 based formats. This is because many IPX Ethernet implementations also pad frames to an even byte length. This single byte pad when performed on 802.3 based frames is included in the 802.3 length field. Thus the generic 802.3 based stripping technique is not sufficient to strip this odd-byte padding. When performing any to any switching FDDI implementations of IPX were found to be tolerant of this extra byte whereas Token Ring implementations would not work with it present. By adopting the single IPX stripping strategy of using the IPX header length these problems are avoided thus the OmniAccess 408 unconditionally strips all padding from IPX frames.

Also, it *does not* support odd-byte pad insertion when switching to Ethernet. This was a feature added to overcome limitations of some NIC cards which is now of only historical importance and in fact, Netware 4.1 servers provide this insertion as a port configuration option.

MTU Handling

Routers address the problem of maximum frame size limitations with the notion found in many protocols of a Maximum Transmission Unit (MTU) size. Protocols use this notion in two possible ways.

- PDU Fragmentation/Reassembly

The router is configured with the MTU of each port. If a frame that is too large is required to be sent on a port, the Protocol Data Unit (PDU) within the frame is fragmented into many smaller PDUs, each of which is re-encapsulated and sent as a frame that fits within the MTU.

- Connection-oriented end-to-end MTU negotiation

  When an end-station enters into a protocol to communicate with another station the initial PDU exchanges are guaranteed to fit all possible MTUs. In the handshaking between end-stations to establish the connection a phase is entered where large frames are sent. If an intervening link has an MTU too small for these frames it will be dropped and the handshaking will time out. The end-stations send progressively smaller frames until the handshaking succeeds and hence establish the MTU to be used between the two stations for the remainder of their connection use.

IP supports the former mechanism and IPX the latter.

IP Fragmentation

The OmniAccess 408 Ethernet interfaces will use IP fragmentation if they are allowed too (i.e., if the *Don't Fragment* bit is not set.) Fragmentation by FDDI and Token Ring is not supported though technically the Token Ring could send frames larger than those supported by FDDI and LAN Emulation could generate frames larger than both.

ICMP Based MTU Discovery

IP uses the Don't Fragment bit to support an MTU discovery protocol that superficially resembles the negotiation of IPX. The difference is that when IP stations attempt to discover an MTU size for their use, which doesn't require fragmentation by intermediate routers, the protocol expects a protocol response *by the intermediate router*, this is an ICMP reporting that a frame was dropped because it couldn't be fragmented.

The OmniAccess 408 transformation function of any to any switching *does not* support this ICMP generation but just silently drops IP frames which can't be fragmented. The IP *router* in the OmniAccess 408 does honor this protocol and support ICMP. It is only the any to any switching which doesn't because it is not a router and may not even have an IP address with which to respond.

IPX Packet Size Negotiation

For IPX the requirement of intervening devices is simply to drop frames that are too large to be forwarded. This is what the OmniAccess 408 does.

Other Protocols

Dropping oversize frames is the approach for all protocols other than IP. If the protocol in question is modeled like IPX this will be the correct thing to do and will not cause problems. If the protocol is modeled like IP and expects fragmentation to occur or requires explicit response from the OmniAccess 408 then the protocol will not succeed in any to any switching.

Banyan Vines

Banyan Vines supports Ethernet. The Banyan Vines protocol only uses one frame format per network type—no user configuration of translations is necessary. This protocol uses Ethernet II frames on Ethernet, SNAP frames on FDDI, and IEEE 802.2 (LLC) frames on Token Ring. The OmniAccess 408 uses these frame formats when translating Banyan Vines frames.

Note

Checksums for Banyan Vines frames are automatically set to the null checksum, 0xFFFF, so that the checksum header does not require recalculation. Receiving stations will ignore this field and assume the sender is not using checksums.

Configuring Encapsulation Options

You will configure frame encapsulation based on the destination MAC address or the destination switch port. Whether a a frame is encapsulation based on the destination MAC or the port depends whether the frame has a unicast, multicast, or broadcast destination.

Forwarding versus Flooding

Such frames will be handled in two ways:

- **Forwarded Frames**. If the frame has a unicast destination address which has been learned on a particular port, the encapsulation translation choices are driven by options associated with the destination MAC address.
- **Flooded/Multicast Frames**. If the frame has a unicast destination address which has not been learned on a particular port, or if the destination address is a multicast address, then the frame has to be transmitted on potentially many ports. In this case the encapsulation translation choices are driven by options associated with each destination port.

Port Based Translation Options

The translation options for ports allow configuration of IP and IPX protocols on a per encapsulation basis.

MAC Address Based Translation Options

The translation options for MACs arises from two possible sources.

- Inheritance from Port Options During Source Address Learning
- When a source MAC address is learned, the translation options of the port on which it is learned are copied into the MAC-based database.
- Automatic Determination by AutoTracker
- When a frame is processed by AutoTracker as part of determining the VLAN to be associated with the MAC the frames protocol type and encapsulation are also determined. This information is used to update/set the translation options in the MAC based database.

Which of these options is used is determined by setting the autoencaps option.

