IPX

August 2000

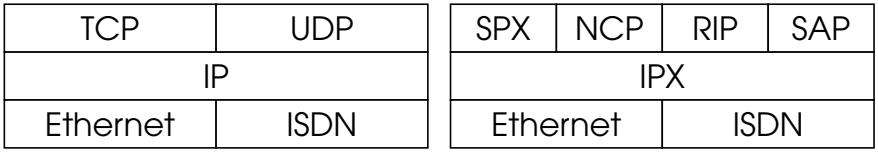
IPX

REFERENCE

1 Configuring the BinTec router as an IPX Router

1.1 Introduction to IPX

IPX (Internetwork Packet exchange) is a Network Layer protocol, similar to IP in TCP/IP. An IPX network allows DOS/Windows PCs to share networked services and devices. Services are provided by special PCs which are assigned the duties of, for example, a file or print server.



TCP/IP Networks IPX Networks

IPX (Internetwork Packet exchange) is a connectionless service used to transmit data.

SPX (Sequenced Packet Exchange) is a connection-oriented service used to monitor connections between stations (e.g., a connection to a print service).

Using RIP and SAP routing and service information is periodically exchanged between IPX routers and servers on the network using the RIP and SAP (Service Advertising Protocol) packets.

1.1.1 IPX Stations: Servers and Clients

In an IPX network, stations on the network are classified as either a client or server; and have different characteristics.

Servers

1. Provide special services, (e.g., remote file access, printing, databank access, etc.) to clients.
2. Have a unique name.
3. Can communicate with both servers and clients.

Clients

1. Use the services provided by server stations.
2. Do NOT have unique names.
3. Can ONLY communicate with servers.

1.1.2 IPX Networks: Network Numbers and Addresses

In an IPX network, a network address consists of:

4 byte Network Number
6 byte Node Number
2 byte Socket Number

In contrast to IP, where hosts are assigned addresses statically, clients are assigned the Network Number portion of their address dynamically. Servers, on the other hand, have their complete address assigned statically.

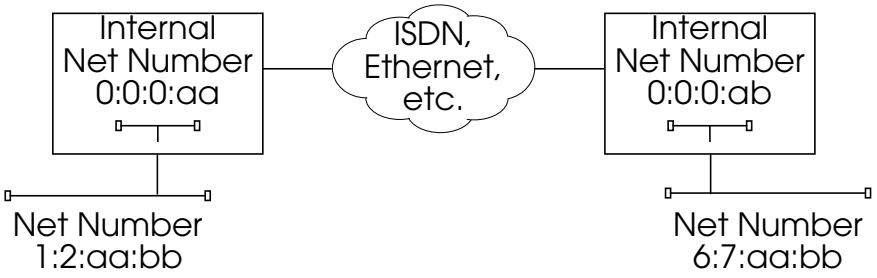
Initially, a client asks for its network number by broadcasting a request. A server or router on the network will answer the request with the correct network number. The client then uses the Network Number (received from the server) and its Node Number (normally the MAC address is used), to establish a connection to a server.

Internal Network Numbers

Since IPX uses each stations MAC address for its network address, stations with more than one interface to the net-

work can be reached at different addresses. This can be a problem for a server that advertises services or an IPX router that links multiple IPX LANs.

To get around this problem, servers and IPX routers are assigned Internal Network Numbers. The respective server or router is the only station on this network. By sending RIP packets, routers and servers can inform other stations on the network.



1.2 Configuring IPX Routing

1.2.1 Adding Routes and Services

Routes and services the BinTec router knows of are learned using the RIP and SAP protocols. This information often changes dynamically. Additional routes and services can be set statically, using the ***ipxStaticRouteTable*** and ***ipxStaticServTable***.

Adding Static Routes

To create a static route to a server the you will need to know the server's internal network number, its name, and the interface the connection should use. The following commands could be used to add a static route to the file server "PHOENIX" which has the internal network number of

