Workgroup Access 1MB (BIANCA/BRICK-XM, BIANCA/BRICK-XS)

Final Release Notes

System Software Release 5.1 Revision 2

September, 2000

New System Software

Release 5.1 Revision 2

This document describes the new features, changes, bugfixes and known issues contained in Release 5.1 Revision 2 for the 1 MB Workgroup Access product group (BIANCA/BRICK-XM and BIANCA/BRICK-XS).

This special release intended only for the above 1 MB products is the **FINAL** software release for these products. The following subsystems are not included in this release:

- X.25
- Bridging
- OSPF

Bridging and OSPF were also not included for the BIANCA/BRICK-XM in the previous software release.

From Release 5.1 Revision 1, Release Notes of Bintec´s software images are divided into three categories based on the three product groups: Corporate Access, which includes BIANCA/BRICK-XL/XL2 and BIANCA/BRICK-XMP; Workgroup Access, which includes BIANCA/BRICK-XM2 and BIANCA/BRICK-XS2; and Personal Access, including BinGO!

1 Upgrading System Software

- Retrieve the current system software image from BinTec's WWW server at http://www.bintec.de (Section: Download).
- With this image you can upgrade the BIANCA/BRICK with the `update` command from the SNMP shell via a remote host (i.e. using telnet, minipad, or isdnlogin) or by using the BOOTmonitor, if you are logged in directly on the console.
  Information on using the BOOTmonitor can be found in the BIANCA/BRICK User's Guides under Firmware Upgrades.

Caution!

- Do not update your Logic or BOOTmonitor images unless expressly instructed to do so.
  Normally, it is not necessary to upgrade these images. Only in exceptional cases is an upgrade explicitly recommended.

  The upgrade process involves an element of risk. Should the update of either of these files fail as the result of a power failure for example, the BRICK may not be able to boot.
- If you are unsure whether to upgrade or not, read the BOOTmonitor and Firmware Logic Release Notes (available below the images on the FTP server, section: Download) where you will find tables that specify the appropriate Logic and BOOTmonitor versions available for your BinTec product, and if an update is recommended or not.

Please note that there is an update procedure in case there is not enough memory available to perform a software update via the `update` command from the SNMP shell. The incremental update loads the new software image in blocks of 64 KB via TFTP and writes it to the Flash ROM immediately.

Because this procedure offers no possibility to check the integrity of the image:

- first use the option “-v” that verifies the image file.

- Once you’ve installed Release 5.1 Revision 2, you may want to retrieve the latest documentation (in Adobe’s PDF format), which is also available from BinTec’s WWW server at the address noted above.

When upgrading system software, it is also recommended that you use the most current versions of BRICKware for Windows and UNIX Tools. Both can be retrieved from BinTec’s WWW server.

If you are updating from a software release equal to or older than 4.7.x to the current version, 5.1.2 or later, it is necessary to firstly upgrade to 4.9.3, save the configuration, then upgrade to 5.1.2. If you update directly from 4.7.x to 5.1.x, the old IP access lists can not be automatically converted and are lost.

2 New Features in Release 5.1.2

2.1 Windows Activity Monitor

With Release 5.1.2, you can configure a **BRICK** to be monitored by the new Windows Activity Monitor which will be available with the new BRICKware Release 5.1.1.

Why the Activity Monitor?

With the Activity Monitor, Windows users can monitor the activities of a **BRICK**. Important information like system status of physical interfaces (e. g. ISDN line) and virtual interfaces (e. g. WAN partners) are easily available with ONE tool. A clear and complete overview of the load of a **BRICK**'s interfaces is possible at any time. The following illustration shows the state of a CM-PRI interface.

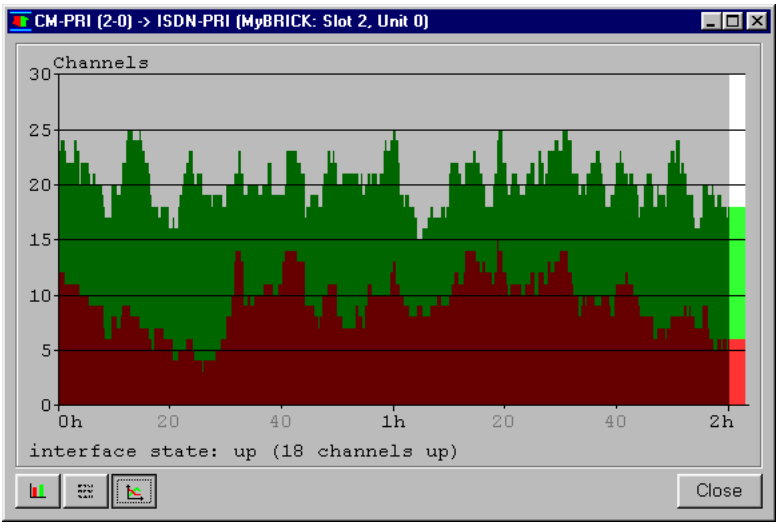

Figure 2-1: Activity Monitor display of a CM-PRI interface

How does it work?

A status daemon collects information about the **BRICK** and transmits it in the form of UDP packets to the LAN's broadcast address (default) or to a specified IP address. One packet per **BRICK** interface and time interval, which is individually adjustable from 1 to 60 seconds, is sent. All physical interfaces and up to 100 virtual interfaces can be monitored unless the packet size of approx. 4000 bytes is not exceeded. A Windows application on your PC, available with the BRICKware Release 5.1.1, receives the packets and displays the information in different ways.

To activate the Activity Monitor you have to

- configure the **BRICK**(s) to be monitored (this step is described in this Release Note)
- start the Windows application on your PC and use it (this step will be described in the new, updated **BRICKware for Windows**, available with the BRICKware Release 5.1.1)

2.1.1 Configure the BRICK

You can perform the required configuration steps on the **BRICK** via

- MIB variables
- Setup Tool

MIB variables:

The configuration is made by entries in the MIB table **ExtAdmin** using the following MIB variables:

Variable	Meaning
ExtAdminMonPort	Number of port for the Activity Monitor (default: *2107*, registered by the IANA - Internet Assigned Numbers Authority).
ExtAdminMonAddress	IP address to which the **BRICK** sends the UDP packets. With the default value 255.255.255.255 the broadcasrt address of the first LAN interface is used. Be aware that if you enter the IP address of a WAN partner, connections liable to charges will be made at very regular intervals (default is every 5 seconds).
ExtAdminMonType	Type of information sent with the UDP packets to the Windows application. Possible values: ■ *off:* deactivates the Activity Monitor (default value) ■ *physical:* only information about physical interfaces ■ *physical_virt:* information about physical and virtual interfaces
ExtAdminMonUpdate	Update time in seconds. Possible values: *0* to *60* (default: *5*).

Table 2-1: **ExtAdmin**

To change the settings, make the following entry (as an example):

```
biboExtAdmMonAddress( rw):      192.168.1.1
biboExtAdmMonPort( rw):         2107
biboExtAdmMonType( rw):         physical_virt
biboExtAdmMonUpdate( rw):       5
```

Setup Tool

The configuration is made in ***SYSTEM*** ➧ ***EXTERNAL ACTIVITY MONITOR:***

```
BRICK Setup Tool                                BinTec Communications AG
[SYSTEM][ACTIVMON]:External Activity Monitor                       MyXS


    Client IP Address               192.168.1.1
    Client UDP Port                 2107
    Type                            physical_virt
    Update Interval (sec)           5


                    SAVE                          CANCEL

Use <Space> to select
```

Field	Meaning
Client IP Address	See **ExtAdminMonAddress** above.
Client UDP port	See **ExtAdminMonPort** above.
Type	See **ExtAdminMonType** above.
Update Interval (sec)	See **ExtAdminMonUpdate** above.

Table 2-2: ***SYSTEM*** ➧ ***EXTERNAL ACTIVITY MONITOR***

Proceed as follows:

- Go to ***System*** ➧ ***External Activity Monitor.***
- Enter **Client IP Address**, **Client UDP port**, **Type** and **Update Interval (sec)**.
- Press **SAVE**.

2.1.2 Windows Application

How to use the Activity Monitor on your PC will be described in the new, updated **BRICKware for Windows** (available with the new BRICKware Release 5.1.1).

2.2 ISDN Channel Reservation

In order to give you even more control over the number and direction of your calls, the **isdnCreditsTable** has been extended. Up to now it was already possible to specify the maximum number of allowed incoming connections over a certain definable period of time (**isdnCreditsMeasureTime**). Once these limits were reached, they could not be exceeded and no more calls could be made in the direction in which the limitation was set. If a hundred outgoing calls could be made over the period of one day, in theory all those calls could be made in the first hour. This would have meant that during that hour it may have been very difficult to receive any incoming calls at all.

Now, however, this new feature allows you to set the maximum number of incoming calls, outgoing calls, as well as the total number of calls being made at the current moment in time.

Example 1:

This means that if it is very important for you that at least half of your PRI B-channels (30 in total) remain open for incoming calls, you can set one of the three new variables, **MaxCurrentOutCon** (the maximum number of outgoing

calls), to 15, this would mean that at least 15 channels would be reserved for incoming calls.

Example 2:

You may only have two B-channels available. In such a case, it might be prudent to set the maximum number of outgoing calls to 1 as well as the maximum number of incoming calls to 1. You could thus limit the likelihood of missing any important incoming calls. It would, however, mean that, if one outgoing call were being made, an attempt to make a second outgoing call would not be possible.

Finally, you can limit the total number of both incoming and outgoing current connections. You could specify that only twenty of your B-channels are to be made available at any one given time. This last option can only be configured directly in the **isdnCreditsTable** not, however, over Setup Tool.

Configuration

The three new variables that have been added to the **isdnCreditsTable** can be configured in the MIB using the SNMP shell:

Variable	Meaning
MaxCurrentInCon	This variable allows you to set the maximum number of current incoming connections
MaxCurrentOutCon	This variable allows you to set the maximum number of current outgoing connections.
MaxCurrentCon	This variable allows you to set the maximum number of incoming as well as outgoing calls currently being made.

Table 2-3: **isdnCreditsTable**

MaxCurrentInCon and **MaxCurrentOutCon** can be configured in Setup Tool in the menu **ISDN** ➧ **CREDITS** ➧ **EDIT**.

```
 BRICK Setup Tool                              BinTec Communications AG
[ISDN][CREDITS][EDIT]: Configure ppp Credits                       MyXS

Surveillance                                                    on

Measure Time (sec)                                              6400

Maximum Number of Incoming Connections                          off

Maximum Number of Outgoing Connections                          on
                                                                100
Maximum Charge                                                  off

Maximum Time for Incoming Connections (sec)                     on
                                                                28800
Maximum Time for Outgoing Connections (sec)                     on
                                                                28800
Maximum Number of Current Incoming Connections                  on
                                                                1
Maximum Number of Current Outgoing Connections                  on
                                                                1

                SAVE                                CANCEL

Enter integer range 0..2147483647
```

- Activate **Maximum Number of Incoming/Outgoing Connections** by pressing the Spacebar and changing *off* to *on*.
- Enter the number of B-channels you want to reserve for that direction.
- Press **SAVE.**

In the above example, the two new variables at the bottom of the table are both set to one. This means that it is impossible to make more than one incoming or outgoing call at any one given time.

Surveillance

It is also possible to observe the number of connections currently being made by going to ***Monitoring and Debugging*** ➧ ***ISDN Credits*** ➧ ***Select Subsystem.***

```
BRICK Setup Tool                                BinTec Communications AG
[Monitor][CREDITS][STAT]: Monitor ppp Credits                       MyXS


                                                 Total Maximum %reached

Time till end of measure interval (sec)          84400 86400    2

Number of Incoming Connections                   1              1
Number of Outgoing Connections                   1              1

Time of Incoming Connections                     734
Time of Outgoing Connections                     244

Charge                                            0

Number of Current Incoming Connections            22     22
Number of Current Outgoing Connections            6      12
Number of Current Connections                     28     28

Exit
```

In this case, the new variables show that twenty-two incoming calls are being made out of a possible total of twenty-two, only six of the permitted twelve outgoing calls are being made, however, because the maximum number of current connections is restricted to twenty-eight.

2.3 Improved Bandwidth On Demand

2.3.1 Bandwidth On Demand for leased lines

In order to support high levels of traffic over leased lines, bandwidth on demand over different media, i.e. dynamic channel bundling of leased lines and dialup lines is possible.