"Native" versus "Non-Native" on Ethernet

For the Ethernet one further distinction is made. If the frame received from the backplane is an Ethernet media type frame from another Ethernet switching module in the same chassis, then *no encapsulation translations are applied*. Such frames are referred to as Native frames.

If the frame is of an Ethernet media type but was put onto the backplane by some other type of switching module, for example, the frame came from a FDDI card via a trunk port, or from the MPM via routing, then *encapsulation translations are applied*. Such frames are referred to as Non-Native frames.

No Translation on Trunk or PTOP ports

Switching modules which support encapsulation mechanisms, such as Trunking ports on FDDI and Token Ring, and Point to Point ports on ATM do not apply translation to frames destined to such ports.

All other aspects of the transformation process are driven by the media type of the frame, the media type of the port on which the frame is to be transmitted and the protocol type determined for the frame. Thus frame padding insertion/stripping, IP fragmentation, IP ARP bit swapping, etc., are all automatic.

The User Interface

Simple encapsulation options can be configured through the *modvp, addvp, crgp* commands. More advanced encapsulation options can be found in the commands under the Switch menu

Essentially, the forwarding code is now capable of applying the transformation function per protocol per encapsulation per port for flooded/mcast traffic and per protocol per encapsulation per destination MAC address for forwarded unicast traffic. The old interface provides a small subset of these possible port translation options.

The addvp, modvp and crgp Commands

All of these commands include in their dialogue an Output Format question for ports and a subsidiary IEEE 802.2 Pass through option.

The options offered are:

- a default,
- Ethertype,
- SNAP and
- LLC.

Each of these represents a set of translation options for the IP and IPX protocols. The names chosen for these sets basically represent the translations for IPX with the translation for IP being implied.

For example, LLC represents a translation set where all IPX encapsulations are configured to translate to IEEE 802.2. This is not a valid encapsulation for IP which is therefore configured to a default appropriate to the media, Ethertype for Ethernet ports and SNAP for FDDI and Token Ring ports. The translation of all other protocol types and encapsulations is fixed by the OmniAccess 408. Thus AppleTalk is never translated and Ethertype/SNAP based protocols follow the IP option.

For those options which imply a translation of IEEE 802.2 IPX frames to something else a subsidiary question is asked, “IEEE 802.2 IPX Pass Through(y/n):” An IEEE 802.2 pass through option is provided because 4.1 Novell servers use this encapsulation by default and it is becoming Novell’s encapsulation of choice.

The Default Translation Option

The meaning of the default is determined separately for each media type and is fully configurable. The factory defaults are chosen so that the latest release is fully compliant with earlier ones. The default translation option is provided to allow a “single point of configuration of all ports” capability. When the default option for a media is changed all ports of that media type whose encapsulation is configured as default will inherit the new translation setting. All MAC address-based translation options which were inherited from those ports, as opposed to those set by AutoTracker, will also be updated. Ports which have an encapsulation setting other than default will be unaffected.

Ethernet Factory Default Translations

For Ethernet switching module ports the factory default is set to the following:

Ethernet Media - Default Mode
No translation is performed on outbound Ethernet frames where the inbound interface was Ethernet.
IP frames of any encapsulation are transmitted as Ethernet II frames.
IPX frames are transmitted as IEEE 802.3 Proprietary as the default setting. The only exception is when LLC passthrough mode is enabled, then the IEEE 802.2 (LLC) frames are forwarded as is.
No translation is performed on Appletalk frames, and we currently support only Appletalk Phase II (SNAP format).
Banyan Vines frames are transmitted as Ethernet II frames.
Other than IP and IPX, all other Ethernet II and SNAP encapsulated protocols are sent as Ethernet II frames.
All other IEEE 802.3 with LLC encapsulated protocols are not translated.

ATM LANE Factory Default Translations

For ATM LAN Emulation service ports the factory default is set to the following:

ATM LANE - Default Mode
No translations performed on Ethernet frames.
FDDI and Token Ring frames are translated to either SNAP or LLC and are transmitted as such on ATM LANE.
Banyan Vines Token Ring and FDDI frames are translated to Ethertype.

The Ethertype Option

This option can only be applied to Ethernet switching module ports. It is set to the following:

Ethernet Media - Ethernet II Mode
No translation is performed on outbound Ethernet frames where the inbound interface was Ethernet.
IP frames are transmitted as Ethernet II frames.
All IPX frames are transmitted as Ethernet II frames. The only exception is when LLC passthrough mode is enabled, then the IEEE 802.2 (LLC) frames are forwarded as is.
No translation is performed on Appletalk frames, and we currently support only Appletalk Phase II (SNAP format).
Other than IP and IPX, all other Ethernet II or SNAP frames are transmitted as Ethernet II frames.
Other IEEE 802.3 with LLC are not translated.

ATM LANE - Ethernet II Mode
IPX frames from FDDI, Token Ring, and Ethernet SNAP frames are translated to Ethertype.
All other SNAP frames from FDDI, Token Ring, and Ethernet SNAP are translated to Ethertype. However, Appletalk ARP SNAP frames from Token Ring and FDDI are left as SNAP; Banyan Vines frames from FDDI are translated to Ethertype.
All other 802.2 frames from FDDI, Token Ring, and Ethernet are left as is. The exception are Banyan Vine frames from Token Ring, which are translated to Ethertype.
All Ethernet Ethertype frames are not translated.

