If the connection is being created for direct local access using a COM port, you should set the phone number to 123. This number will be intercepted by the unit and recognized as an attempt to connect locally.

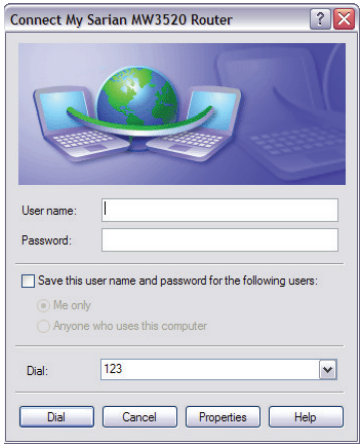
If the connection is being created for remote access, enter the correct ISDN telephone number (including the area code) for the remote unit.

When you have done this click Next >. The final dialog screen will confirm that the connection has been created and includes a check box to allow you to create a shortcut on your desktop if necessary. Click on Finish to complete the task.

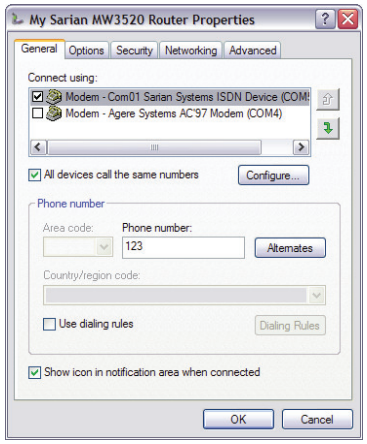
Configuring the New DUN Connection

The new DUN connection that you have just created may now be used to connect to the unit but before you do this, you will need to check some of the configuration properties.

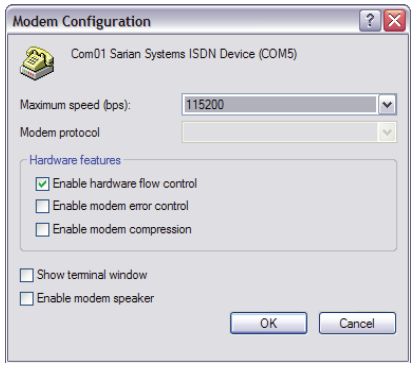
1. Click on the Start button and select Connect To > My Digi Router (substituting the connection name you chose).



2. Click on the **Properties** button to display the properties dialog for the connection:

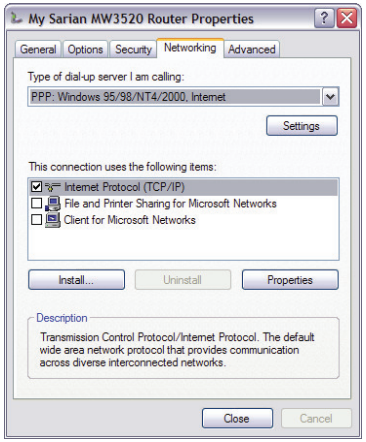


3. On the **General** tab, click the **Configure...** button to display the Modem Configuration dialog:

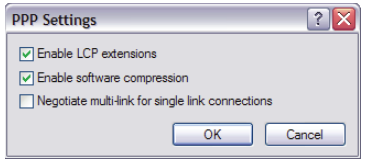


Make sure that the **Maximum speed (bps):** value is set to **115200** and that the **Enable hardware flow control** box is checked. Click **OK** when you have finished to return to the main properties dialog.

4. Now select the **Networking** tab:



Make sure that the Type of dial-up server I am calling is set to PPP: Windows 95/98/NT/2000, Internet and click on Settings:



Make sure that all three options are **unchecked** before clicking **OK** to return to the **Networking** tab. In the **This connection uses the following items** list, **Internet Protocol (TCP/IP)** should be the only item that is checked. Make sure that this is the case and then click **OK** to return to the main dialog. You are now ready to initiate a connection.

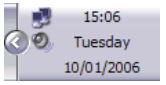
Initiating a DUN Connection

In the main dialog, you are asked to enter a username and password. The default settings for your unit are "**username**" and "**password**" respectively but you should change as soon as possible in order to prevent unauthorised access to your unit (refer to the section entitled Configuration - Security > Users for instructions on how to do this). The username is not case sensitive, but the password is.