You can bundle channels over an optional table, the **pppExtIfTable**, which can also be configured over Setup Tool. You can, for example, specify if you want one or more dialup lines to be bundled with your leased line when the load on the leased line reaches a certain level for a certain period of time. You can also

specify the kind of algorithmus used by which the load of your leased line or bundle is calculated or the number of seconds such a sample calculation should take, giving you more control over the switching of dialup lines.

BOD is activated by setting the **pppExtIfBodMode** variable to *BOD_active* or *BOD_passive.* Switching on and off of additional B-channels only occurs in the active mode, i.e. one partner must be configured as an active part the other as a passive part, otherwise call collisions are unavoidable.

Dialup lines are dynamically switched on and off when the line utilization of the leased line/bundle reaches a certain level for a certain period of time, see Switchover thresholds for BOD.

2.3.2 Backup for leased line connections

Backup operation is also available for leased line connections. Should a leased line fail, for example, a dialup connection is dynamically initiated.

There is no necessity to define another interface for the backup case. IPX back-up configuration is also possible.

2.3.3 BOD when leased line is down

Bandwidth On Demand is also available for backup connections, the modes used can be either *BOD_active*, *BOD_passive* or *BOD_backup*. To avail of BOD when the leased line is down, the maximum number of switchable B-channels in the **biboPPPTable** (**biboPPPMaxConn**) must be correspondingly configured, i.e. set to greater than *1*.

If the leased line fails, the first switchable B-channel is used as the backup channel, so a second is required to provide this channel with Bandwidth On Demand. Load-dependent switching then occurs from the side that established the backup connection.

2.3.4 Bandwidth On Demand for pure dialup lines

The extensions to the **pppExtIfTable** can also be used for pure dial-up interfaces. In contrast to the behaviour of leased-line interfaces, there is no static configuration of the partner to activate switching. The **pppExtIfBodMode** variable should always be set to *BOD_enabled* to activate BOD for a dialup interface**.** Load-dependent switching then always occurs from the side that established the initial connection.

The new MIB table variables in the PPP group (*pppExtIfTable*) with example and default values:

```
inx   Index (*rw)       BodMode (-rw)     Algorithm (rw)
      Interval (rw)     Load (ro)         MlpFragmentation (rw)
      MlpFragSize (rw)

      5000              backup            equal
      5                 0                 proportional
      50
```

The meanings of the different variables:

Variable	Meaning
IfIndex	Interface index of the relevant leased line interface.

Variable	Meaning
BodMode	Contains the following values: *Bandwidth On Demand disabled* No extra channels are opened to support leased lines. Default value. *Bandwidth On Demand backup* (only for leased lines) If the leased line fails, backup operation is activated. When the leased line is available again, the backup connection is terminated. BOD is also available for this mode, provided **biboPPPMaxConn** is set to more than 1. *Bandwidth On Demand active* (only for leased lines) Only one partner should be configured as the active partner. Switching on and off of additional B-channels then only occurs from this side. *Bandwidth On Demand passive* (only for leased lines) No switching on and off of additional B-channels occurs from the side of the partner configured as BOD_passive. He participates as a passive partner in the channel bundle. *Bandwidth On Demand Enabled* (only for dialup lines) Activates BOD, additional channels can be opened. The partner that initiates the connection carries the charges.

Variable	Meaning
Algorithm	An algorithm for the weighting of throughput within a specified time frame for the calculation of capacity utilization (load). The following values are possible: *equal*: constantly weighted load over the given time frame. This means that all values considered in the time frame contribute in equal measure to the calculation of load. *proportional*: current proportional weighting within the time frame in favour of the latest throughput values. This means the calculation of **load** is influenced least by the first throughput value in the time frame and most by the most recent value added to the time frame.
Interval	Maximum period in seconds for a throughput measurement sample which is used to calculate the load.
Load	A calculation of the capacity utilization of the bundle in % depending on the selected algorithmus and used to determine whether another line should be switched either on or off.

Variable	Meaning
MlpFragmentation	A mode according to which MLP fragments are formed. *proportional*: the fragment size taken from the available bandwidth of the individual links in relation to the total bandwidth of the bundle. This means that if, for example, an X.21 link has 128 000 kbit/s bandwidth and dialup lines 2 and 3 have 64 000 kbit/s each, the X.21 link will receive a greater proportion of each packet, i.e. a larger fragment, than lines 2 and 3. *equal*: as far as is possible, fragments of equal size are formed, the weighting of the bandwidth of a link is made by the number of fragments to be sent on the link. Using the above example, the leased line simply receives more fragments of packets; the fragments, however, remain of equal size.
MlpFragSize	Here you can configure the size of the fragment to be sent. If you have left the setting *proportional* above (default), the maximum size is used of the number you have configured (you can, of course, leave the default value of 50) and the calculation according to the **MlpFragmentation** mode. If you have selected equal above, the number you configure represents the size of the fragments.

Table 2-4: **PPPExtIfTable**

MlpFragmentation and **MlpFragSize** can not be configured in Setup Tool and should be left with the default values which are *proportional* and *50* respectively.

Setup Tool configuration

➤ In the main menu of Setup Tool, go to **WAN Partner**.

```
BIANCA/BRICK-XS Setup Tool                    BinTec Communications AG
[WAN]: WAN Partners                                           mybrick

Current WAN Partner Configuration

Partnername                  Protocol                State
Leased, Slot 2 (0)           ppp                     down
Partner_1                    ppp                     down



    ADD              DELETE                    EXIT

Press <Ctrl-n>, <Ctrl-p> to scroll, <Space> tag/untag DELETE,
<Return> to edit
```

➤ Select the configured WAN partner you want Bandwidth On Demand for, in the example Partner 1. A leased line is configured for this partner..

```
BIANCA/BRICK-XS Setup Tool                    BinTec Communications AG
[WAN][EDIT]: Configure WAN Partner                            mybrick


Partner Name                              Partner 1


Encapsulation                             PPP
Compression                               none
Encryption                                none

PPP >
Advanced Settings >

IP >
IPX >

            SAVE                              CANCEL

Enter string, max length = 25 Chars
```

➤ Select **Advanced Settings**.

```
BIANCA/BRICK-XS Setup Tool                  BinTec Communications AG
[WAN][EDIT][ADVANCED]:Advanced Settings (Partner 1)            mybrick


    Extended Interface Settings (optional)

    Layer 1 Protocol                          ISDN 64 kbps


                    OK                        CANCEL

Use <Space> to select
```

➤ Select **Extended Interface Settings**

```
BIANCA/BRICK-XS Setup Tool                  BinTec Communications AG
[WAN][EDIT][ADVANCED][EXTIF] :Extended Interface Settings (Ptnr1) mybrick

        Optional Extended Interface Settings not configured yet!

        Mode                              Bandwidth on Demand active
        Line Utilization Weighting        equal
        Line Utilization Sample (sec)     5


        SAVE                                          CANCEL

Press, to scroll, tag/untag DELETE, to edit
```

The variables from the new MIB table, **pppExtIfTable,** can be identified now on the above Setup Tool page as follows:

- **BodMode** is configured under **Mode**.
- **Algorithm** is configured under **Line Utilization Weighting**.
- **Interval** is configured under **Line Utilization Sample**.

Additionally on this Setup Tool page, **Maximum Number of Dialup Channels** (**PPPMaxConn** of the **biboPPPTable**), corresponds to the number of channels to be dynamically switched.

If you have configured **Mode** to *Bandwidth On Demand active*, *Bandwidth On Demand passive* or *Bandwidth On Demand backup* and then you press **SAVE**, you will return to the previous page, confirm with **OK** and **WAN** ➧ **EDIT** is now supplemented by the WAN Numbers variable, see below:

```
BIANCA/BRICK-XS Setup Tool                    BinTec Communications AG
[WAN] [EDIT]: Configure WAN Partner                            mybrick


        Partner Name                  Partner 1


        Encapsulation                 ppp
        Compression                   none
        Encryption                    none

        PPP >
        Advanced Settings >
        WAN Numbers >

        IP >
        IPX >

                    SAVE                        CANCEL
```

In order for the dialup connection to be dynamically switched to your partner, it is now necessary to enter the WAN number of the partner and direction, either incoming if you have configured *Bandwidth on Demand passive*, outgoing if you have configured Bandwidth on Demand active or both (these settings correspond to the **biboDialDirection** or **biboDialNumber** variables in the **biboDialTable**).

This concludes the configuration of dynamic channel bundling for leased lines over Setup Tool.

Switchover thresholds for BOD

BOD is activated by setting the **pppExtIfBodMode** variable to *BOD-active* or *BOD-passive*, depending on which side should actively switch on and bear the costs. The maximum number of B-channels to be dynamically switched corresponds to the value of the variable **biboPPPMaxConn** of the **biboPPPTable**; this is configured in Setup Tool under **Maximum Number of Dialup Channels**.

- Switching on of B-channels:
  If the **pppExtIfLoad** corresponds to the value 90 (%) or more for at least 5 seconds, a B-channel is switched on.

- Switching off of a B-channel:
  The current value of **pppExtIfLoad** does not serve as the basis for switching off, the calculated (fictitious) bundle load after switching off of a B-channel does. For example, the fifth dialup line is switched off if the remaining four lines in the bundle would have a load of less than 80% for 10 seconds. There are three mechanisms for deciding when a dialup connection is switched off. The first is fixed and is a precondition for the third to take effect, the second and third can be configured separately.
  - If this value drops below 80 (%) for at least 10 seconds, a B-channel is switched off.
  - Static Short Hold: terminates all BOD/backup links after expiry of the inactivity timeout configured. Static Short Hold always takes priority over the load utilization calculation. If, for example, static Short Hold is set to 2 seconds and there is no more data exchange on a channel bundle, the dialup line is terminated after the two seconds and not after the 10 seconds of number 1.
  - Dynamic Short Hold: if the **PPPTable** is correspondingly configured and AOCD (advice of charging during the call) is available, a B-channel is switched off just shortly before the beginning of the next charging unit, provided that the terms of number 1 are fulfilled, i.e. the current capacity utilization is less than 80% for 10 seconds.

Authentication

On establishing a PPP leased line, though accepted, no authentication is required by the partner. Authentication is essential, however, for the dialup link bundled with the leased line and should be configured in the **biboPPPTable** accordingly. Authentication over Setup Tool is set in the menu ***WAN PARTNER*** ➧ ***EDIT*** ➧ ***PPP***. In this case, authentication of the partner is requested for incoming BOD/backup calls.

LCP echo requests (PPP keepalive)

LCP Echo Requests are only generated on existing leased lines, not, however, on the switched B-channels.

X.21 leased lines

The setting for **X.21IfLeads** can have a significant bearing on costs incurred and is thus worthy of some attention. When **IfLeads** is set to *enabled*, a switched backup connection is immediately initiated on the failing of an X.21 leased line. This can, however, lead to excessive and undesired dialup connections if, for example, a flickering leased line sends repeated signals to establish dialup connections. On the other hand, it provides a means of assuring the speedy transmission of data.

By setting **X.21IfLeads** to *disabled*, the backup connection is only established after a period of time set in **biboPPPTimeOut**. This is set to 10 x 3000 milliseconds which is equal to 30 seconds by default. You can thus be sure that due to an unsteady leased line, a series of unwanted dialup connections is not established, and that a backup connection is only made when the leased line has been down for a set time.

A tabular representation of the remaining variables in the biboPPPTable and their relevance for leased with BOD/backup lines.

Variable	Leased with BOD/ BACKUP
Encapsulation	only ppp, x75_ppp, x75btx_ppp
Timeout	Supported
IpAddress	Not supported
RetryTime	Not supported
BlockTime	Supported
MaxRetries	Supported
ShortHold	Supported
InitConn	Not supported
MaxConn	Supported
MinConn	Not supported
CallBack	Not supported
Layer1Protocol	Supported
LoginString	Not supported
VJHeaderComp	Supported
Layer2Mode	Not supported
DynShortHold	Supported

Table 2-5: biboPPPTable

2.4 Filtering of Services in IPX Networks (SAP Filters)

With Release 5.1 Revision 2 for the 1 MB products BIANCA/BRICK XS1 and BIANCA/BRICK XM1, one can filter services in IPX networks with 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 BRICK 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 (described in Software Reference).

2.4.1 The Variables, Values and their Meanings

The following are tabular depictions of the variables, values and meanings of the two new MIB tables **SapDenyTable** and **SapAllowTable**.