The SNAP Option

This option can be applied to all media type ports and is set to the following:

Ethernet Media - SNAP Mode
No translation is performed on outbound Ethernet frames where the inbound interface was Ethernet.
IP frames are transmitted as SNAP frames.
All IPX frames are transmitted as SNAP frames.
No translation is performed on Appletalk frames, and we currently support only Appletalk Phase II (SNAP format).
Other than IP and IPX, all other Ethernet II or SNAP frames are transmitted as SNAP frames.
Other IEEE 802.2 with LLC are not translated.

In the **modvp** or **addvp** commands for FDDI and Token Ring the only choices other than default are SNAP or LLC and the default must be one of these. As the factory default is SNAP with IPX 802.2 Pass through and the SNAP does not imply pass through the additional question about pass through is not asked on FDDI and Token Ring ports as the preference can be expressed by choosing default or SNAP explicitly.

ATM LANE - SNAP Mode
All IPX frames are translated to SNAP unless they are already SNAP, in which case they are forwarded as is.
All Ethertype or SNAP frames from Ethernet and SNAP frames from Token Ring or FDDI are translated to SNAP or left as SNAP. The exception is Banyan Vines frames from FDDI, which are translated to Ethertype.
All other LLC frames are left as is. The exception is Banyan Vines from Token Ring, which is translated to Ethertype.

The LLC Option

This option can be applied to all media type ports and is set to the following:

Ethernet Media - LLC Mode
No translation is performed on outbound Ethernet frames where the inbound interface was Ethernet.
IP frames are transmitted as Ethernet II frames.
All IPX frames are transmitted as IEEE 802.2 (LLC) frames.
No translation is performed on Appletalk frames, and we currently support only Appletalk Phase II (SNAP format).
Other than IP and IPX, all other Ethernet II or SNAP frames are transmitted as Ethernet II frames.
Other IEEE 802.2 with LLC are not translated.

In the **modvp** or **addvp** commands for FDDI and Token Ring the only choices other than default are SNAP or LLC and the default must be one of these. As the factory default is SNAP with IPX 802.2 Pass through and SNAP does not imply IPX 802.2 Pass through, the additional question about pass through is not asked on FDDI and Token Ring ports. By choosing SNAP, it is implied that there is no IPX 802.2 Pass through.

ATM LANE - LLC Mode
IPX frames are translated to 802.2 LLC.
All other SNAP frames from FDDI, Token Ring, and Ethernet SNAP are translated to Ethertype. However, Appletalk ARP SNAP frames from Token Ring and FDDI are left as SNAP; Banyan Vines frames from FDDI are translated to Ethertype.
All other LLC frames are not translated. The exception is Banyan Vines frames from Token Ring, which are translated to Ethertype

Interaction with the new interface

If the port to which these commands are being applied has been configured with the new interface commands its encapsulation will be displayed as **SWCH** in the **vi** command output. The user is alerted to this fact in these commands by the default response to the output format question in the **modvp** command being displayed as "*" instead of **d**,**e**,**s** or **l**. A simple return will leave the options unchanged in this case. If the port is currently one of **d**,**e**,**s** or **l** and the user types "*" in response the encapsulation is changed to **SWCH** and the options are set to a null translation set.

The "vi" Command

The encaps column displays the encapsulation subset options set for each port. If the port has been configured with the new interface this is indicated by displaying "SWCH." The "canned" subsets offered in this interface are displayed as follows:

- **DFLT**. This indicates that the port is using the default translation options applicable to the media type of this port. See above.
- **802.2**. This indicates that IPX frames of any encapsulation will be encapsulated with IEEE 802.2. Non-IPX frames other than AppleTalk will be transformed to Ethertype on Ethernet ports and SNAP on FDDI or Token Ring ports. AppleTalk frames are never transformed.
- **SNAP**. This indicates that Ethertype frames of all protocols and IPX proprietary frames will be translated to SNAP and all SNAP frames will be left as is.

  IEEE 802.2 encapsulated IPX frames may be left as is if the IEEE 802.2 pass through option is in effect for this port. All other IEEE 802.2 encapsulated protocols are left as is.
- **ETH**. This indicates that SNAP frames of all protocols except AppleTalk will be translated to Ethertype.

  SNAP and Proprietary IPX frames will be transformed to Ethertype.

  IEEE 802.2 encapsulated IPX frames may be left as is if the IEEE 802.2 pass through option is in effect for this port.

  All other IEEE 802.2 encapsulated protocols are left as is.

To discover whether IEEE 802.2 pass through is in effect on a port the user must either use the swch command from the switch menu or use modvp and observe the encapsulation offered and/or the default response for the pass through question.

The Switch Menu

The switch menu contains commands that allow you to set translation options discussed earlier in this chapter. It also contains commands to change the default values.