0:2:2:2, the route will use the dialup1 interface (*ifIndex* 10001).

```
mybrick : system > ipxStaticRouteTable

inx SysInstance(*rw)     CircIndex(*rw)      NetNum(*rw)        ExistState(-rw)
    Ticks(rw)            HopCount(rw)

mybrick : ipxStaticRouteTable > ipxStaticServSysInstance=0 ipxStaticRouteCircIndex=10001
                ipxStaticRouteNetNum=0:2:2:2
00: ipxStaticRouteSysInstance.0.10001.0.2.2.2( rw):      0
00: ipxStaticRouteCircIndex.0.10001.0.2.2.2( rw):        10001
00: ipxStaticRouteNetNum.0.10001.0.2.2.2( rw):           0:2:2:2

mybrick : ipxStaticRouteTable > ipxStaticRouteTable

inx SysInstance(*rw)     CircIndex(*rw)      NetNum(*rw)        ExistState(-rw)
    Ticks(rw)            HopCount(rw)

 00 0                     10001               0:2:2:2            on
    0                    0

mybrick : ipxStaticRouteTable >
```

Adding Static Services

A service can also be added statically using the ***ipxStaticServTable***. For services, you will also need to know:

- The socket number
- The type of service the server provides
- The server's Node Number.

Note:

For each ***ipxStaticServNetNum***, the BinTec router needs to have a route to the server in ***ipxStaticRouteNetNum***.

The following could be used to add a static service for "PHOENIX" from the previous section.

```
mybrick : ipxStaticRouteTable > ipxStaticServTable

inx SysInstance(*rw)        CircIndex(*rw)        Name(*rw)        Type(*rw)
    ExistState(-rw)         NetNum(rw)            Node(rw)         Socket(rw)
    HopCount(rw)

mybrick:ipxStaticServTable> SysInstance=0 CircIndex=10001 Name=PHOENIX Type=0:4
                            NetNum=0:2:2:2 Node=0:0:0:0:0:1 Socket=4:51

00: ipxStaticServSysInstance.0.10001.7.80.72.79.69.78.73.88.0.4( rw):   0
00: ipxStaticServCircIndex.0.10001.7.80.72.79.69.78.73.88.0.4( rw):     10001
00: ipxStaticServName.0.10001.7.80.72.79.69.78.73.88.0.4( rw):          "PHOENIX"
00: ipxStaticServType.0.10001.7.80.72.79.69.78.73.88.0.4( rw):          0:4
00: ipxStaticServNetNum.0.10001.7.80.72.79.69.78.73.88.0.4( rw):        0:2:2:2
00: ipxStaticServNode.0.10001.7.80.72.79.69.78.73.88.0.4( rw):          0:0:0:0:0:1
00: ipxStaticServSocket.0.10001.7.80.72.79.69.78.73.88.0.4( rw):        4:51

mybrick:ipxStaticServTable> ipxStatisServTable

inx SysInstance(*rw)        CircIndex(*rw)        Name(*rw)        Type(*rw)
    ExistState(-rw)         NetNum(rw)            Node(rw)         Socket(rw)
    HopCount(rw)

 00 0                       10001                 "PHOENIX"          0:4
    on                      0:2:2:2               0:0:0:0:0:1        4:51
    0
mybrick:ipxStaticServTable>
```

1.2.2 Learning Routes and Services

Adding static routes and services for IPX network that change often, or have many servers can be demanding. You can allow the BinTec router to learn of routes and services using RIP and SAP and then have the BinTec router move all learned information to the Static tables. This is done as follows:

1. Enable RIP/SAP for the PPP interface.

2. Wait until the desired routes and services appear in the
   ipxDestTable and ***ipxDestServTable***.
3. Set ***ipxAdminLearnStatics*** to **`both`**.
4. Disable RIP and SAP for the PPP interface.

The result of this is that all routes and services learned from PPP interfaces are copied appended from ***ipxDestTable*** and ***ipxDestServTable*** to the ***ipxStaticRouteTable*** and ***ipxStaticServTable***.

Note:

Each time the BinTec router is allowed to learn statics, the learned information is appended to the Static tables. This may result in duplicate static entries.

1.2.3 Filtering IPX Packets

An important characteristic of IPX networks is the periodic sending of IPX packets between communicating stations over the network. For LAN traffic this is acceptable, but when connecting IPX LANs over ISDN, the amount of RIP and SAP traffic can lead to long (or often) connection times. In addition to the spoofing mechanism IPX traffic can be filtered using the ***ipxAllowTable*** and ***ipxDenyTable***.