Variable	Meaning
sapDenyIfIndexMode	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 BRICK 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 direction will be set to *incoming*.
sapDenyDirection	The direction that is to be subject to the rule. Possible values: *incoming, outgoing, both, dont_verify.*
sapDenyTypeMode	The SAP service type to be checked or not. Possible values: *verify, dont_verify.*
sapDenyType	The various SAP service types to be checked. For example: 4: file server 7: print server.
sapDenyNetMode	The network number to be checked or not. Possible values: *verify, dont_verify.*
sapDenyNet	The service´s network number to be checked.
sapDenyNodeMode	The node number to be checked or not. Possible values: *verify, dont_verify.*

Variable	Meaning
sapDenyNode	The service´s node number to be checked.
sapDenySockMode	The socket number to be checked or not. Possible values: *verify, dont_verify.*
sapDenySock	The service´s socket number to be checked.
sapDenyName	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 BRICK IPX. The values of the Type/Net/Node/Socket fields contained in the **ipxDestServTable** will then be copied to the **sapDenyTable**.

Table 2-6: **SapDenyTable**

Variable	Meaning
sapAllowIfIndexMode	The interface index to be verified or not. Possible values: *verify, dont_verify, delete* Default: *dont_verify*
sapAllowIfIndex	This rule is applied to services originating and/or (see **sapDenyDirection**) destined for the interface with this index number. If, in the case of a service known to the BRICK 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 direction will be set to *incoming*.
sapAllowDirection	The direction that is to be subject to the rule. Possible values: *incoming, outgoing, both, dont_verify.*
sapAllowTypeMode	The SAP service type to be checked or not. Possible values: *verify, dont_verify.*
sapAllowType	The various SAP service types to be checked. For example: 4: file server 7: print server.
sapAllowNetMode	The network number to be checked or not. Possible values: *verify, dont_verify.*
sapAllowNet	The service´s network number to be checked.
sapAllowNodeMode	The node number to be checked or not. Possible values: *verify, dont_verify.*

Variable	Meaning
sapAllowNode	The service´s node number to be checked.
sapAllowSockMode	The socket number to be checked or not. Possible values: *verify, dont_verify.*
sapAllowSock	The service´s socket number to be checked.
sapAllowName	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 BRICK IPX. The values of the Type/Net/Node/Socket fields contained in the **ipxDestServTable** will then be copied to the **sapAllowTable**.

Table 2-7: **SapAllowTable**

2.4.2 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 BRICK 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 BRICK if:

1. it matches an entry in the **sapDenyTable**,
2. 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  direc-
tion=outgoing     typemode=verify     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         172:36:10:62dont_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 service 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
```

Now the service known as FILESERVER will not be propagated over any interface.

- 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.

2.5 VPN and NAT

A typical application of VPN (Virtual Private Networks) with PPTP (Point-to-Point Tunneling Protocol) consists of using the Internet to make a connection to the headquarters. To do this, clients establishing those links use a connection to a local ISP (Internet Service Provider) with dynamic IP address assignment or static IP address. The **BRICK** establishes the VPN tunnel, the ISP is not involved. These constellations are called symmetric PPTP, the **BRICK** combines the functions of PAC (PPTP Access Concentrator) and PNS (PPTP Network Server).

If the client wants to use the connection to the ISP not only for establishing a VPN connection to the headquarters, but also for using other services of the Internet, NAT (Network Address Translation) has to be activated on the client **BRICK**. Then all client PCs connected to the client **BRICK** appear in the Internet with the same IP address.

Up to now BinTec's NAT implementation has supported connections with the protocols ICMP, TCP and UDP, but not with GRE (Generic Routing Encapsulation). All VPN connections using the protocol GRE for transport, e. g. PPTP and L2TP (Layer 2 Tunneling Protocol) could not be switched through a BinTec router when NAT was activated on the WAN interface.

With Release 5.1.2 BinTec's NAT implementation allows forwarding GRE packets to a specified endpoint. So it is possible now to establish VPN connections even with NAT.

This description is based on the example VPN LAN-LAN configuration from the **Extended Feature Reference**, which can be retrieved from BinTec's WWW Server at http://www.bintec.de (Section: Download). In this Release Note the required additional configuration steps, when activating NAT on the WAN interface, are explained. The following points will be covered:

- What could a typical constellation look like?
- What's new with Release 5.1.2?
- What additional configuration steps are required?

2.5.1 Constellation

The following scenario displays a LAN-LAN connection using a VPN tunnel between two partners (CentralSite and SupplierNet):

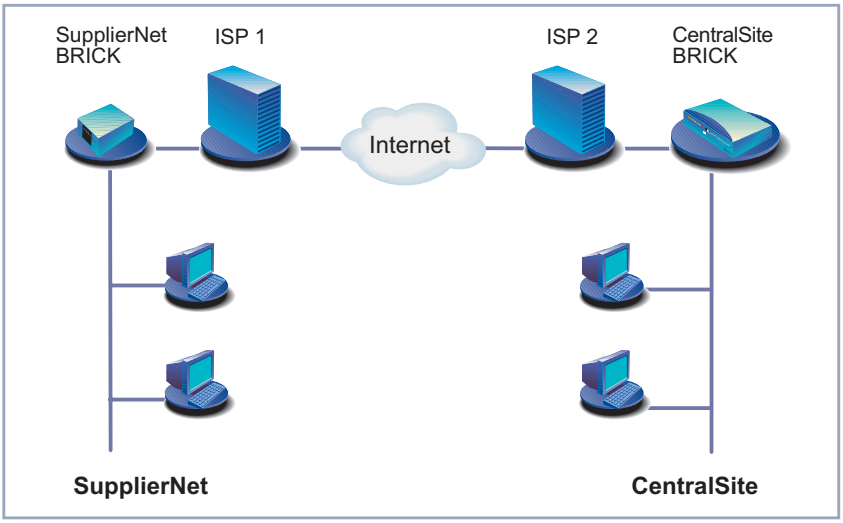

Figure 2-2: LAN-LAN connection using a VPN tunnel

Establishing and using a VPN connection with PPTP (see figure 2-3, page 38 below) requires two protocols between the two tunnel endpoints – TCP (over PPTP) and GRE (over IP):

1. A TCP connection to establish the tunnel (PPTP call control):
   In our example SupplierNet opens the TCP connection to the CentralSite (destination port 1723) to establish a PPTP connection.
2. A GRE session to use the tunnel as a transport medium:
   After establishing the tunnel, SupplierNet can use the GRE session to exchange data with the CentralSite (PPP packets are encapsulated by GRE headers). If NAT is activated on the VPN interface of SupplierNet, the connection can't be realized because GRE packets from CentralSite can't be switched through the NAT firewall of SupplierNet. So the VPN connection fails.

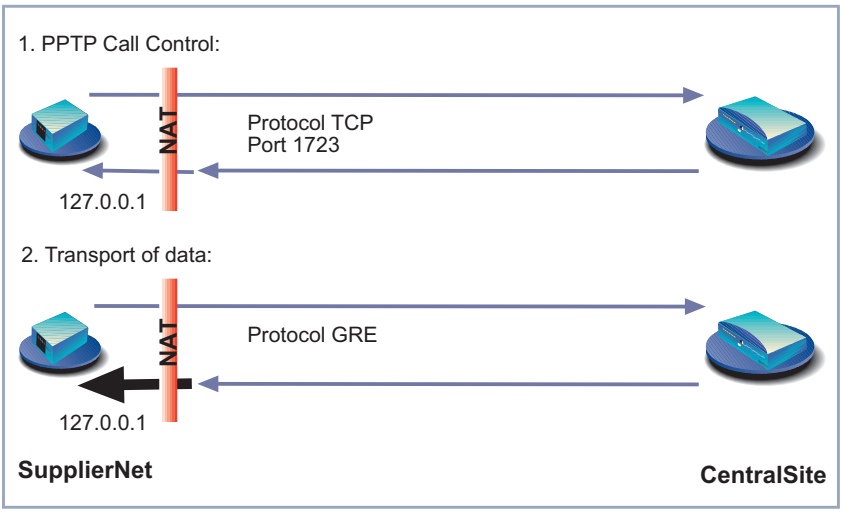

Figure 2-3: Establishing and using a VPN tunnel with PPTP (the changes in Release 5.1.2 allow incoming GRE packets to switch through the NAT firewall)

2.5.2 What's new?

BinTec Communications AG offers the following improvements to enable VPN connections using NAT:

Setup Tool

The range of values **Protocol** can receive in the menu ***IP*** ➧ ***NETWORK ADDRESS TRANSLATION*** ➧ ***RETURN*** ➧ ***ADD*** has been extended with the value *gre* as protocol ID for the protocol GRE. In addition, the values *ah*, *esp* and *l2tp* have been introduced as protocol IDs for Authentication Header, Encapsulated Security Payload and Layer Two Tunneling Protocol respectively.

```
BRICK                                              BinTec Communications AG
[IP][NAT][CONFIG][EDIT]:NAT Configuration (headquarters)          MyXS


    Service                       user defined
    Protocol                      gre
    Port (-1 for any)             -1