To view the switch menu, enter **switch** at the prompt. If you are in verbose mode, the following screen is displayed. Otherwise, type a **?** at the switch menu prompt to display the Switch Menu:

```
Command              Switch Menu
-------------------- ------------------------------------------------------------------------------------------
propipx              Configure Default Proprietary IPX Token Ring to any switching
facdef               Configure Defaults to Factory values
ethdef               Configure Default Ethernet Translation
fddidef              Configure Default FDDI Translation
trdef                Configure Default TR Translation
swch                 Configure Any To Any Switching Port Translations
swchmac              View per MAC Translation Options
autoencaps           Turn AutoTracker translations On or OFF
```

The commands above and their operations are described in the sections below.

Factory Defaults

You can reset all ports in the switch to their default factory settings. Any custom translations you configured through **modvp**, **ethdef**, **fddidef**, **trdef**, or **swch** commands will be overridden by the default translation for the given media type (i.e., Ethernet, FDDI, etc.). Factory defaults for each media type are described earlier in this chapter.

To reset to factory defaults, enter the **facdef** command at the system prompt. The following screen displays:

```
This will reset the default translations for each media type to a factory default.
It will then set all port translation options to inherit these defaults.
It will then reset the forwarding table translation options for all addresses learnt on
those ports to those port defaults.
Do you want to do this? (no):
```

Enter a **Y** to reset all port settings.

Default Ethernet Translations

The **ethdef** allows you to set up default translations for all Ethernet ports. To do so:

1. Enter **ethdef** at the system prompt. The following screen displays:

```
This will reset the default translations for Ethernet media to a new value.
All Ethernet ports currently set to default will inherit these new translation options.
It will thenreset the forwarding table translation options for all addresses learnt on
those ports to those port defaults.
Do you want to do this? (no):
```

2. Press **Y** at the **Do you wish to do this?** prompt to indicate that you want to change the defaults. The current settings for Ethernet ports are displayed, in a screen similar to the following:

```
Translation Options:
1       IP Ethertype                                  -> Ethertype
2       IP IEEE 802 SNAP                              -> Ethertype

3       IPX ETHERNET_II                               -> 802.3
4       IPX ETHERNET_802.3                            -> 802.3
5       IPX ETHERNET_802.3/FDDI/TOKEN_RING            -> 802.3
6       IPX ETHERNET_SNAP/FDDI_SNAP/TOKEN-RING_SNAP   -> 802.3
```

   There are six frame types for which you can set translation options. The frame type in the left column indicates the incoming frame, and the frame type in the right column (after the **->**) indicates the outgoing frame. You can configure the outgoing frame type for each incoming frame.

3. You change an outgoing frame type by entering its line number, an equal sign (=) and a frame type indicator (**e**, **s**, **2**, or **3**). The frame type indicators represent the following frames:

e	Ethernet II or Ethertype
s	SNAP
2	802.2 or LLC
3	Ethernet 802.3

   For example, if you wanted to change incoming IPX Ethernet II frames to Ethernet 802.3 frames, then you would enter

   ```
   3=3
   ```

   Please note that the IP Translation Options accept only Ethertype (**e**) or SNAP (**s**).

4. When are done changing translations, enter **save** to save all of your settings. If you enter **quit**, you will exit the **ethdef** command without saving your changes.

Port Translations

The **swch** command allows you configure translations on a port-by-port basis. Its translation options are similar to those for **ethdef**, **fddidef**, and **trdef**. However, instead of applying translations to all ports for a particular media type, **swch** applies translations only to the port you specify.

To specify translation for a single port:

1. Start the **swch** command by entering it at the prompt as shown:

   swch <slot>/<port>

   where **<slot>** is the board on which the port is located and **<port>** is the port number. For example, to set the translation for port 1 on slot 2, enter the following:

   swch 2/1

2. Something like the following screen displays, showing the current translation settings for the port:

```
                Port Translations for Ethernet port 2/1/brg/1

0               Framing Type: DFLT

Translation Options:
1               IP Ethertype                                -> Ethertype
2               IP IEEE 802 SNAP                            -> Ethertype

3               IPX ETHERNET_II                             -> 802.3
4               IPX ETHERNET_802.3                          -> 802.3
5               IPX ETHERNET_802.3/FDDI/TOKEN_RING          -> 802.3
6               IPX ETHERNET_SNAP/FDDI_SNAP/TOKEN-RING_SNAP -> 802.3
```

   The top line of the display indicates the media type of the port as well as the slot number, port number, service type, and service number. The next line, **Framing Type**, indicates the framing type applied to this port through the **modvp** command. If the framing type had been defined through the Switch menu, then this field would read **SWCH**.

3. The Translation Options section shows the six frame types for which you can set translation options. The frame type in the left column indicates the incoming frame, and the frame type in the right column (after the **->**) indicates the outgoing frame. You can configure the outgoing frame type for each incoming frame.

   Note that the default option is a question mark (**?**). If you press **<Return>**, the help information will be redisplayed

4. You change an outgoing frame type by entering its line number, an equal sign (=) and a frame type indicator (**e**, **s**, **2**, or **3**). The frame type indicators represent the following frames:

e	Ethernet II or Ethertype
s	SNAP
2	802.2 or LLC
3	Ethernet 802.3

   For example, if you wanted to translate incoming IPX SNAP frames to LLC frames, then you would enter

   6=2

5. When are done changing translations, enter **save** to save all your settings. If you enter **quit**, you will exit the **swch** command without saving your changes.