Note:
When you type the password it will appear as a series of dots to ensure privacy.

Once you have entered these, initiate a connection to your unit by clicking the **Dial** button. During the dialling and connection process, you may see a series of status dialog boxes and, if the connection is successful, the final dialog box will indicate that the PPP login has been authenticated.

After a short delay, this dialog will minimise to a "linked computers" icon in the Windows taskbar:



You should now be ready to access the built-in web pages using your Web browser. The default "web address" for the unit is **http://1.2.3.4**. By default, this is also mapped to the system IP hostname **digi.router**.

You will need a valid username and password to access the web interface. Once again, the default settings are **username** and **password** respectively. If these values do not allow access, you should contact your system administrator.

SQL commands

When IPSec Egroups are used with a SQL database for dynamic Eroute configuration, there are CLI commands that will help with configuration and troubleshooting on the Digi router.

Local Database commands

As well as using an external SQL database, the Digi can cache the SQL table entries it learns from the SQL server in RAM so if the SQL server goes offline for any reason, the database entries are still available to renew existing IPSec SA's.

To configure the caching options the command used is *sql 0 <parameter> <value>*

The following parameters are available to configure the caching of database entries:

dbsrvmem <n>

This parameter is used to specify the amount of memory (RAM) the MySQL server cache should use. Where <n> is specified in multiples of 1k. e.g. 10Mb = 10240

To calculate the amount of memory to specify in this parameter:

1. Look at the size of the database file (.csv) that will be loaded into the Digi memory.

2. Double this value and add 100Kb, for example, if the csv file is 200Kb, this would make a value of 500Kb for the memory allocation. Use the command *sql 0 dbsrvmem 500*

3. Load the database file into memory and check the memory allocated and free using the *smem* command. This will show the memory allocated and left available. Increase the memory in the *dbsrvmem* command if required.

dbfile <name>

This is the name of the csv file that the Digi will use to store the table definitions (1st line) and data records. This file is stored in flash and is used to populate the database stored in RAM on power up or when a new file matching this name has just been stored. The dbfile can be populated with records or be empty except for the definitions line. The dbfile stored in RAM will be populated from both the dbfile stored in flash and (if configured) via caching items learnt from the main SQL server. The dbfile in flash can then be updated from the dbfile in RAM and saved.

dbname <name>

This is the name of the backup database in case the main database goes offline. This name needs to match the database name in use on the SQL server.

learn <off|on>

When enabled, the Digi will cache entries learnt via the main SQL database in a file stored in RAM. This can be used as a backup in the event of the main SQL database going offline. To use learning mode, at least one column in the csv dbfile must be marked as a unique key, with the U prefix.

For example, remip is marked as the unique key:

peerip[IP],bakpeerid[IP],peerid[K20],password[20],ourid[20],remip[UKIP],remmsk[IP]

Learning mode - Saving entries

When learning mode is used, the dynamic backup database is stored in RAM. This database will be lost if the Digi router is power cycled. The database in RAM can be saved to flash to over-write the dbfile with the one in RAM that includes the learnt entries or it can be saved to a new file.

To save the dbfile to flash from RAM, use the following command.

sqlsave 0 <filename>

Where *<filename>* is the name of the destination file.

For example, to save the learnt database entries to a file called backup.csv

sqlsave 0 backup.csv

If there are no learnt entries, this command will not create a file. To view the number to learnt entries, use the command sql 0 ? and refer to the section headed *Learning info*.

```
Learning info.
items learned:0
matched retrievals:0
OK
```

Configure a TransPort to use a backup database

Once the Digi has been configured to run a SQL csv database locally, this backup csv database can be used in the event of the main SQL database going offline. The configuration parameters required are:

Configure the IP address of the SQL server to use.

egroup 0 dbhost "192.168.0.50"

Configure the IP address of the SQL server that will have a backup database. If a socket connection fails to this IP address, the Digi will use the backup IP address.

ipbu 0 IPaddr "192.168.0.50"