    Destination                   127.0.0.1

            SAVE                                  CANCEL


Use <Space> to select
```

Now the following values are available for **Protocol**:

Field	Meaning
Protocol	Protocol to allow. Possible values: *icmp*, *tcp*, *udp*, *esp*, *ah*, *l2tp*, *gre*.

Table 2-8: ***IP*** ➧ ***Network Address Translation*** ➧ ***Return*** ➧ ***ADD***

MIB variables

The range of values of the MIB variables **ipNatProtocol** and **ipNatPrProtocol** in the **IpNatTable** and **IpNatPresetTable** respectively has been extended with the value *gre* as protocol ID for GRE. In addition, the values *ah*, *esp* and *l2tp* have been introduced as protocol IDs for Authentication Header, Encapsulated Security Payload and Layer Two Tunneling Protocol respectively.

The following values are available now:

Variable	Meaning
ipNatProtocol	Specifies the protocol the session is using. Possible values: *udp*, *tcp*, *icmp*, *ospf*, *esp*, *ah*, *l2tp*, *gre*.

Table 2-9: **IpNatTable**

Variable	Meaning
ipNatPrProtocol	Specifies the protocol for which the table entry shall be valid. Possible values: *udp*, *tcp*, *icmp*, *ospf*, *delete*, *esp*, *ah*, *l2tp*, *gre*.

Table 2-10: **IpNatPresetTable**

2.5.3 Configuration

The configuration of NAT for a VPN connection can be done

- via Setup Tool
- via MIB variables

In this Release Note the configuration is described via Setup Tool.

For the configuration of the VPN connection (figure 2-2, page 37), see **Extended Features Reference**.

Client with dynamic IP address assignment

If the client, i. e. SupplierNet, gets its IP address dynamically assigned by its ISP, the establishing of the VPN connection can only be done by SupplierNet, not by CentralSite.

CentralSite has to have a fixed IP address access to the Internet to enable SupplierNet to establish a VPN connection to the CentralSite.

To configure the client **BRICK**, proceed as follows:

- Go to ***IP*** ➧ ***NETWORK ADDRESS TRANSLATION***
- Select the interface to be configured for NAT (i. e. the interface to the ISP) and press **Return**
- Select **Network Address Translation**: *on*
- Press **ADD**
- Select **Service**: *user defined*
- Select **Protocol**: *gre*
- Enter **Port (-1 for any)**: *-1*
- Enter **Destination**: *127.0.0.1*
- Press **SAVE**

Client with static IP address

If the client, i. e. SupplierNet, has a static IP address, the VPN connection can be established by both sites, SupplierNet or CentralSite.

Both CentralSite and SupplierNet have to have a fixed IP address access to the Internet to enable VPN connections to be established in both directions.

To configure the client **BRICK**, proceed as follows:

- Go to ***IP*** ➧ ***NETWORK ADDRESS TRANSLATION***
- Select the interface to be configured for NAT (i. e. the interface to the ISP) and press **Return**

- Select **Network Address Translation**: *on*
- Press **ADD**
- Select **Service**: *user defined*
- Select **Protocol**: *gre*
- Enter **Port (-1 for any)**: *-1*
- Enter **Destination**: *127.0.0.1*
- Press **SAVE**
- Press **ADD**
- Select **Service**: *user defined*
- Select **Protocol**: *tcp*
- Enter **Port (-1 for any)**: *1723*
- Enter **Destination**: *127.0.0.1*
- Press **SAVE**

Central Site with static IP address

To configure the CentralSite **BRICK**, proceed as follows:

- Go to ***IP*** ➧ ***NETWORK ADDRESS TRANSLATION***
- Select the interface to be configured for NAT (i. e. the interface to the ISP) and press **Return**
- Select **Network Address Translation**: *on*
- Press **ADD**
- Select **Service**: *user defined*
- Select **Protocol**: *gre*
- Enter **Port (-1 for any)**: *-1*
- Enter **Destination**: *127.0.0.1*
- Press **SAVE**

- Press **ADD**
- Select **Service**: *user defined*
- Select **Protocol**: *tcp*
- Enter **Port (-1 for any)**: *1723*
- Enter **Destination**: *127.0.0.1*
- Press **SAVE**

127.0.0.1 is the loopback address. It is entered as **IntAddr** or as **Destination** because the **BRICK** itself is an endpoint of the VPN tunnel.

When configuring a VPN connection over a dialup connection it's recommended to set Short Hold for the VPN connection with a shorter time interval than the Short Hold for the underlying dialup connection. Otherwise, unnecessary connections could be established because of termination of the VPN connection.

Testing the configuration

To test the VPN connection, e. g. with the `ping` command:

- do it from a host in your LAN to a host in the partner LAN or
- if you want to test from a **BRICK** to a partner **BRICK**, enter the **BRICK**'s LAN IP address as **Unique Source IP Address** in Setup Tool menu ***IP*** ➧ ***STATIC SETTINGS*** before testing. Otherwise, the **BRICK** will put the IP address of the WAN interface as source address into the ping packets originated by the **BRICK** and the consequence is that outgoing packets to the VPN partner are sent through the tunnel, but can only be returned outside the tunnel (on the underlying WAN connection).

2.6 NetBIOS over NAT

In networks using Windows computers, several network functions such as domain registration, access to drives and printers of other computers are based on the NetBIOS protocol.

WAN connections with NetBIOS over IP

NetBIOS was actually designed for use in LANs. NetBIOS addressing constraints make it impossible to split up a network into subnets with different locations and thereby establishing a hierarchical construction, for example.

In order to avail of the network services mentioned above over WANs, the NetBIOS packets are packed in IP packets, for example, (NBT or NetBIOS over TCP/IP). This can be activated for Windows computers under **Start** ➧ **Settings** ➧ **Control Panel** ➧ **Network** ➧ **Protocols** ➧ **Properties** where TCP/IP is installed.

Once packed in IP packets, NetBIOS packets can also be sent to destinations over routers and WAN connections. In this way, hierarchies are formed.

Heavy traffic loads with NetBIOS over IP

The amount of data traffic between two computers or applications connected by NetBIOS can often be unexpectedly heavy. Frequently, data exchange can take place even when no activity is apparent. For "on demand" WAN connections for which costs are charged, even in local networks as is usual in Europe, the switching of NetBIOS over IP involves an element of risk in terms of costs.

The reduction of traffic

In order to reduce traffic between locations, Microsoft recommends a concept by which a domain controller is used at each location (Windows NT server configured as primary and backup domain controller). This already reduces traffic levels.

In addition, on the support side, Microsoft provides a number of tuning measures (Microsoft Knowledgebase) which help to reduce traffic levels even further. To explain these changes (mostly registry changes), would exceed the scope of this document.

Network construction without an additional domain controller

In many scenarios, e.g. teleworkers or smaller branch offices using their own router, a concept with several domain controllers is inappropriate due to the increased financial and administrative costs involved.

The construction of such networks without the use of domain controllers was already possible using BinTec routers, provided all computers, including those on the central side, had different network addresses.

Establishing a network with NAT

Release 5.1.2 now allows that remote BinTec routers can be configured with NAT for WAN connections with NetBIOS over IP. The BRICK thus independently manipulates not only the IP packets themselves, but also the NetBIOS packets contained within them, enabling domain registration at the central side and access to central computers, printers and drives.

The use of NAT simplifies routing to the central side as each location is represented by just the one IP address. As the illustration shows, packets returning

from the WAN partner are sent to all computers in the LAN with a subnet broadcast address.

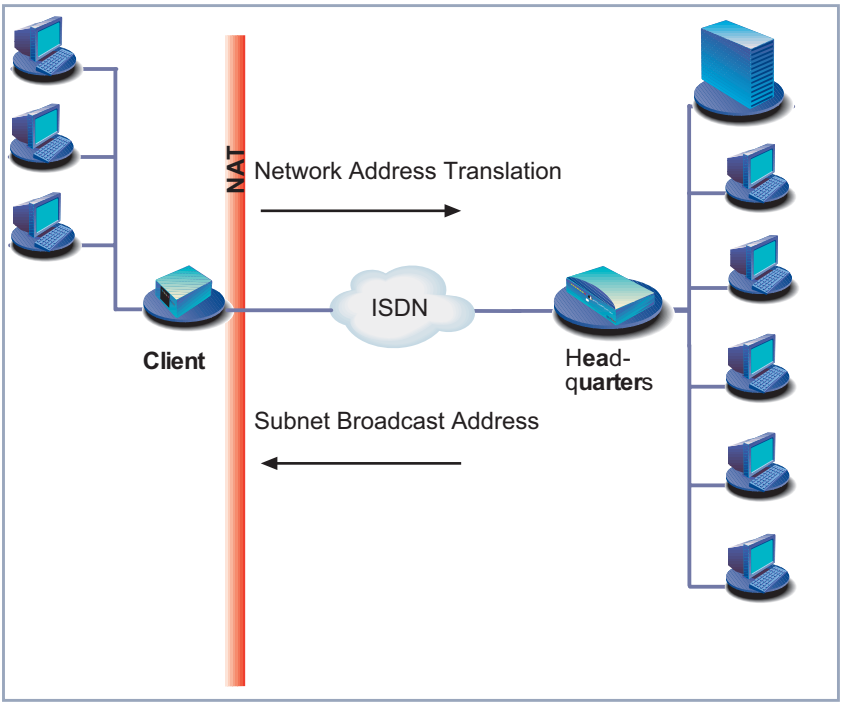

Figure 2-4: NetBIOS over NAT

Prerequisite – Switching Access Lists

If the resources of a central side are to be accessed over WAN links and remote access servers, all routers involved must be configured to route NetBIOS over IP traffic (TCP and UDP packets from and to ports 137 to 139). This is frequently prevented by access lists or packet filters.

Caution!

By the switching of NetBIOS over IP, unexpectedly high connection costs can be incurred on WAN links. The real volume depends on the applications and services used. Consequently, the costs of the connections should be regularly monitored (in the beginning daily). To guard against such unintentional costs, you can avail of the Credits Based Accounting System feature which is available with all BinTec routers and with which connections and costs can be limited.

Activating NAT

In order to activate NAT, proceed as follows:

➤ Go to ***IP*** ➧ ***NETWORK ADDRESS TRANSLATION***

```
BRICK Setup Tool                                BinTec Communications AG
[IP][NAT]: NAT Configuration                                       MyXS

Select IP Interface to be configured for NAT

HQ
en1
en1-snap




EXIT

press <Ctrl-n>, <Ctrl-p> to scroll, <Return> to edit/select
```

➤ Mark the interface or the WAN partner for which you want to activate NAT (e.g. HQ) and press **Return**.

➤ Another menu window opens:

```
BRICK Setup Tool                              BinTec Communications AG
[IP][NAT][CONFIG]: NAT Configuration (HQ)                        MyXS

Network Address Translation                on

Configuration for sessions requested from outside

Service  Destination  Source Dep.  Dest.Dep.  Port Remap




    ADD          DELETE          SAVE          CANCEL

Use <Space> to select
```

Make the following entries:

- Select Network Address Translation: **on**.
- Press **SAVE.**
  Network Address Translation is activated for the selected interface or WAN partner.

Now on routing, all source IP addresses in the LAN are replaced by this address. The NAT address is assigned to the BRICK by IPCP from the WAN partner. The client PC appears on the WINS server on the central side in the WINS databank under this address.

Entering a subnet broadcast address

Essentially, there are three TCP/UDP ports involved in the process of domain registration over NAT: 137, 138 and 139. UDP port 138 is of special interest. Here an additional entry in the **ipNatPresetTable** is essential for NetBIOS over NAT to function properly. The reason is this. According to Microsoft´s implementation, the source and destination ports of these packets destined for the domain controller are always port 138. This means that it is not possible to change and link the source port number with the client PC´s IP address, thus specifying the PC to which the packet should be sent back within the LAN. The domain controller will always return the packet over the port number 138. The

solution to this is simply to enter a subnet broadcast address under **Destination** so that all PCs in the LAN receive the packets. This can also be done over Setup Tool in the menu ***IP*** ➧ ***NETWORK ADDRESS TRANSLATION*** ➧ ***ADD:***

```
BRICKSetup Tool                                BinTec Communications AG
[IP][NAT][CONFIG][ADD]: NAT Configuration                        MyXS

Service                        user defined
Protocol                       udp
Port (-1 for any)              138

Destination                    172.16.98.255 (e.g. a broadcast address)

         SAVE                          CANCEL

Use <Space> to select
```

Finally, actual domain registration and access to the external resources in the **Network Neighbourhood** take place over the NetBIOS session service over port 139. For each of these TCP/IP connections an entry is automatically made in the **ipNatTable.**

Example:

```
inx   IfIndex(*ro)   protocol(*ro   IntAddr(*ro)   IntPort(*ro)
      ExtAddr(ro)    ExtPort(ro)    RemoteAddr(ro) RemotePort(ro)
      Direction(ro)  Age(ro)

00    10001          tcp            172.16.100.99  687
      172.16.200.20  1023           172.16.201.10  139
      outgoing       50
```

Determining a subnet broadcast address

If you want to save yourself the trouble or time of calculating the broadcast address yourself, you can use a subnet calculator tool like the one you can find at this address:

http://www.cci.com/tools/subcalc/index.html

In order to determine the broadcast address, you need to know the IP address of the **BRICK** and the subnet mask. It is then necessary to identify which portion of the IP address is the host number. Each IP address consists of a network por-

tion that identifies the network number and a host portion that identifies the host´s number on that network. The dividing line between the two portions of the address depends on the network "Class" the address belongs to. We can pinpoint this dividing line for all classes, however, by examining the bit values in the netmask as follows:

3. If the bit in the mask is ON (=1), the respective bit in the IP address belongs to the network portion.
4. If the bit in the mask is OFF (=0), the respective bit in the IP address belongs to the host portion.

Example 1

Class B: IP address 128.66.12.1, mask 255.255.255.0, subnet broadcast address 128.66.12.255.

Here is an example of a class B subnet mask. In a standard Class B netmask, the last two bytes identify the host portion of the IP address; in the case of this subnet mask, all eight bits of the third byte define the subnet part of the address, while all eight bits of the fourth byte correspond to the host portion:

In this case, the first three bytes identify the network, including subnets, and the last byte identifies the host. Consequently, the host number to use below is 1.