Configuring Additional Ports

If you want to configure additional ports, you can use the **n** option of the **swch** command to configure the next port, or the **p** option of the **swch** command to configure the previous port. For example, if want to configure translations on port 2 for the card in slot 4 after configuring Port 1 in Slot 4, enter

n

at the prompt. You are now ready to configure port 3 of slot 4.

If you want to configure translations on port 1 for the card in slot 5 after configuring Port 2 in Slot 5, enter

p

at the prompt. You are now ready to configure port 1 of slot 5.

When are done changing translations, enter **save** to save all your settings. If you enter **quit**, you will exit the **swch** command without saving your changes.

Displaying Ethernet Switch Statistics

The **swch** command can also be used to display basic statistics for Ethernet ports. These statistics are the lowest level, most primitive statistics maintained by an Ethernet board. The more familiar RMON and MIB II statistics are generated from these statistics. If you want to display the switch statistics for an Ethernet port, enter

swch <slot>/<port>

where **<slot>** is the slot number of the module, and **<port>** is the number of the port for which you want to view statistics. For example, to look at statistics for port 4 in slot 3, enter:

swch 3/4

A screen similar to the following is displayed:

```
                Port Translations for Ethernet port 3/4/brg/1

0               Framing Type: DFLT

Translation Options:
1               IP Ethertype                                    -> Ethertype
2               IP IEEE 802 SNAP                                -> Ethertype

3               IPX ETHERNET_II                                 -> 802.3
4               IPX ETHERNET_802.3                              -> 802.3
5               IPX ETHERNET_802.3/FDDI/TOKEN_RING              -> 802.3
6               IPX ETHERNET_SNAP/FDDI_SNAP/TOKEN-RING_SNAP     -> 802.3
```

If this port is an Ethernet media port, enter **r** at the system prompt and then press **<Return>**. If you do this for a port other than an Ethernet port, this will be ignored.

If the port selected is an Ethernet based port, something like the following would be displayed:

```
Ethernet Statistics for Ethernet port 3/4/Brg/1
Received Good Octets              0     Transmitted Good Octets          0
Received Bad Octets               0
        Total Octets              0
Received Unicasts                 0     Transmitted Unicasts             0
Received Multicasts               0     Transmitted Multicasts           0
Received Broadcasts               0     Transmitted Broadcasts           0
Received Buffer Discards          0     Transmitted Buffer Discards      0
Received Collision Count          0     Transmitted Retry Count          0
Received Runt Count               0     Transmitted More Count           0
Received Error Discard            0     Transmitted Once Count           0
Drop Event Count                  0     Transmitted Defer Count          0
Received Jabbers                  0     Loss Carrier Count               0
Received Over Size                0      Transmitted Late Collisions     0
Received Late Collision           0     Transmit Underflow               0
Received 1024 +                   0     Port Filtered                    0
Received 512 +                    0     Vlan Filtered                    0
Received 256 +                    0     Mtu Exceeded                     0
Received 128 +                    0
Received 65 +                     0
Received 64                       0
vseTxDiscard                      0
```

The fields displayed by the **r** option of the **swch** command are described below:

♦ Note ♦

The first group of statistics are the numbers of bytes transmitted and received. These are useful in working out bandwidth usage by the port. Bad octets are important to count in the total octets count as they consume bandwidth at the expense of useful traffic. To ignore them would lead to mysterious loss of bandwidth in any calculations performed.

Received Good Octets. The total number of bytes received in good frames.

Received Bad Octets. The total number of bytes received in bad frames.

Total Octets. The total number of octets transmitted or received in good or bad frames on this port.

Transmitted Good Octets. The total number of bytes successfully transmitted.

Received Unicasts. The number of frames received on this port whose destination address is a unicast format.

Transmitted Unicasts. The number of frames transmitted on this port whose destination address is a unicast format.

Received Multicasts. The number of frames received on this port whose destination address is a multicast format.

Transmitted Multicasts. The number of frames transmitted on this port whose destination address is a multicast format.

Received Broadcasts. The number of frames received on this port whose destination address is the broadcast address.

Transmitted Broadcasts. The number of frames transmitted on this port whose destination address is the broadcast address.

Note that these statistics merely indicate the format of the destination address of frames transmitted/received on this port, not that the addressed device and/or devices necessarily reside on that port. For example, unknown unicast addressed frames are flooded to many ports.

Received Buffer Discards. The number of frames discarded due to congestion of traffic from multiple ports on the board.

Transmitted Buffer Discards. The number of frames received from the backplane destined to this port that were dropped.

Transmit Underflow. Due to congestion of traffic from multiple ports on the board, timely access to the buffer containing the frame currently being transmitted by this port was not obtained and the frame had to be aborted and discarded.

vseTxDiscard. Due to congestion of traffic from multiple ports and boards in the system, traffic received from the network port could not be queued to the backplane due to buffer availability.

Received Collision Count, Received Runt Count. These counts may be considered normal on a shared segment (e.g., AUI and BNC connected Ethernet) where more than two stations exist. The first indicates that a frame which the port started to receive from a station was subjected to a collision from a third station. This is normal. Such collisions between third party stations may cause this port to see fragments of a frame which are discarded as runts. This too is normal on multiple station Ethernet segments. On point to point 10Base-T connections these events may be considered abnormal, indicating a possible intermittent wiring problem (unless hubs which propagate fragments are in use). These statistics do not indicate the loss of any frame but rather events associated with the attempts to finally successfully transfer the frame.