For example, serialization packets could be filtered with the following.

```
mybrick : ipxStaticRouteTable > ipxDenyTable

inx PktTypeMode(*-rw)       PktType(rw)          DstIfStatus(rw)
    DstNetMode(rw)          DstNet(rw)           DstNodeMode(rw)
    DstNode(rw)             DstSockMode(rw)      DstSock(rw)
    SrcIfIndexMode(*rw)     SrcIfIndex(rw)       SrcNetMode(rw)
    SrcNet(rw)              SrcNodeMode(rw)      SrcNode(rw)
    SrcSockMode(rw)         SrcSock(rw)

mybrick: ipxDenyTable > DstSockMode=verify DstSock=1111 DstIfStatus=dormant
                        PktTypeMode=dont_verify SrcIfIndexMode=dont_verify

01: ipxDenyDstSockMode.1.1.2( rw):          verify
01: ipxDenyDstSock.1.1.2( rw):              1111
01: ipxDenyDstIfStatus.1.1.2( rw):          dormant
01: ipxDenyPktTypeMode.1.1.2(-rw):          dont_verify
01: ipxDenySrcIfIndexMode.1.1.2( rw):       dont_verify

mybrick : ipxStaticRouteTable> ipxDenyTable

inx PktTypeMode(*-rw)       PktType(rw)          DstIfStatus(rw)
    DstNetMode(rw)          DstNet(rw)           DstNodeMode(rw)
    DstNode(rw)             DstSockMode(rw)      DstSock(rw)
    SrcIfIndexMode(*rw)     SrcIfIndex(rw)       SrcNetMode(rw)
    SrcNet(rw)              SrcNodeMode(rw)      SrcNode(rw)
    SrcSockMode(rw)         SrcSock(rw)

 00 dont_verify             unknown              dormant
    dont_verify             0                    dont_verify
                            verify               1111
    dont_verify             0                    dont_verify
    0                       dont_verify
    dont_verify             0

mybrick:ipxDenyTable>
```

This filter would not allow ISDN connections to be opened for Novell serialization packets. If an ISDN connection is already open, serialization packets would be allowed through. By default this filter is automatically added to the ***ipxDenyTable*** at boot time, and can be removed.

1.2.4 Filtering of Services in IPX Networks (SAP Filters)

If the number of services in an IPX network is very high, this can lead to various performance problems with WAN links or routers because of the periodic sending of SAP packets. Workstations rarely need to see all the services in a network. So the administrator can now solve these performance problems by configuring SAP filters to reduce the number of services to be learned by the BinTec router and to be forwarded to other interfaces.
Filtering of services can be done by:

- interface index
- direction (incoming / outgoing / both)
- service type
- service's network number
- service's network node
- service's socket
- service´s name

It is up to you to decide which criteria to employ by setting the value of the above variables to either *verify* or *dont_verify* (see below). The procedure is similiar to configuring IPX packet filters.

The Variables, Values and their Meanings

Here are the variables, values and meanings of the **SapDenyTable**. Besides the central difference of permission or denial to learn or propagate services, the variables and meanings of the **SapAllowTable** are identical to the variables and meanings of the **SapDenyTable**.

ipDenyIfIndexMode The interface index to be verified or not. Possible

	values: *verify*, *dont_verify*, *delete* Default: *dont_verify*
sapDenyIfIndex	This rule is applied to services originating from or (see **sapDenyDirection**) destined for the interface with this index number. If, in the case of a service known to the BinTec router and where the service name is entered, the **IfIndex** is set to *0* and a direction is set to either *incoming* or *outgoing,* all interfaces are affected by the rule. If, however, the service name is used and the **IfIndex** is set to *0,* but NO direction is given, the entry will assume the interface over which that service was learned and the direction will be set to *incoming*.
sapDenyDirection	The direction that is to be subject to the rule. Possible values: *incoming*, *outgoing*, *both*, *dont_verify.*
sapDeny*TypeMode*	The SAP service type to be checked or not. Possible values: *verify*, *dont_verify.*
sapDeny*Type*	The various SAP service types to be checked. For example: 4: file server, 7: print server.
sapDeny*NetMode*	The network number to be checked or not. Possible values: *verify*, *dont_verify.*
sapDeny*Net*	The service´s network number to be checked.

sapDeny*NodeMode*	The node number to be checked or not. Possible values: *verify*, *dont_verify.*
sapDeny*Node*	The service´s node number to be checked.
sapDeny*SockMode*	The socket number to be checked or not. Possible values: *verify*, *dont_verify.*
sapDeny*Sock*	The service´s socket number to be checked.
sapDeny*Name*	Instead of entering Type/Net/Node/Socket directly, you need only fill in the service name here, provided the service has been learned by the BinTec router IPX. The values of the Type/Net/Node/Socket fields contained in the **ipxDestServTable** will then be copied to the **sapDenyTable**.