```
Address    (dec.):128.          66.          12.            1
           (bin.):1000 0000     0100 0010    0000 1100      0000 0001
SubNetmask(dec.):255            255          255            0
           (bin.):1111 1111     1111 1111    1111 1111      0000 0000
```

Example 2

Class C: IP address 192.178.16.66, mask 255.255.255.192, subnet broadcast address 192.178.16.66.127.

Next, an example of a subnet netmask from Class C. In a standard netmask, the last byte indicates the host portion of the IP address; in this case, the first two (or high-order) bits of the fourth byte (11) define the subnet part of the ad-

dress, and the last six bits of that same byte (00 0000) correspond to the host portion.

The host number to use below is 66:

```
Address    (dec.):192.         178.        16.           66
           (bin.):1100 0000    1010 1010   0001 0000     0100 0010
Subnetmask(dec.):255           255         255           192
           (bin.):11111111     11111111    11111111      1100 0000
```

- Convert the host number (IP address) from decimal to binary. To do this, you can use a calculator tool such as the one supplied with Windows NT: Click the Windows **Start** button and point to **Programs** ➧ **Accessories**.
- Convert the corresponding bytes of the subnet mask from decimal to binary.
- Combine the binary subnet mask and the host number as follows:
  If the subnet mask bit is 1 and the host number bit is 0, place a 0 in that position.
  If the subnet mask bit is 1 and the host number bit is 1, place a 1 in that position.
  If the subnet mask bit is 0, place a 1 in that position.
- Convert the resulting binary number back to its decimal equivalent.
- Join the decimal number you have just determined with the network number.

Figure 2-5 is an illustration of the sequence of commands leading to the calculation of a subnet broadcast address outlined above.

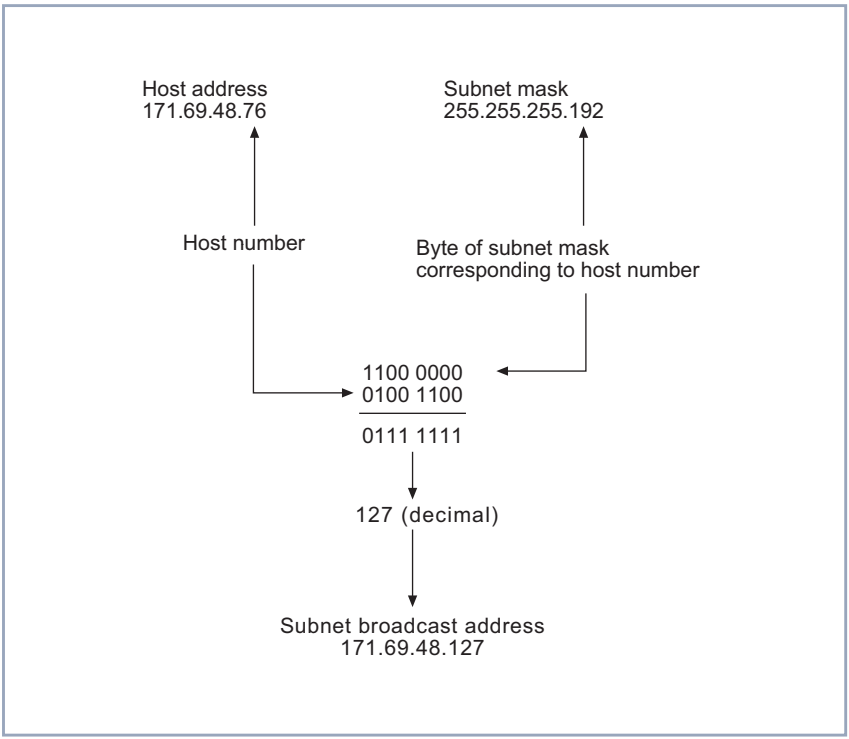

Figure 2-5: Determining a subnet broadcast address

Here are some more examples of IP addresses, standard and subnet masks and their corresponding broadcast addresses from classes A, B and C.

Class	IP Address	Standard / subnet mask	Broadcast Address
A	18.20.16.91	255.0.0.0	18.255.255.255
A (subnet)	18.20.16.91	255.255.0.0	18.20.255.255
B	171.69.48.18	255.255.0.0	171.69.255.255
B (subnet)	171.69.48.18	255.255.255.248	171.69.48.23
C	192.178.16.66	255.255.255.0	192.178.16.255
C (subnet)	192.178.16.66	255.255.255.192	192.178.16.127

Table 2-11: Three classes of IP addresses and broadcast addresses

Bidirectional access

After domain registration has been successfully negotiated, the PDC (Primary Domain Controller) in turn tries periodically to establish a TCP connection with the client over port 139. These connection attempts should normally fail if no additional entry is made in the **ipNatPresetTable**. Thus, only one-way access is configured from the client to the resources of the central network.

You may, however, want to permit bidirectional access (and the costs that might involve). This permission can be given in Setup Tool by adding the **Destination**

of the IP address of the client PC in ***IP*** ➧ ***NETWORK ADDRESS TRANSLATION*** ➧ ***ADD:***

```
BRICKSetup Tool                                BinTec Communications AG
[IP][NAT][CONFIG][ADD]: NAT Configuration                         MyXS


Service                         user defined
Protocol                        tcp
Port (-1 for any)               139



Destination                     172.16.100.99 (IP address of client PC)

Use <Space> to select
```

2.7 NetBIOS Node Type by DHCP

Up to now the only way to define the NetBIOS node type was to set it directly in the registry under Windows 95/98/NT; now, however, configuration (via DHCP) on the BRICK is possible. The **ipDhcpTable** has been extended by the **ipDhcpNodeType** variable. This enables a NetBIOS node type to be set for one IP address pool, making it applicable for all of the DHCP clients.

Methods of name resolution

Basically, the node type defines the way in which Windows has names resolved into their corresponding IP addresses. Each node type contains various methods of name resolution. The following are the methods of name resolution contained in the different node types.

- Broadcast resolution
  This is a means within a LAN by which the owner of a NetBIOS name is requested directly for his IP address.
- WINS (Windows Internet Name Service)
  A server is configured as a NetBIOS databank containing lists of NetBIOS names of clients registered with the server.

- LMHOSTS
  This is a file on the Windows client which contains lists of NetBIOS names and their corresponding IP addresses.

NetBIOS node types

The following node types employ some or all of the above means of name resolution, though arranging and employing them in different sequences. The most appropriate node types for name resolution by a WAN partner are either the P or M-Nodes; default is none or **not specified** in SetupTool.

- B-Node (**Broadcast Node**)

1. Broadcast resolution
2. LMHOSTS or Domain Name Service (DNS) if configured

- P-Node (**Point-to-Point Node**)

1. WINS
2. LMHOSTS or Domain Name Service (DNS) if configured

- M-Node (**Mixed Node**)

1. Broadcast resolution
2. WINS
3. LMHOSTS or Domain Name Service (DNS) if configured

- H-Node (**Hybrid Node**)

1. WINS
2. Broadcast resolution
3. LMHOSTS or Domain Name Service (DNS) if configured

Setup Tool

This setting can also be conveniently configured over Setup Tool in the menu ***IP*** ➧ ***DHCP*** ➧ ***ADD***. The menu looks like this:

```
BRICKSetup Tool                                  BinTec Communications AG
[IP][DHCP][ADD]: Add Range of IP Addresses                          MyXS

Interface                                          en1
IP Address                                         172.16.100.50
Number of consecutive addresses                    50
Lease Time (Minutes)                               300
MAC Address                                        0060B283D02F

NetBT Node Type                                    Point-to-Point Node



                         SAVE                    CANCEL

Use <Space> to select
```

2.8 MS-Callback Termination Option

When callback is configured on a BRICK for a Windows 95/98/NT client, normally the procedure is that you dial in to the central-side BRICK, a window opens in which you enter the dial number of the terminal from which you are presently calling, the original call is disconnected and callback is initiated, i.e. your headquarters calls you back and bears the charges. It is also possible for the administrator at your head office to configure the telephone number from which you regularly require the callback function, in which case you need only confirm the callback mode without entering your telephone number.

If, however, you want to retain the existing connection to your head office without initiating callback, if you do not know the number of the phone you are calling from or you cannot be called back (if the necessary authentication, dialing code or extension are not available, for example), a newly implemented feature to terminate the callback function is available.

Configuration

It is now possible to give the Windows client the option to demand a callback or to access the head office by the initial connection.

➤ Configure the central-side BRICK by setting the **biboPPPCallback** variable to *callback_optional* in the **biboPPPTable**. This has the effect that the following options are now available to the Windows client.

Application

- Windows 95/98 Clients:
  - for callback numbers to be specified by the caller (user-defined number):
    A window appears in which the caller can select either **OK**, to enter a number to be called back or **CANCEL**, to retain the existing connection.
  - for callback numbers predefined by the central-side administrator (administrator-defined number):
    A window appears requesting the user to either confirm the callback mode with **OK**, or to retain the existing connection by clicking **CANCEL.**
- Windows NT clients:
  - for callback numbers to be specified by the caller:
    A window appears in which the caller can select either **OK**, to enter a number to be called back or **CANCEL**, to retain the existing connection.
  - for callback numbers predefined by the central-side administrator:
    In this case, no Window appears and the callback termination option can not be availed of.

2.9 RADIUS for Dial-Out

2.9.1 Introduction

As the name suggests (Remote Authentication Dial-In User Service), RADIUS was designed as a client-server system for authenticating dial-in connections. The BRICK can be configured to operate as a RADIUS client that consults the RADIUS server at connection time for the authentication and identification of specified dial-in partners.

It is now possible, however, for the BRICK to request user data from the server in order to establish a connection also for outgoing calls.

Why RADIUS for dial-out

The principal objectives that lay behind the implementation of this new feature are two-fold:

- Firstly, in view of the fact that at most 500 WAN partners can be configured on the BRICK and some installations can greatly exceed this figure, this feature provides an alternative to configuring WAN partners on the router. The entries for WAN partners are no longer made locally on the BRICK via Setup Tool, but now on the RADIUS server. There can thus be considerable savings in terms of Flash memory.
- Secondly, RADIUS for dial-out is easier to manage in terms of configuration. The many entries over Setup Tool are replaced by the more convenient administration of the RADIUS server over the usual editor tools.

How does it work?

Firstly, the BRICK is configured to request data from the server. Then all the information required for the establishment of a PPP connection is configured in the database of the RADIUS server.

The BRICK firstly requests all the routing information from the RADIUS server and stores it in the **ipRouteTable**. Loading of this initial information is driven over the **RadiusSrvDialout** variable. On the one hand, the variable can be set to *enabled* and then saved with the configuration so that initial loading occurs immediately after every reboot. Alternatively, by setting to *reload*, it is possible to load or reload the routing information at any time you choose.

When a dial-out call to a WAN partner is to be made on one of these loaded routes, another request is sent to the RADIUS server in order to receive the necessary partner-specific information (e.g. data for the authentication, encapsulation, extension number etc.), each partner can have more than just one entry in the **ipRouteTable**. If the partner is configured on the server, the necessary information entries are transferred to the BRICK and generated in the respective MIB tables for the duration of the call.

After the end of the call, all entries on the BRICK´s MIB tables are deleted with the exception of the routing information in the **ipRouteTable**, which is loaded initially.

Configuration

Configuration of RADIUS for dial-out takes place on two levels:

- Configuration on the BRICK side: entries are made over the **RadiusServerTable**.
- Configuration on the RADIUS server: configuration on the "users" file in the corresponding directory on the RADIUS server.

2.9.2 Configuration on the BRICK

Configuration on the BRICK is made over the **RadiusServerTable**. The following basic settings are required for dial-out and important for the initiation of the

dialog between BRICK and server so that the necessary routing and partner-specific information is transferred:

Variable	Meaning
RadiusSrvProtocol	In order to configure RADIUS authentication, leave the value *authentication*.
RadiusSrvAddress	The IP address of the RADIUS server.
RadiusSrvPort	RFC 2138 assigns port 1812 and 1813 for authentication and accounting respectively. Many RADIUS servers, including Merit, still use 1645 and 1646. As RADIUS servers use different port numbers, you should refer to the documentation for the RADIUS server you are using.
RadiusSrvSecret	This is a shared secret between the RADIUS server and the BRICK.
RadiusSrvDialout	This option provides the means for RADIUS dial-out configuration. The possible values are *enabled*, *disabled* or *reload.*
RadiusSrvDefaultPW	Is not required with certain RADIUS implementations, such as Merit, here again you should consult the documentation for your RADIUS server. Some RADIUS servers rely on a configured user or CHAP password for any RADIUS request.

Table 2-12: **RadiusServerTable**

Here are examples of three different entries in the **RadiusServerTable**:

```
inx Protocol(*rw)    Address(rw)    Port(rw)       Secret(rw)
    Priority(rw)     Timeout(rw)    Retries(rw)    State(-rw)
    Policy(rw)       Validate(rw)   Dialout(rw)    DefaultPW(rw)

00  authentication   172.16.70.14   1645           secret
    0                1000           5              active
    authoritative    enabled        enabled

01  authentication   172.16.70.93   1645           secret
    1                1000           5              active
    authoritative    enabled        enabled