Transmitted Defer Count, Transmitted Once Count, Transmitted More Count, and Transmitted Retry Count. These statistics are all related to collisions and deferrals, where this port is actively trying to transmit a frame. The CSMA part of CSMA/CD (the protocol of Ethernet) requires that a station that wishes to transmit must first listen to the media to see if a transmission is already in progress. If it is, then the station must defer transmission until the media is quiet. The **defer count** is the number of times this happens. A high defer count, relative to total numbers of frames transmitted by the port, can be indicative of a busy segment.

If a transmission is not in progress, the station may begin to transmit. Due to propagation delays it is possible for a station to suffer a collision from another station trying to transmit, even though both listened for quiet media. When this occurs, both stations "back off" for a random time before attempting transmission again. In theory, subsequent collisions may occur on these retries. **Once**, **more**, and **retry** indicate whether this is occurring. If a collision occurs but succeeds on the retry, the **once** counter is incremented (i.e., we collided once). If more than one retry is required, the **more** count is incremented. If up to 16 retries are attempted and all collide, then the frame is dropped and the **retry** count is incremented. High numbers, relative to total transmitted frames, are again indicative of a very busy segment whose throughput could be increased by further segmentation.

Received Error Discard. A frame was received with an FCS and/or alignment error. A high count here, relative to total received frames, is indicative of a noisy media subject to errors.

Loss Carrier Count. This is a count of transmitted frames which are lost due to a loss of carrier. This is indicative of poor quality/noisy wiring or adapter cards.

Received Late Collision, Transmitted Late Collisions. A late collision is a collision which occurs in a frame when more than 64 bytes have been received/transmitted. On a correctly configured network, which doesn't exceed physical limits of size, impedance, station spacing, etc., stations should always collide within 64 bytes due to propagation times. **Late collisions** indicate that the network is violating such restrictions or some stations are having a problem which prevents them correctly implementing the CSMA/CD protocol. For example, a station with a faulty receiver can not "hear" transmissions in progress and so may fail to defer its transmissions causing late collisions to be seen by other stations.

Received Jabbers, Received Over Size. The maximum frame size on Ethernet is 1518 bytes. Frames longer than this are illegal. When such a frame has a valid FCS it is counted as **oversized**. If it has an FCS error then it is counted as a **jabber**. The former is indicative of a device with improper software, the latter of a device with some hardware fault on its transmitter. In both cases the faulty station causes other devices, such as this port, to see these errors.

Drop Event Count. When a frame is dropped (for example, frame reception is aborted because of lack of buffers) there is a single occurrence of an "event" during which frames were lost. This statistic counts then number of events.

Received 1024 +, Received 512 +, Received 256 +, Received 128 +, Received 65 +, and Received 64. These count the number of frames in the indicated frame sizes: **Received 64** counts 64 byte frames, **Received 65+** counts frames between 65 and 127 inclusive, **Received 128+** counts between 128 and 255, etc. These statistics are only applied to received frames.

Port Filtered. On shared media ports, Station A transmitting to Station B will be directly delivered. Therefore, the frame received by this port just needs to be dropped. This action is referred to as filtering and this counts the number of frames so filtered.

Vlan Filtered. The OmniAccess 408 restricts traffic above the normal Level 2 filtering by applying VLAN rules. Frames which are dropped because of VLAN rules are counted here.

Mtu Exceeded. This statistic is not currently supported and is always zero.

Any to Any MAC Translations

The **swchmac** command allows you to view the current frame translation settings for a given MAC address. Follow these steps:

1. Enter **swchmac** and the following prompt displays:

   Enter MAC address ([XXYYZZ:AABBCC] or return for none :

2. Enter the MAC for which you want to view translations. The following prompt displays:

   Is this MAC in Canonical or Non-Canonical (C or N) [C] :

3. Enter if the MAC address you entered is expressed in canonical (**C**) or non-canonical format. The default is canonical. A screen similar to the following displays:

```
Port Translations for Ethernet port 3/4/brg/1

Translation Options:
        IP Ethertype                                    -> Ethertype
        IP IEEE 802 SNAP                                -> Ethertype

        IPX ETHERNET_II                                 -> 802.3
        IPX ETHERNET_802.3                              -> 802.3
        IPX ETHERNET_802.3/FDDI/TOKEN_RING              -> 802.3
        IPX ETHERNET_SNAP/FDDI_SNAP/TOKEN-RING_SNAP     -> 802.3

        Proprietary Token Ring IPX Switching            -> Off
```

The screen shows how each incoming frame type is translated. The frame type in the left column indicates the incoming frame type, and the frame type in the right column (after the **->**) indicates the outgoing frame translation.

Default Autoencapsulation

Autoencapsulation is a technique employed by AutoTracker software to learn the protocol and encapsulation type used by a source MAC address and automatically translate frames bound to that MAC address to the appropriate encapsulation type.