Configure the backup database IP address. eg. the loopback address of the Digi router or an alternative SQL server, this eaxmple shows the loopback IP address of the Digi router.

ipbu 0 BUIPaddr "127.0.0.1"

Set the amount of time in seconds that the connection to the main SQL server will be retried.

ipbu 0 retrysec 30

Set the Digi to use the backup IP address if the main database is unavailable.

ipbu 0 donext ON

For example, to configure and use a local backup database when the main SQL database at 192.168.0.50 is offline, the configuration may look similar to this:

```
egroup 0 dbhost "192.168.0.50"
sql 0 dbsrvmem 200
sql 0 dbfile "sardb.csv"
sql 0 dbname "sarvpns"
sql 0 learn ON
sqlsave 0 backup.csv
ipbu 0 IPaddr "192.168.0.50"
ipbu 0 BUIPaddr "127.0.0.1"
ipbu 0 retrysec 30
ipbu 0 donext ON
```

Memory info

smem

Displays the amount of memory allocated, in use and available for use by the MySQL server on the Digi.

Transact SQL commands

To query a SQL database manually using transact SQL statements, the following commands can be used.

To connect to the SQL server and database:

sqlcon <host> <user> <pwd> <database>

For example:

sqlcon 192.168.0.50 sqluser sqlpass eroute-db

To issue transact SQL statements:

sqldo <"cmd">

For example:

*sqldo "select * from site where subnet='10.110.100.0' limit 3"*

To limit the sqldo command to only act on specified fields, the following command can be used:

sqlfields "<field1> <field2> <field3>"

For example:

sqlfields "remmsk password peerip"

After issuing the sqlfields command, all further sqldo commands will apply to these fields only.

When finished, to close the SQL server connection correctly:

sqlclose

If the database being queried is held locally on the Digi, these commands can be preceded with the SQL debug command to give extra feedback on any commands issued.

To enable the SQL debug:

```
sql 0 debug_opts 3
```

To view the debug data via the ASY 0 port:

```
debug 0
```

To view the debug data via telnet:

```
debug t
```

To disable the SQL debug:

```
sql 0 debug_opts 0
debug off
```


Answering V.120 Calls

V.120 is a protocol designed to provide high-speed point-to-point communication over ISDN. It provides rate adaptation and can optionally provide error control. Both the calling and called units must be configured to use V.120 before data can be transferred. Similarly, if one unit is configured to use the error control facility, the other must be configured in the same way.

Initial Set Up

Before using V.120 you must first bind one of the two available V.120 instances to the required ASY port using the ***Configuration - Network > Interfaces > Serial > Protocol Bindings*** page or by using the bind command from the command line, for example:

```
bind v120 0 asy 0
```

You should also select the appropriate method of flow control for the ASY port using the ***Configuration - Network > Interfaces > Serial > Serial Port n*** page or by using the AT&K command from the command line. Other ASY port options such as command echo, result code format, etc. should also be configured as necessary.

Initiating a V.120 Call

Once the initial configuration is complete, V.120 calls may be initiated using the appropriate ATD command. For example:

```
atd01234567890
```

A successful connection will be indicated by a CONNECT result code being issued to the ASY port and the unit will switch into on-line mode. In this mode, all data from the terminal attached to the bound ASY port will be passed transparently through the unit across the ISDN network to the remote system. Similarly, all data from the remote system will be passed directly to the terminal attached to the bound ASY port.

If a V.120 call fails the unit will issue the NO ANSWER or NO CARRIER result code to the ASY port and remain in command mode.

The ATD command may also be used to route a call to an ISDN sub-address by following the telephone number with the letter S and the required sub-address value. For example:

```
atd01234567890s003
```

In this case, the remote system will only answer the call if it has been configured to accept incoming calls on the specified sub-address.

Answering V.120 Calls

V.120 answering can be enabled from the command interface by setting register S0 for the appropriate ASY port to a non-zero value. For example:

```
ats0=1
```

You should ensure that you have set S0 for the correct ASY port by either entering it directly on that port or by using the AT\PORT command to select the correct port first.

The actual value used for the parameter sets the number of rings the unit will wait before answering.