02 accounting        172.16.70.98   1646           secret
    2                1000           5              active
    authoritative    enabled        enabled
```

What happens on the BRICK?

According to the example above, once a dial-out request is made that is to be sent on one of the routes loaded from the RADIUS server and thus occupying an **IfIndex** above 30000, the RADIUS server with the IP address 172.16.70.14 receives a request, as this entry has the lowest **RadiusServerPriority** setting.

Backup

If this server does not reply after 5 attempts (**RadiusSrvRetries**), each after an interval (**RadiusSrvTimeout**) of 1000 seconds, the **RadiusSrvState** is set to *inactive*. The server with the next lowest priority setting, in this case the server with the IP address *172.16.70.93*, then receives a request from the BRICK. If this server responds to the BRICK request, the partner-specific information for an outgoing call to a WAN partner can be loaded from the RADIUS server to the corresponding MIB tables on the BRICK.

2.9.3 Configuration on the RADIUS server

(The following description is taken from the RADIUS implementation for Merit.) The second part of the configuration deals with telling the "users" file on the RA-

DIUS server firstly the routing information required for the WAN partner and secondly the partner-specific information assigned to each routing entry. A significant advantage of the implementation from BinTec Communications AG is that only one entry in the "users" file is sufficient to enable both dial-in as well as dial-out.

Finally, it is necessary that the BinTec-specific extensions are included in the dictionary file of the server.

IP routing

The first part deals with defining the necessary IP routing information. Here the following syntax should be obeyed:

```
Framed-Route = destaddr/mask gateway userid userpw private metric
```

Routing info	Meaning
`destaddr/mask`	Destination address with netmask, required for the dial-out request.
`gateway`	Gateway address (nexthop) (optional).
`userid`	The partner´s user ID, necessary for the dial-out request.
`userpw`	This password must match the password attribute in the "users file", see Examples of Framed-Routes and their corresponding partner-specific entries in the "users file", page 70 below. You need only make the one entry for both dial-in and dial-out. It may not consist only of digits.
`private`	Selection of the routing protocols, RIP, OSPF, or the PROXYARP used for the propagation of this IP route (optional).
`metric1 - 5`	Sets the variables metric1 to metric5 in the **ipRouteTable**; metric1 should always lie above the value for the dial-in case, i.e. the worse metric. If no metric is given, metric1 is set to 5, while metric2 to metric5 are set to 0 (optional).

Table 2-13: "users file" of the RADIUS server

If only dial-out (without Callback) is being configured, `destaddr` and `userid` are the minimum entries required.

Several of these "routes" are then compiled and arranged under a fictitious user "dialout-X", which begins with the number 1. The number of entries under one of these dummy-users is restricted to the UDP limit of 4096 bytes. Whereby the optimum numbers in terms of loading times and system utilization lie at around 20-40 entries inside the "Framed-Route" record. On initial loading, the BRICK

asks for a user by the name of **dialout1**, then **dialout 2** and so on. Here is an example of what a dummy user could look like:

```
dialout-1
   Framed-Route = "1.2.1.1 user1 secret1 3",
   Framed-Route = "1.2.1.2 user2 secret2 3",
   Framed-Route = "1.2.2.0/24 network1 secret3 3",
   Framed-Route = "1.2.1.3 user3 secret OSPF 5",
   Framed-Route = "1.2.1.4 user4 secret OSPF 5",
   Framed-Route = "1.2.1.5 user5 more_secret RIP 5",
   Framed-Route = "1.2.1.6 user6 secret6 RIP6",
   Framed-Route = "1.2.1.7 user7 secret7 RIP 7",
   Framed-Route = "1.2.1.8 user8 secret8 RIP8",
   Framed-Route = "1.2.1.9 user9 secret9 RIP9",
   Framed-Route = "1.3.1.0/24 network10 passwdnetwork10 10",
   Framed-Route = "1.3.2.0/24 network11 passwdnetwork11 11",
   Framed-Route = "1.3.3.0/24 network12 passwdnetwork12 12",
   Framed-Route = "1.3.4.0/24 network13 passwdnetwork13 13",
   Framed-Route = "1.3.5.0/24 network14 passwdnetwork14 14",
   Framed-Route = "1.4.6.0/24 network15 passwdnetwork15 OSPF 15",
   Framed-Route = "1.4.2.0/24 network16 passwdnetwork16 OSPF 16",
   Framed-Route = "1.4.2.0/24 network17 passwdnetwork17 OSPF 17",
   Framed-Route = "1.4.2.0/24 network18 passwdnetwork18 18",
   Framed-Route = "1.4.2.0/24 network19 passwdnetwork19 RIP 19",
   Framed-Route = "1.5.1.0/24 network20 passwdnetwork20 RIP 20",

dialout-2
   ..............................
   ..............................
dialout-3
   ..............................
   ..............................
```

Specifying the BRICK to which the IP routing information should go

In the event that you have more than just one BRICK to which your IP routing information is to be transferred, it is possible to differentiate between the routers using the following syntax for the aforementioned dummy user, the following **sysName** variable is from the MIB II table **system**:

```
dialout-[sysName]-x
dialout-brick1-1
   ..............................
   ..............................
dialout-brick1-2
   ..............................
   ..............................
dialout-brick1-3
   ..............................
   ..............................
dialout-brick2-1
   ..............................
   ..............................
dialout-brick2-1
   ..............................
   ..............................
dialout-brick2-3
   ..............................
   ..............................
```

When the IP routing information is loaded to the BRICK (usually on booting when **RadiusSrvDialout** is set to *enabled*), the information is stored in the **ipRouteTable.** The indices for the as yet unused interfaces for these route entries extend from 30000. The **ipRouteTable** could look something like this:

```
inx   Dest(*rw)      Ifindex(rw)    Metric1(rw)    Metric2(rw)
      Metric3(rw)    Metric4(rw)    NextHop(rw)    Type(-rw)
      Proto(ro)      Age(rw)        Mask(rw)       Metric5(rw)
      Info

03    1.2.1.1        30001          3                0
      0              1              0.0.0.0          indirect
      other          1538           255.255.255.25   0
      .0.0
04    1.2.1.2        30002          3                0
      0              3              0.0.0.0          indirect
      other          1540           255.255.255.255  0
      .0.0
05    1.2.2.0        30003          3                0
      0              3              0.0.0.0          indirect
      other          1540           255.255.255.255  0
      .0.0
```

Propagating dial-out IP routes via RIP

Example

```
dialout-1
   Framed-Route = destaddr/mask userid userpw RIP
```

For settings made in the **ipExtIfTable** on the BRICK and derived from the BinTec-specific RADIUS attributes, the variable **RouteAnnounce** is rendered ineffectual.

In principle, this is possible but not advisable if there is a large number of IP routes.

Propagating dial-out IP routes via OSPF

Example

```
dialout-1
   Framed-Route = destaddr/mask userid userpw OSPF
```

For settings made in the **ipExtIfTable** on the BRICK and derived from the BinTec-specific RADIUS attributes, the variables **Ospf**, **RouteAnnounce** and

OspfMetric are rendered ineffectual. The dial-out IP routes are thus propagated with an OSPF metric calculated as follows:

IpMetric1 + 20

Dial-out IP routes and Proxy-ARP

Example:

```
dialout-1
   Framed-Route = destaddr/mask userid userpw PROXYARP
```

For settings made in the **ipExtIfTable** on the BRICK and derived from the BinTec-specific RADIUS attributes, the variable **ProxyArp** is rendered ineffectual.

Entries in the "users" file

Now it is necessary to assign specific details about the partner to each IP route entry, again in the "users file" of the RADIUS server. Here it is possible that sev-

eral routes refer to just the one user entry. The minimum configuration entries must include the following:

Attribute	Meaning
`Service-Type = Framed`	This is the default value for PPP connections
`Framed-Protocol = PPP`	This is the type of encapsulation used. If not set PPP is used
`Framed-IP-Address = X . X . X . X`	This is the IP address of the WAN partner and must correspond to the destination address in the routing entry described above in Framed-Route
`Framed-IP-Netmask = Y. Y . Y . Y`	IP netmask, it could be something like *255.255.255.255*
`BinTec-biboDialTable = "direction=outgoing number=********"`	A temporary entry for dial-out, including the phone number of the WAN partner, is made in the **biboDialTable**
`Password`	A user password that must match the password used in the Framed-Route (see **userpw** above).

Table 2-14: "users file" of the RADIUS server

Caution!

- Setting **direction** to *outgoing* is important for security reasons so that no incoming calls are authenticated over this entry.

The following attributes are optional:

Attribute	Meaning
`Framed-MTU`	Sets the **ifMtu** variable in the **IfTable**
`Framed-Compression`	Sets the **VJHeaderComp** variable of the **PPPTable** if necessary
`Idle-Timeout`	Sets the **ShortHold** variable in the **PPPTable**
`Port-Limit`	Sets the **MaxConn** variable in the **PPPTable**
`BinTec-biboPPPTable = "biboPPPAuthentica-tion=pap/chap/ ms_chap"`	The authentication protocol used for dial-out; if this is not explicitly specified, the authentication protocol CHAP is set.
`BinTec-biboPPPTable = "biboPPPLocalI-dent=local_pppid"`	This is the local ppp ID for authentication at the WAN partner´s (optional) and the default setting

Table 2-15: Attributes in the "users file" of the RADIUS server

Examples of Framed-Routes and their corresponding partner-specific entries in the "users file"

Example 1:

```
dialout-1
   Framed-Route = "1.2.1.1 user1 topsecret3"

user1
   Password = topsecret3,
   Service-Type = Framed,
   Framed-Protocol = PPP,
   Framed-IP-Address = 1.2.1.1.
   Framed-IP-Netmask = 255.255.255.255,
   Idle-Timeout = 25,
   BinTec-biboPPPTable = "biboPPPAuthentication=chap",
   BinTec-biboPPPTable = "biboPPPLocalIdent=mylocalid",
   BinTec-biboDialTable = "direction=outgoing number=00815123456"
```

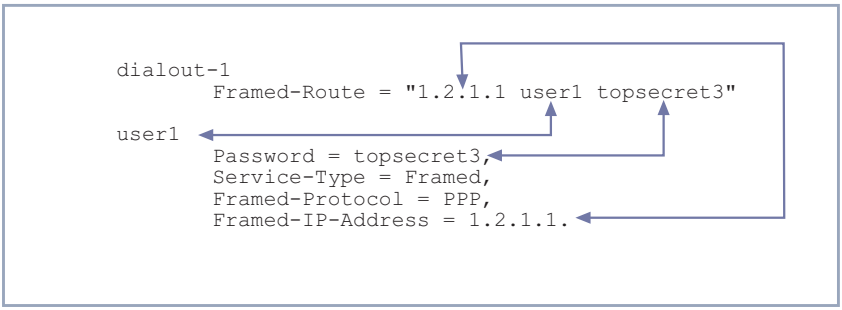

Figure 2-6: Matching Framed-Routes and partner-specific entries

Example 2:

```
dialout-1
   Framed-Route = "1.2.1.2 user2 mysecret2 3"

user2
   Password = "mysecret2",
   Service-Type = Framed,
   Framed-Protocol = PPP,
   Framed-IP-Address = 1.2.1.2.
   Framed-IP-Netmask = 255.255.255.255,
   Idle-Timeout = 30,
   BinTec-biboPPPTable = "biboPPPAuthentication=chap",
   BinTec-biboPPPTable = "biboPPPLocalIdent=mylocalid",
   BinTec-biboDialTable = "direction=outgoing number=00815123456"
```

For the purpose of clarification, the above examples place the Framed-Routes together with the partner-specific information. As shown above, user name, IP address and password are identical in the routing and partner-specific information included in each example. This is essential for RADIUS for dialout to function properly.

BinTec-specific dictionary

This dictionary is also a part of and can be retrieved from the RADIUS server directory, often the dictionary needs to be supplemented. Among information for other systems, it must contain BinTec-specific information, for example concerning the BinTec MIB tables. The inclusion in the dictionary of the attributes of these tables is then essential for the information written to the "users file" for both dial-in as well as dial-out and to be used by the RADIUS server. The following is a list of BinTec extensions for Merit:

BinTec extensions

```
VALUE       Framed-Protocol      IP-LAPB             17825796
VALUE       Framed-Protocol      IP-HDLC             17825797
VALUE       Framed-Protocol      MPR-LAPB            17825798
VALUE       Framed-Protocol      MPR-HDLC            17825799
VALUE       Framed-Protocol      FRAME-RELAY         17825800
VALUE       Framed-Protocol      X75-PPP             17825802
VALUE       Framed-Protocol      X75BTX-PPP          17825803