Normally all devices attached to a switch port receive frames translated according to the translation options defined for that port. However, some devices attached to the same port may require different frame formats.

For example, one workstation may support IPX 802.3 frames and another may support IPX SNAP frames. The switch port may be configured to translate incoming IPX 802.3 frames to LLC frames, which would not satisfy either of the workstations. If autoencapsulation is on, then the switch would translate frames for the first workstation to IPX 802.3 and frames for the second workstation to IPX SNAP. The translation setting for the port is overridden for those ports that require a special translation.

Autoencapsulation operates only on learned unicast frames. It does not work for broadcast, multicast, or unlearned unicast frames. For this reason is recommended only for ports attached to client devices. It is not recommended for ports attached to servers due to high volume of broadcast traffic on such a connection.

In addition, autoencapsulations is not supported for Banyan Vines frames. It operates only on IP and IPX frames.

To turn on autoencapsulation type the following at the prompt:

autoencaps on

To turn off autoencapsulation type the following at the prompt:

autoencaps off

Translational Bridging

Translational Bridging enables internetworking between FDDI, Ethernet, and Token Ring LANs. There is no standard which encompasses this. The OmniAccess 408's features focus on bridging of frames between media and translating the MAC and LLC headers into the appropriate "native" frame formats. This provides media-independent internetworking.

Learning

For VLAN trunk frames, the switch will learn the source MAC address of the encapsulated frame and associate this with the source MAC address of the originating switch. When a frame arrives, the switch checks to see if the frame has been learned. If so, then the frame will be encapsulated and sent directly to the destination switch. If not, then the switch will learn the association of VLAN, trunk service, virtual port, source, and destination MACs. If the switch has no ports in the VLAN associated with the frame's destination, the frame is dropped.

Translations across Trunks

The OmniAccess 408 sends frames onto the trunk in the same format as the original LAN type. Any required translation is done at the destination switch.

Dissimilar LAN Switching Capabilities

Switching traffic between like media requires no changes to the frame, whereas switching traffic between unlike media requires some level of change to the frame. To fully explain the various changes possible we need to define the portion of the frame where changes could occur.

Media Specific fields and MAC address fields are different for Token Ring, FDDI, and Ethernet. For Token Ring and FDDI, the switch generates MAC addresses in non-canonical format, where Ethernet generates MAC addresses in canonical format. The OmniAccess 408 will perform media translations which means the media specific, source MAC and destination MAC will be changed for each frame which changes media.

The source routing field is optional, and use of this field is driven by endstations who wish to communicate using source routing. The OmniAccess 408 participates in source routing on FDDI and Token Ring interfaces when it is configured as a Source Route Bridge. The OmniAccess 408 will also forward source route frames transparently while performing standard switching of frames on Token Ring and FDDI interfaces as well as when using the virtual ring feature.

The encapsulation type field can be a number of different encapsulations, which really includes the Media Specific fields, source MAC address, and destination MAC address. The choices are Ethernet II, IEEE 802.2 (LLC), SNAP, and Novell 802.3 or FDDI proprietary formats. There are configuration options for Ethernet, FDDI, and Token Ring interfaces. The encapsulation type field may or may not be changed. This decision is made based on the incoming encapsulation type, the user configuration, and the topology that frame is traveling.

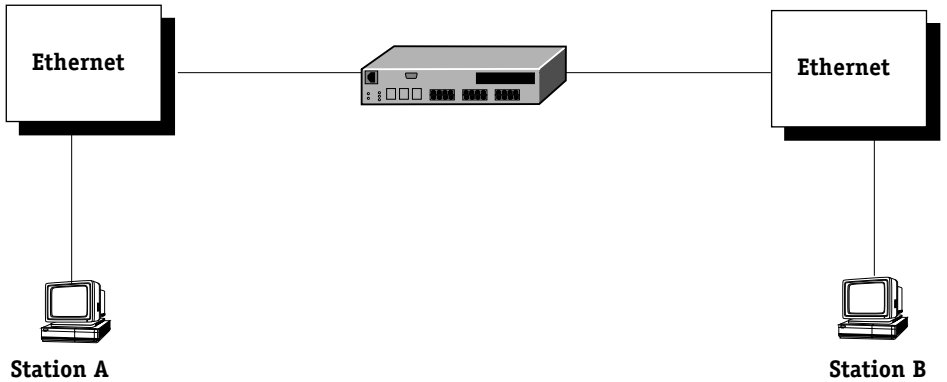
The data field is the remainder of the frame which is application dependent. This data field is not changed for switched traffic. Each frame is followed by a CRC.

Below are some examples when translation can occur.

Switching Between Similar LANs

Translations are not performed for switched traffic between similar LANs within one OmniAccess 408. For example in the diagram below, if Station A on an Ethernet segment wants to talk to Station B on another Ethernet segment, the switched frames are not changed.

This is true for any two media where the originating media and the destination media are of the same type (i.e. Ethernet, FDDI, Token Ring).