Examples

In order to create SAP filters for the services of a file server, entries must be made in the **sapDenyTable** and/or in the **sapAllowTable**: in the first, to specify the services to be prevented from being learned or propagated; and in the second, to specify those to be allowed to be learned or propagated.
To block or allow a single service the administrator has to look up type, net, node and socket in the **IpxDestServTable** or at the server's console. Then these values can be used to create an entry in the **sapDenyTable** or **sapAllowTable**.
A service *x* is allowed to enter or leave the BinTec router if:

1. it matches an entry in the **sapAllowTable** and there is no matching entry in the **sapDenyTable**,
2. there is no entry in the **sapAllowTable** and no matching entry in the **sapDenyTable**,
3. there is no entry in either table.

A service *y* is denied entry to or exit from the BinTec router if:

1. it matches an entry in the **sapDenyTable**, there is no entry in the **sapDenyTable** and no matching entry in the **sapAllowTable**.

Let´s have a look at some of the various configuration scenarios:

- You could specify only those services you wish to allow the BRICK to propagate over one particular interface; all other services are prevented from being propagated over that interface. This would be done by

making outgoing entries in the **sapAllowTable** over the interface 10001, for example:

```
brick:sapAllowTable
inx IfIndexMode(-rw)    IfIndex(*rw)      Direction(rw) TypeMode(rw
    Type(rw)            NetMode(rw)       Net(rw)       NodeMode(rw)
    Node(rw)            SockMode(rw)      Sock(rw)      Name(rw)

brick:sapAllowTable> IfIndexMode=verify ifindex=10001 direction=outgoing typemode=ver-
ify type=0:4 netmode=verify net=172:36:10:62
00: sapAllowIfIndex.0(rw):          10001
00: sapAllowDirection.0(rw):        outgoing
00: sapAllowTypeMode.0(rw):         verify
00: sapAllowType.0(rw):             0:4
00: sapAllowNetMode.0(rw):          verify
00: sapAllowNet.0(rw):              172:36:10:62

brick:sapAllowTable> sapAllowTable
inx IfIndexMode(-rw)    IfIndex(*rw)      Direction(rw) TypeMode(rw
    Type(rw)            NetMode(rw)       Net(rw)       NodeMode(rw)
    Node(rw)            SockMode(rw)      Sock(rw)      Name(rw)

    verify              10001             outgoing      verify
    0:4                 verify            172:36:10:62  dont_verify
                        dont_verify
brick:sapAllowTable
```

- You could, of course, specify only those services you wish to prohibit the BRICK to propagate; all others are propagated. This would be done by making outgoing entries in the **sapDenyTable**. In this case, as the service is known to the BRICK, it is sufficient to merely enter the name of the service, the direction and the interface, the rest (Type/Net/Node/Socket) will be read from the **ipxDestServTable**. In the following example where the BRICK has already learned the ser-

vice and the service name is being used and index=0 and direction=outgoing, all interfaces are affected:

```
brick:sapDenyTable
inx IfIndexMode(-rw)    IfIndex(*rw)      Direction(rw) TypeMode(rw
    Type(rw)            NetMode(rw)       Net(rw)       NodeMode(rw)
    Node(rw)            SockMode(rw)      Sock(rw)      Name(rw)

brick:sapDenyTable> ifindex=0 direction=outgoing name=FILESERVER
00: sapDenyIfIndex.0(rw):          0
00: sapDenyDirection.0(rw):        outgoing
00: sapAllowTypeMode.0(rw):        FILESERVER
brick:sapDenyTable> sapDenyTable>
inx IfIndexMode(-rw)    IfIndex(*rw)      Direction(rw) TypeMode(rw
    Type(rw)            NetMode(rw)       Net(rw)       NodeMode(rw)
    Node(rw)            SockMode(rw)      Sock(rw)      Name(rw)

    dont_verify         0                 outgoing      verify
    0:4                 verify            aa:bb:cc:dd   verify
    0:0:0:0:0:1         verify            40:00         FILESERVER
```

- Alternatively, you could specify those services you wish to prohibit from being learned by the BRICK; all other services are learned and propagated. This would be done by making incoming entries in the **sapDenyTable**.
- You could specify only those services you wish to allow the BRICK to learn; all others are denied access. This would be done by making incoming entries in the **sapAllowTable**.
- Finally, it is possible to make entries in both tables. In this case, you would explicitly specify which services are to be denied and which are to be allowed. This would involve either incoming or outgoing entries in both tables.