Finally, you must ensure that there are no conflicts with other protocols configured to answer on other ASY ports. This can be done by disabling answering for the other ports/protocols or by using the MSN and/or Sub-address parameters to selectively answer calls to different telephone numbers using different protocols.

For example, if you have subscribed to the ISDN MSN facility, you may have been allocated say four telephone numbers ending in 4, 5, 6 and 7. You could then set the MSN parameter for the appropriate V.120 instance to 4 to configure V.120 to answer only incoming calls to the MSN number ending in 4.

You should check that if PPP answering is enabled you have NOT selected the same MSN and Sub-address values for PPP. If they are the same, V.120 will answer the call ONLY if S0 is set to 1. Otherwise, PPP will take priority and answer the call.

ANSWERING ISDN CALLS

Digi routers are capable of answering incoming B-channel ISDN calls with 3 main protocols. Usually several instances of these protocols exist. This section explains how answering priorities work for the different protocols.

Protocol Entities

The following protocol instances are capable of answering an incoming ISDN call:

Adapt

Adapt instances provide rate adaptation protocols such as V.120 or V.110.

LAPB

LAPB instances allow the unit to answer incoming X.25 calls over ISDN. They can optionally connect the caller to a synchronous serial port, an asynchronous serial port bound to a PAD, or switch the call to another interface.

PPP

IP data tunnelled over PPP instances allow remote access to the unit's IP-based management features and also facilitate onward IP routing through any of the unit's IP enabled interfaces.

The unit will automatically answer an incoming ISDN call if any of the following statements are true (subject to the entity MSN, Calling Number and Sub-address parameters being set to their default values):

- An Adapt instance is bound to an asynchronous serial port (ASY) and the answer ring count (S0) for that serial port is set to 1
- A LAPB instance has its answering parameter set to On
- A PPP instance has its answering parameter set to On

If more than one of these protocols are configured to auto answer then the priority is as follows:

Adapt instances (normally V.120) will take priority over LAPB, which will take priority over PPP. If an Adapt instance is bound to an asynchronous serial port (ASY port) but the answer ring count (ATS0) is not set to 1 for that same serial port then Adapt entity will not answer automatically. If any other protocol entities (e.g. LAPB, PPP or another Adapt instance) are configured to answer then one of these protocol entities will answer the call. If no other protocol entities are configured to answer then a repeating RING message will be sent out of the serial port and the RS232 ring indicator control will be activated. If a terminal attached to the serial port sends ATA followed by carriage return then the ISDN call will be answered by the Adapt entity and any incoming data will be channelled out of the serial port and vice-versa.

Multiple Subscriber Numbers

An MSN (multiple subscriber number) is an alternative number provided by the telephone service provider which when dialled will also route through to your ISDN line. It is possible to purchase several MSNs for an ISDN line. This means that in effect one ISDN line can have several ISDN numbers.

Every entity in the router which is capable of answering an ISDN call (Adapt, LABP and PPP) has an MSN parameter.

A protocol entity's MSN parameter can be used to:

• cause a protocol instance not to answer an incoming ISDN call (if the trailing digits of the ISDN number called do not match the entry in this field).

• increase the answering priority of an instance (if more than one protocol instance is configured to answer and the trailing digits of the ISDN number called match the value of the MSN parameter for a particular protocol instance).

Example

Consider the following:

- an Adapt instance is bound to a serial port and ATS0 for that serial port is set to 1
- PPP instance 0 has answering turned On
- the ISDN line to which the router is connected has two numbers: the main number is 123456 and the MSN number is 123789

Normally, because ADAPT has a higher answering priority than PPP, the Adapt instance will answer when either of the numbers are called. However if the ISDN number dialled is 123456 and 456 is entered into the MSN parameter of PPP then PPP will answer instead. This will also have the effect of preventing PPP from answering if any other ISDN number (e.g. 123457) has been called.

This means that whenever 123456 is called the PPP instance will answer and that whenever 123789 is called the V120 instance will answer.

It is possible to connect multiple ISDN devices to the same ISDN line. MSNs can then be used to allow the different ISDN devices to be dialled individually (i.e. dial the main ISDN number and get through to ISDN device one, dial the first MSN and get through to ISDN device number two, dial the second MSN and get through to ISDN device number three, etc.).