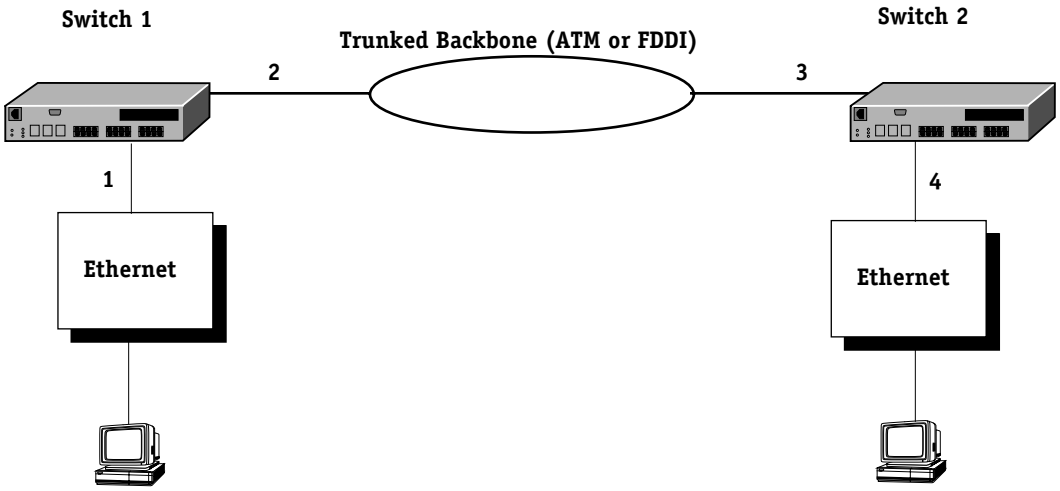


Similar LANs

Switching Between Ethernet LANs Across a Trunked Backbone

Frames that are switched between like media across a Trunked backbone will only be translated at the egress port of the egress OmniAccess 408. For example, in the figure below, frames switched from Station A to Station B will be translated at point 4, where point 4 is the egress port of Switch 2. Frames switched from Station B to Station A will be translated only at point 1, where point 1 is the egress port of Switch 1.

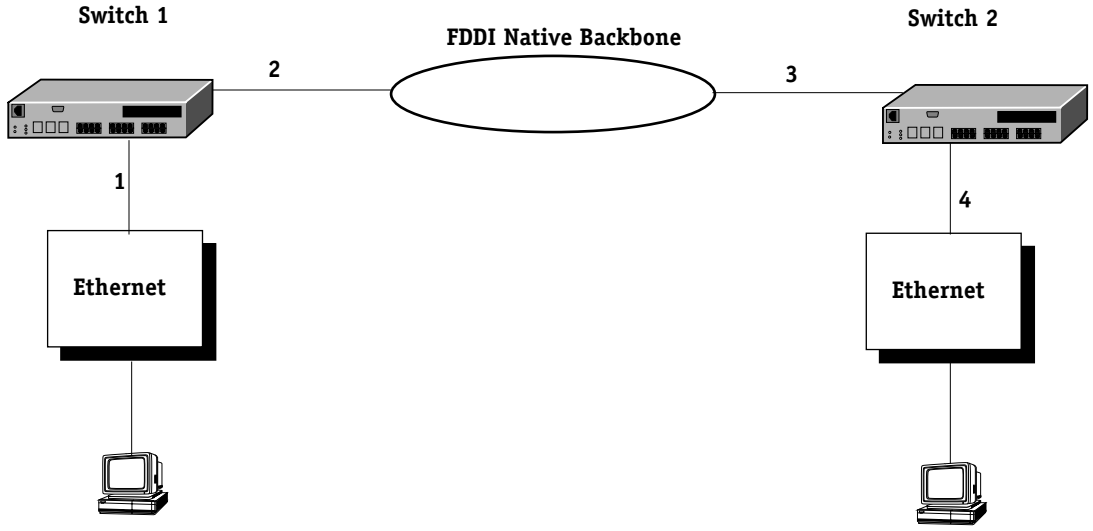
This is true if the originating media and destination media are Ethernet. It is not true if the originating media and destination media are either Token Ring or FDDI.



Ethernet LANs Across a Trunked Backbone

Switching Between Similar LANs across a Native Backbone

Switched traffic between similar LANs across a non-trunked or native backbone will have translations performed at each egress point. In the figure below, for traffic originating from Station A destined to Station B, point 1 represents the ingress (input) port of Switch 1. Likewise, point 2 represents the egress (output) port of Switch 1, point 3 represents the ingress (input) port of Switch 2 and the point 4 represents the egress (output) port of Switch 2. Translations will occur at each egress port. For traffic from Station A to Station B, output translations will occur at points 2 and 4. For traffic from Station B to Station A, output translations will occur at points 3, and 1.



Similar LANs Across a Native Backbone

In the above example, the backbone could be of any media type other than Ethernet. If all three media types were Ethernet, then no translations would occur, because the traffic is being switched from like media to like media.

The following table shows interoperability between dissimilar LANs with two switches where the client and server are resident on like media types and the connection is switched over various LAN backbone types. This table is representative of the IP and IPX protocol only.

	Backbone			
	Token Ring	***FDDI***	***Ethernet***	***ATM***
Token Ring to Token Ring	No	Yes	Yes	No
FDDI to FDDI	Yes	No	Yes	No
Ethernet to Ethernet	Yes	Yes	No	No

Dissimilar LANs