BinTec.attr BinTec-biboPPPTable        224 string(*, 0, NOENCAPS)
BinTec.attr BinTec-biboDialTable       225 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ipExtIfTable        226 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ipRouteTable        227 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ipExtRtTable        228 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ipNatPresetTable    229 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ipxCircTable        230 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ripCircTable        231 string(*, 0, NOENCAPS)
BinTec.attr BinTec-sapCircTable        232 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ipxStaticRouteTable 233 string(*, 0, NOENCAPS)
BinTec.attr BinTec-ipxStaticServTable  234 string(*, 0, NOENCAPS)
```

3 Changes/Improvements

3.1 PPP

3.1.1 Inconsistent Encryption Configuration Leading to Repeated Connection Attempts

If a dialout partner tries to negotiate MPPE, but the encryption is not enabled by the WAN partner, the connection is correctly terminated and the state of the interface of the client turns to *blocked*.

Problems occurred in the reverse case, however, where the dialout partner did not enable MPPE encryption, although the WAN partner required it. The connection was terminated by the WAN partner. The state of the client interface, however, did not turn to *blocked*, as it did not know the reason for the failure to connect. Continued attempts to establish the connection from client to server were made.

Now in the latter case, the dialout client partner not supporting encryption receives notification of an encryption requirement, causing the state of the client interface to turn to the *blocked* state. Thus, the repeated connection attempts are prevented.

This proprietary solution can only function provided both sides are BinTec routers and both sides are running Release 5.1.2 or greater.

3.1.2 Permanent Connection

The **PPPShortHold** variable in the **biboPPPTable** is the time in seconds which must elapse after no further data exchange occurs before the link is terminated.

By setting PPPShortHold to *-1,* however, it is now possible to set this variable in a way in which after termination of the link, a dial-up connection is automatically initiated and the link is reestablished. The current operational status of the

interface, **ifOperStatus**, only takes the values *up* or *down*, no longer *dormant* or *blocked*. Configuration of this feature can only be made in the MIB table and not in Setup Tool.

Caution!

This immediate reestablishment of the link should be expressly wished as setting **PPPShorthold** to *-1* can obviously have considerable financial implications.

➤ If you wish to prevent constant reestablishment of a link, make sure to set **PPPShorthold** to a value other than *-1.*

3.2 IP

3.2.1 No Longer Automatic IP Conversion

If you are updating from a software release equal to or older than 4.7.x to the current version, 5.1.2 or later, it is necessary to firstly upgrade to 4.9.3, save the configuration (cmd = save), then upgrade to 5.1.2. If you update directly from 4.7.x to 5.1.x, the old access lists (**ipAllowTable**, **ipDenyTable**) can not be automatically converted and are lost.

3.2.2 Transit Network Settings

A new field has been added to the ***WAN*** ➧***EDIT*** ➧***IP*** menu in Setup Tool. If you are not using a transit network (and **no** is selected after **IP Transit Network**), it is nevertheless possible to enter a local IP address: the new field **local IP Address** appears. This field does not appear if either **yes**, **dynamic client** or **dynamic server** is selected after **IP Transit Network**. An entry is then required in the new field if:

1. there are other WAN partners with transit networks
2. there are several Ethernet modules

3. there is another IP configuration for which the BRICK has several different IP addresses.

3.2.3 Local IP Address in IPCP Negotiation

If the local IP address variable in ***STATIC SETTINGS*** ➧ ***UNIQUE SOURCE IP ADDRESS*** (**biboAdmIpAddr**) is set, its value is used as the source address of all IP packets and transmitted to the WAN partner in the course of the IPCP negotiation. When a transit network is configured to the WAN partner, however, the transmission of the local IP address prevents a PPP connection from being made.

From this release, the ***UNIQUE SOURCE IP ADDRESS*** variable (**biboAdmIpAddr**) will only be used for IPCP negotiations if no corresponding entry has been made in the **ipRouteTable** for the interface over which the IP packet is to be transmitted, or if there are no entries in the **ipAddrTable**.

3.2.4 NAT

1. The aging interval for entries to the TCP port 1723 (PPTP) has been increased to 24 hours. This value is fixed and can no longer be adjusted.
2. The limitation of NAT sessions has increased from 230 to 4000.

```
BIANCA/BRICK-XS Setup Tool                    BinTec Communications AG
[WAN][EDIT][IP]:IP Configuration(MYBOSS)                          MYXS

IP Transit Network                            no

Local IP Address

Partner´s LAN IP Address
Partner´s LAN Netmask
Advanced Settings

            SAVE                          CANCEL

Enter IP address (a.b.c.d or resolvable hostname)
```

3.3 RADIUS

3.3.1 Additional RADIUS Attributes now Supported

The following RADIUS Attributes are being added , if available, and transferred on Authentication (Access-Request) or on Accounting Start/Stop.

RADIUS Attribute	Meaning
CLASS (Authentication (Access-Request) and Accounting)	This Attribute can be used for the synchronisation of RADIUS authentication and accounting. It is useful for a Steel-Belted RADIUS Server or another RADIUS implementation, for example, that has dynamically assigned an IP address from its pool. With the help of this Attribute, the corresponding IP address can be returned to the pool for the Accounting-Stop record. The RADIUS server can set the class attribute with the authentication response. The BRICK will return it with the accounting start and stop record.

RADIUS Attribute	Meaning
`CALLED-STATION-ID` (Authentication only (Access-Request))	The Called Party Number is useful, for example, to allow certain user groups access to certain services.
`CALLING-STATION-ID` (Authentication only (Access-Request))	The Calling Party Number gives additional authentication support.
`FRAMED-IP-ADDRESS` (Accounting)	This is the IP address assigned to the partner after authentication and enables a simpler arrangement of accounting data.

Table 3-1: RADIUS server attributes

3.4 CAPI

3.4.1 CAPI Info Syslog Message

Applications that started 30 CAPI listener(s) and received an incoming call also received very many Syslog messages, creating a considerable load. For this reason, the classification of the Syslog message "setup ignored" has been changed from "info" to "debug".

3.4.2 New CAPI Variables

Three new variables have been added to the **capiConfigTable.**

1. **Fax12000(rw)**: This enables or disables the 12000bps mode for fax transmission. If the value of the variable is set to *on*, the fax speed will fall back from 14400 to 12000 bps during a retrain. If set to *off*, it will fall straight back to 9600 bps. The default value is *off*.

2. **FaxTXLevel(rw)**: The transmission level can be set to -x dB db0 = 0dB, db3 = -3dB. The default value and the value normally used for fax transmission in Germany is -6dB (db6).
3. **FaxModulation(rw)**: With this variable you can set the following default transmission protocols for fax. The default value is v17.
   – v17 max. 14400 bps new standard
   – v33 max. 14000 bps early standard
   – v29 max. 9600 bps fax standard
   – v17s: v17 with extended fax-on-demand capability
   – v33s: v33 with extended fax-on-demand capability

3.4.3 Error Correction Mode for G3 Faxing = On

The default value for **capiConfigFaxG3ECM** is now *on.* This specifies whether ECM (Error Correction Mode) should be used for the T30 protocol in G3 facsimile transmissions.

3.5 XBRI

3.5.1 XM as Fax Server on 2XBRI Connections

4 channels can be used over a CM 2XBRI with modem functionality installed in a BRICK XM. With the maximum number of two installed CM 2XBRIs, the number of channels has increased to eight.

3.6 System

3.6.1 No Autologout during Update

In the past, it may have been necessary to disable autologout for the course of an update or to reset it to a higher value if a rather low time interval was set.

Now, however, autologout can not interrupt the installation of an image even if the autologout time interval is less than the time it takes to update.

Regardless of the value for autologout prior to the update, once the update is complete, autologout is reset to its default value of 900 or 15 mins. If, however, you want to change back to the value set prior to the update, type `t` in the SNMP shell and your old value will be reset.

3.6.2 Trace Application Modification

The output of the trace application has been slightly improved. The facilities traced in the D-channel can now be directly compared to their respective ASN.1 specifications.

3.6.3 Setup Tool *SYSTEM* Menu

With the implementation of the new Windows Activity Monitor feature, the Setup Tool ***SYSTEM*** menu has changed. This new feature is a surveillance tool for Windows users to oversee the functions of a **BRICK**, and is described above in Set-up Tool, page 12.

The following changes have been made:

- The ***SYSTEM*** ➧ ***EXTERNAL ACTIVITY MONITOR*** submenu has been added.
- The password settings are made in the new submenu ***SYSTEM*** ➧ ***PASSWORD SETTINGS***.

The ***SYSTEM*** menu now appears as shown in the example below.

```
BRICK Setup Tool                              BinTec Communications AG
[System]: Change System Parameters                                MyXS

System Name                              brick
Local PPP ID (default)                   bricklocal
Location                                 Country
Contact                                  BinTec


Syslog output on serial console          no
Message level for the syslog table       info
Maximum Number of Syslog Entries         20


External Activity Monitor >
External System Logging >
Password Settings >


            SAVE                                  CANCEL

Enter string, max length = 34 chars
```

4 Bug Fixes

4.1 Setup Tool

4.1.1 SetupTool Crash

If, after getting and loading a configuration file by TFTP that has been initially created with the Configuration Wizard, a Setup Tool crash may have occurred on leaving the ***PPP*** ➧ ***PPP PROFILE CONFIGURATION*** menu via ***SAVE***:

```
BRICK-XS Setup Tool                          BinTec Communications AG
[PPP] PPP Profile Configuration                                  MyXS


        Authentication Protocol          CHAP+PAP+MS-CHAP
        Radius Server Authentication     none