Multiple PPP Instances

It is also possible to configure multiple instances of a particular entity to answer. For example, PPP instance, 0, 1 and 4 could be configured to answer. In this case provided that none of the PPP instances are busy, the PPP instance with the highest number will answer first. MSNs can also be used to ensure that a chosen PPP instance answers the call.

Multiple protocol entity answering instance rules:

ADAPT

The lowest free Adapt instance with auto-answering enabled will answer first.

PPP

The lowest free PPP instance with answering on will answer first.

LAPB

The lowest free LAPB instance with answering on will answer first.

X.25 PACKET SWITCHING

Introduction

X.25 is a data communications protocol that is used throughout the world for wide area networking across Packet Switched Data Networks (PSDNs). The X.25 standard defines the way in which terminal equipment establishes, maintains and clears Switched Virtual Circuits (SVCs), across X.25 networks to other devices operating in packet mode on these networks.

The protocols used in X.25 operate at the lower three layers of the ISO model. At the lowest level the Physical layer defines the electrical and physical interfaces between the DTE and DCE. Layer 2 is the Data Link Layer that defines the unit of data transfer as a "frame" and includes the error control and flow control mechanisms. Layer 3 is the Network layer. This defines the data and control packet structure and the procedures used to access services that are available on PSDNs.

A further standard, X.31 defines the procedures used to access X.25 networks via the ISDN B and D-channels.

Digi ISDN products include support for allowing connected terminals to access X.25 over ISDN B channels, the ISDN D-channel or over TCP. They can also be configured so that if there is a network failure it will automatically switch to using an alternative service. The Packet Assembler/Disassembler (PAD) interface conforms to the X.3, X.28 and X.29 standards.

Up to six PAD instances (from an available pool of 8), can be created and dynamically assigned to the asynchronous serial ports or the REM pseudo-port.

Each application that uses the unit to access an X.25 network will have its own particular configuration requirements. For example, you may need to program your Network User Address (NUA) and specify which Logical Channel Numbers (LCNs) should be used on your X.25 service. This information will be available from your X.25 service provider. You will also need to decide whether your application will use B or D-channel X.25.

Once you have this information, the PAD configuration pages can be used to set up the appropriate parameters.

B-channel X.25

The unit can transfer data to/from X.25 networks over either of the ISDN B-channels.

Once the unit has been configured appropriately, the ISDN call to the X.25 network can be made using an ATD command or by executing a pre-defined macro. The format of the ATD command allows you to combine the ISDN call and the subsequent X.25 call in a single command. Alternatively, the X.25 call may be made separately from the PAD> prompt once the ISDN connection to the X.25 network has been established.

D-channel X.25

The unit can transfer data to/from X.25 networks over the ISDN D-channel if your ISDN service provider supports this facility. The speed at which data can be transferred varies depending on the service provider but is generally 9600bps or less.

X.28 Commands

Once an X.25 session layer has been established the unit switches to "PAD" mode. In this mode operation of the PAD is controlled using the standard X.28 PAD commands listed in the following table:

Command	Description
CALL	Make an X.25 call
CLR	Clear an X.25 call
ICLR	Invitation to CLR
INPAR?	List X.3 parameters of specified PAD instance
INPROF	Load or save specified PAD profile
INSET	Set X.3 parameters of specified PAD instance
INT	Send Interrupt packet
LOG	Logoff and disconnect
PAR?	List local X.3 parameters
PROF	Load or save PAD profile
RESET	Send reset packet
RPAR?	List remote X.3 parameters
RSET	Set remote X.3 parameters
SET	Set local X.3 parameters
STAT	Display channel status

CALL Make an X.25 Call

The full structure of a CALL command is:

CALL [<facilities->]<address>[D<user data>]

where:

<facilities-> is an optional list of codes indicating the facilities to be requested in the call (separated by commas, terminated with a dash)

<address> is the destination network address.

<user data> is any optional user data to be included with the call.