        PPP Link Quality Monitoring      no


                SAVE                          CANCEL

Use <Space> to select
```

This bug has been fixed.

4.1.2 CLID Configuration in Setup Tool Flawed

If two different numbers were configured for the one WAN partner with the **Directions both** and **outgoing** and if **LAPB Framing (only IP)** was selected under **Encapsulation**, the menu could not be exited via **SAVE**. Despite the fact that **Calling Line Identification** was set to **Yes**, the error message **"You must use Calling Line Identification (Direction both or incoming)"** was displayed.

This bug has been fixed.

4.2 CAPI

4.2.1 BRICK XM Fax Application

When sending / receiving simultaneously several faxes via the CM-2XBRI ISDN module, it could have occurred that the system rebooted.

This bug has been fixed.

4.2.2 CAPI via X.31 on D-Channel

With an already existing CAPI connection via X.31 on the D-channel while other ISDN links were established and terminated, it could have occurred that CAPI messages which had to establish the B-channel connection were delivered to the X.31 application. This could have led to the termination of the links.

This bug has been fixed.

4.2.3 Connection Delays with CAPI

When there were particularly heavy loads of data traffic, transmission on the TCP connection may have been delayed by up to 15 seconds.

This bug has been fixed.

4.2.4 Data_B3_IND Data Handle Counted up to 2

Since Release 4.9.3 a valid Data Handle has been assigned. This was, however, counted up to 2 instead of 1.

This bug has been fixed.

4.2.5 CAPI and Incorrect Bearer Capability

Incoming CAPI calls, e.g. GSM calls, with incorrect bearer capability were not signalled to the CAPI application. When the bearer capability contained additional bytes to that contained in the CAPI specifications, no CIP value was recognised.

This bug has been fixed.

4.2.6 Connection Difficulties with CAPI 1.1 Applications

In some cases, when using CAPI 1.1 applications with Releases 4.9.1 and 4.9.3, outgoing connections could not be properly established.

This bug has been fixed.

4.2.7 Transmission Failure after Protocol Switching

There are a few fax and voice applications that use the Select_B_Protocol_REQ message to switch between protocols for fax and voice. Data could not be transmitted after this kind of switching.

This bug has been fixed.

4.3 ISDN

4.3.1 Problems with Autoconfiguration

The autoconfiguration of the ISDN switch type (**autodetect on bootup** in Setup Tool) failed to function properly with Release 4.9.5, and with PRI.

These bugs have been fixed.

4.3.2 Leased Line of Channel 31 Ineffective on Booting

When a leased line bundle, for example, was configured for channels 30 and 31, and the BRICK (XL) was started, only channel 30 was activated. Channel 31 was not activated, but set to dialup in the **isdnChTable**. If channel 31 was then manually set to **Leased Line DTE**, **different endpoints**, in Setup Tool, the bundle functioned properly. However, on every reboot of the BRICK, the channel was reset to **dialup** again.

This bug has been fixed.

4.3.3 Connected Address with 1TR6 Leading to Connection Failure

In very few applications, if the presentation indicator was set in the connected address and the 1TR6 Protocol was being used, the connection could not be made. The 1TR6 does not allow the forwarding of the presentation indicator in the connected address in the D-channel.

Now, however, the connected address is filtered out by the BRICK and the call can be made.

4.4 System

4.4.1 Multiple Simultaneous Logins Jam BRICK

When a lot of people logged in over isdnlogin, it occurred that after a period of time no further logins were possible, and that no further processes could be started on the BRICK.

This bug has been fixed.

4.4.2 Number of Columns in the SNMP Shell

When the number of columns (i.e. each vertical line of character entries) was set to a value which fell below the number of characters required by the longest string using the command u, the SNMP shell crashed.

This bug has been fixed.

4.5 PPP

4.5.1 Channel Bundling and Dynamic Short Hold Malfunction

The channels of a bundle were disconnected at the end of each charging unit and then immediately reconnected, provided the load was high enough. This malfunction did not result in increased costs as the charging units were exploited to the full; there may, however, have been a momentary collapse in bandwidth.

This bug has been fixed.

4.5.2 MS-Callback (CBCP) Working on the 2nd Attempt

The MS-Callback Control Protocol initialized from a Windows client to the **BRICK** was known not to work under a combination of the following conditions:

– on the first attempt after a reboot of the **BRICK**

– no entry in the **biboDialTable**, which implies the "User Defined Numbers" mode.

This bug has been fixed.

4.5.3 Charging Amounts Logged Incorrectly

If the variables **biboPPPTotalCharge** and **biboPPPConnCharge** were set correctly and no value was given to the **biboPPPLinkCharge**, incorrect charging amounts in the syslog message (INFO-Level) may have occurred. If the problem occurred, the amount logged was higher than the real value.

This bug has been fixed.

4.5.4 Credits Based Accounting: Connections not Terminated

The settings **MaxOutDuration** and **MaxInduration** in ***ISDN*** ➧ ***CREDITS*** ➧ ***EDIT*** are designed to limit the total length of all incoming/outgoing calls. Although no further connections could be made, existing connections were not terminated after reaching the values set for these variables.

This bug has been fixed.

4.5.5 Problems Accessing Compuserve for the First Time

If the **BRICK** was newly configured, used as DHCP server on the LAN and you had configured Compuserve as your ISP combined with encapsulation x.75_PPP or x.75_BTX_PPP, it was not possible to use a browser to establish a connection with your provider for the very first time.

In such a case, the router could not find a DNS server, necessary for connections using a name-based browser. Only connections to partners with PPP encapsulation were made and not to partners with x.75_PPP or x.75_BTX_PPP, as in the case of, for example, Compuserve access.

This bug has been fixed.

4.5.6 Termination of VPN Connections

If the **BRICK** is used to create a connection over a VPN tunnel to a PPTP-PNS (PPTP Network Server) via a PPTP-PAC (PPTP Access Concentrator), the **BRICK** typically has to perform two PPP authentication negotiations:

- an initial PPP negotiation with the PPTP-PAC
- a final PPP negotiation with the PPTP-PNS

If both PPP negotiations are successfully performed, a VPN connection between **BRICK** and PPTP-PNS is established.

It could have occurred that the **BRICK** terminated the link to PPTP-PAC and therewith the VPN connection to the PPTP-PNS. The following actions were performed:

1. The **BRICK** started an initial PPP negotiation with the PPTP-PAC.
2. If thus negotiated, the **BRICK** started PPP authentication (for example by using PAP) with the PPTP-PAC sending a PAP authentication request.
3. The PPTP-PAC ignored this PAP authentication request, but established a VPN tunnel to the PPTP-PNS.
4. A new PPP negotiation between **BRICK** and PPTP-PNS was performed, but the **BRICK** did not terminate the PPP authentication negotiation with the PPTP-PAC as required in RFC 1661. The **BRICK** still tried to carry out step 2.
5. The **BRICK** successfully authenticated on the PPTP-PNS.
6. The **BRICK** tried to authenticate on the PPTP-PAC ten times. These trials failed and so the **BRICK** terminated the link to the PPTP-PAC and therewith the VPN connection to the PPTP-PNS.

This bug has been fixed, now the **BRICK** acts as required in RFC 1661.

4.5.7 Tracing a VPN Connection

Tracing the establishment phase of a VPN (PPTP) connection in a WAN as well as in a LAN may have caused a panic, leading to the termination of the trace. In very isolated cases, the BRICK rebooted.

This bug has been fixed.

4.5.8 VPN Links Disconnected

In some isolated cases, after preceisely 7,200 seconds or if the BRICK on one side rebooted, for example, VPN links were disconnected and no further VPN connection could be established.

The result of this type of disconnection was that the state of the VPN interface on one side was set to *dormant*, while the state of the VPN interface on the other was set to *up*. Once these inconsistent states were reached, no further VPN connection could be made.

Now the inconsistency is determined and resolved: the VPN interface with the *up* state is reset to *dormant* and the VPN connection can thus be reestablished.

4.5.9 IPX Compression Protocol has not been Rejected

The **BRICK** does not support IPX Compression Protocol presently. But if a WAN partner offered the option "IPX Compression Protocol" when performing the IPXCP negotiation, the **BRICK** did not reject it with an IPXCP Configure Reject.

This bug has been fixed.

4.5.10 Link Quality Monitoring (LQM)

If Link Quality Monitoring (LQM) had been successfully negotiated, a Link Quality Report (LQR) could be sent after the Link Control Protocol (LCP) was terminated but the ISDN B channel was still established. In consequence, the **BRICK** sometimes received an LCP Protocol Reject on this LQR which resulted in a restart of the **BRICK**.

This bug has been fixed

4.5.11 Link Quality Monitoring Set to "0"

Concerning leased lines, when the variable **biboPPPLQMonitoring** in the **biboPPPTable** was set to the invalid value '*0*' instead of "off (*01*)", certain SNMP managers experienced problems.

This bug has been fixed.

4.5.12 Microsoft Point-to-Point Compression (MPPC)

If Microsoft Point-to-Point Compression (MPPC) was configured and successfully negotiated (MPPC is supported by BinTec's FM-STAC module with Hardware Release 2.1), the transmission of data which is difficult to be compressed could lead to inconsistent states of both partner's compression histories. Especially using Multilink PPP connections, termination of apparently redundant B-channels could occur as a result of stagnant or cancelled data transmission. In rare cases, this could result in the failure of the FM-STAC module or the restart of the **BRICK**.

Additionally configured Microsoft Point-to-Point Encryption (MPPE) increased the problem.

This bug has been fixed.

4.5.13 Deleting WAN Partners with RIP Receive V2

If in Setup Tool a WAN partner was deleted that had RIP Receive V2 enabled over a serial connection, the following error message appeared "Notice:udp:got DL_5".

This bug has been fixed.

4.5.14 ICMP Source Quenches

ICMP source quenches are no longer sent in case of congestion.

4.6 IPX

4.6.1 Netware Login on Booting

On attempting to connect to a BRICK from a dialup workstation (Win 95 or NT) installed as a Novell Netware client, logging in to the netware server or the NDS tree failed on booting, i.e. at the same time as the local login. The error message "*The tree or server cannot be found"* appeared.

This bug has been fixed.

4.6.2 *ipxCircType* Reset by Setup Tool Entry

If a value either *dynamic* or *ipxcpWS* was given to the variable **ipxCircType** in the SNMP shell and subsequently a change was made to the WAN partner configuration in Setup Tool, the value in the MIB table was automatically and unintentionally reset to either an unnumbered RIP or a WAN RIP. Do bear in mind, however, that when **type** = *ipxcpWS* and **NetNumber** = *0:0:0:0,* the value is correctly reset to an unnumbered RIP after such a Setup Tool entry.

This bug has been fixed.

4.6.3 BRICK IPX and Service Name Recognition

Services in an IPX network can be defined and distinguished by means of their network address, node address, socket number, the type and the service name. It is permissible for services to be distinguished merely by their service name. The BRICK IPX, however, did not allow the distinguishing of services by means of the service name.

Services can now be distinguished by means of the service name.

4.7 RADIUS

4.7.1 Framed-IP Address = 255.255.255.254

Referring to RFC 2138, an IP address from an IP address pool on the RADIUS server should be dynamically assigned to a dial-in client when the attribute Framed-IP-Address is set to 255.255.255.254. Instead of this, up to now the IP address 255.255.255.254 has been assigned to those dial-in clients.

This bug has been fixed.

4.7.2 Authentication Caused Memory Leakage

Incoming PPP calls authenticated via RADIUS caused a memory leakage of approx. 100 bytes for every connection establishment. This could lead to a restart of the **BRICK** because of insufficient RAM available.

This bug has been fixed.

4.8 HTTP Status Page

4.8.1 Internet Explorer 4.0

Using a **BRICK**'s HTTP status page with Internet Explorer 4.0, it was not possible to access sites restricted by password. For example, if one tried to show the MIB tables by clicking the link system tables, there occurred not the usual logon dialog but the error message "401Unauthorized".

The reason is that the **BRICK**'s HTTP server did support the HTTP protocol versions 0.9 and 1.0, but not the latest version 1.1. As the Internet Explorer 4.0 uses the version 1.1, the HTTP server realized that it is not its latest protocol version (1.0) and therefore used its old version (0.9), which doesn't support authentication processes.

This bug has been fixed.

If one uses Internet Explorer 4.0 and doesn't want to upgrade the **BRICK** to Release 5.1.2, the following workaround is possible:

- Go to **View** ➧ **Internet Options.**
- Click the **Advanced** tab.
- Deactivate the option **Use HTTP 1.1**, click **OK**.

4.9 IP

4.9.1 IP-Address via DHCP

It could have occurred that assigning IP-addresses via DHCP in your LAN failed, for example, with the HP JetDirect print server. The reason was an invalid string length of the "Hostname" tag, i.e. there was one byte too many.

This bug has been fixed.

4.9.2 Transmission of RIP V1 & V2 Packets

During the establishment phase of a PPP connection, RIP packets were generated, but may have been discarded before transmission. The BRICK would, however, correctly transmit RIP packets cyclically every 30 seconds.

This bug has been fixed and now RIP packets are transmitted during the establishment phase.

4.9.3 File Transfer by TFTP

If the configuration file was first saved by Xmodem, a subsequent file transfer from the BRICK to a TFTP server may have failed.

This bug has been fixed.

5 Known Issues

5.1 Outgoing FTP Connections via NAT

When outgoing FTP connections occur via NAT, data transfer does not work with some FTP servers. The connection is established, the FTP client can register with the server. Commands such as `cd` and `pwd` work, but others such as `dir` and `get` do not.

The problem can be dealt with if the client is switched to the passive mode. This is not, however, possible with all FTP clients.

5.2 Secure VPN and the BRICK

The new Microsoft authentication procedure MSCHAP V2 included in Windows NT 4.0 Service Pack 4, the post-SP3 hotfix, and Windows 95 Dial-Up Networking 1.3 Upgrade is not yet supported by Bintec. This new protocol affects only VPN connections, not, however, dial-up lines. Thus, authentication on VPN connections using these upgrades or the hotfix may be affected. By making the following entries in the Windows Registry (if the entries do not already exist), VPN connections between Windows NT/95/98 servers and workstations can be ensured:

Windows NT

When the value below is set to one, the client is forced to use MSCHAP V2 for all VPN connections. When the default value of zero is set, authentication of VPN connections can take place using MSCHAP (or CHAP or PAP).

➤ Ensure the default value is set.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RasMan\PPP

DWORD: SecureVPN

Value: 0x00000001 == force MSCHAP V2 for VPN connections

Value: 0x00000000 == do not force secure MSCHAP V2 (default)

Service Pack 4 or post-SP3 hotfix also includes a historyless mode for encryption and compression over PPTP connections.

➤ If you want normal MPPE compression and encryption negotiation, set the value below to "Disabled".

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\NdisWan\Parameters

DWORD: Historyless

Value: 0x00000001 == Enabled (default)

Value: 0x00000000 == Disabled

Windows 95/98

For Windows 95/98 users, MSCHAP V2 could be upgraded with Windows 95 Dial-Up Networking 1.3 Upgrade. The same advice as above applies here, i.e. do not force secure mode.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RemoteAccess

DWORD: SecureVPN

Value: 0x00000001 == Force secure mode (MSCHAP V2 plus data encryption) on all PPTP connections

Value: 0x00000000 == Do not force secure mode on PPTP connections (default)

For more information see:

http://support.microsoft.com/support/kb/articles/q154/0/91.asp

http://support.microsoft.com/support/kb/articles/q189/5/94.asp

http://support.microsoft.com/support/kb/articles/q189/5/95.a

5 Known Issues