The facility codes supported are:

F	Fast select - no restriction
Q	Fast select - restricted response
Gnn	Closed User Group
Gnnnn	Extended Closed User Group
R	Reverse charging
N<NUI>	Network User Identity code (NUI)

Example

CALL R,G12,NMYNUI-56512120DHello

places a call to address 56512120 using reverse charging and specifying Closed User Group 12. The string "MYNUI" is your Network User Identity and the string "Hello" appears in the user data field of the call packet.

Note:
The particular facilities that are available will vary between X.25 service providers.

If a CALL command is issued without the address parameter, it is assumed that you wish to go back on-line to a previously established call (having used the PAD recall facility to temporarily return to the PAD> prompt).

Fast select (ISDN B-channel only)

When the standard Fast select facility is requested using the "F" facility code, the call packet generated by the CALL command is extended to allow the inclusion of up to 124 bytes of user data. For example:

CALL F-1234567890DThis DATA sent with call packet

would cause an X.25 CALL packet to be sent using the Fast select facility including the message "This DATA sent with call packet" (the Carriage Return used to enter the command is not transmitted). Without the inclusion of the Fast select facility code, only the first 12 characters would be sent.

When a Fast select CALL has been made the PAD accepts an extended format response from the called address. This response, consisting of up to 124 bytes of user data, may be appended to the returning call accepted or call clear packet. When one of these packets is received, the user data is extracted and passed from the PAD to the terminal immediately prior to the "CLR DTE . . ." message in the case of a call clear packet or "CON COM" message in case of a call accepted packet.

When a restricted response Fast select call has been made using the Q facility code, the call packet indicates that a full connection is not required so that any response to the user data in the CALL packet should be returned in a call clear packet.

When the PAD receives an incoming call specifying Fast select, the call is indicated to the terminal in the normal way. For example:

IC 1234567890 FAC: Q,W:2 COM

would indicate that an incoming call had been received requesting Restricted response fast select and a window size of 2. The user (or system) then has 15 seconds in which to pass up to 124 bytes of data to the PAD to be included in the clear indication packet that is sent in response to the call.

The PAD does NOT differentiate between standard and restricted response Fast select on incoming calls and, consequently, will always respond with a clear indication.

Network User Identity (NUI)

The N facility code allows you to include your Network User Identity in the call packet. For security reasons the PAD echoes each character as an asterisk (*) during the entry of an NUI. Some X.25 services use the NUI field to pass both a username and password for validation. For example, if your Username is MACDONALD and your password is ASDF, a typical CALL command would have the format:

```
CALL NMACDONA;ASDF-56512120
```

where the ";" is used to separate the username from the password.

Closed User Group (CUG)

Most X.25 networks support Closed User Groups. They are used to restrict subscribers to only making calls or receiving calls from other members of the same CUG. The CUG number effectively provides a form of sub-addressing that is used in conjunction with the NUA to identify the destination address for a call.

When the G facility code is specified in a CALL packet, it must be followed by the CUG number. This may be a 2 or 4 digit number. If you are a member of a closed user group, the network may restrict you to only making calls to or receiving calls from other members of the same group.

Reverse charging

Reverse charging, specified using the R facility code, allows outgoing calls to be charged to the account of destination address. Whether or not a call is accepted on a reverse charging basis is determined by the service provider and by the type of account held by the called user.

Calling user data

The calling user data field for a normal call may contain up to 12 bytes of user data. If the first character is an exclamation mark (!), the PAD omits the four byte protocol identifier and allows the full 16 bytes as user data. The same is true for a fast select call except that the maximum amount of user data is increased from 124 to 128 bytes.

When entering user data, the tilde character (~) may be used to toggle between ASCII and binary mode. In ASCII mode data is accepted as typed but in binary mode each byte must be entered as the required decimal ASCII code separated by commas. For example to enter the data "Line1" followed by [CR][LF] and "Line2" you would enter:

```
DLine1~13,10~Line2
```

Aborting a CALL

An X.25 CALL may be aborted using the X.28 CLR command, by pressing [Enter] or by dropping DTR from the terminal while the call is in progress. Dropping DTR will also terminate an established call.

If a call is terminated by the network or by the remote host, the unit returns a diagnostic message before the NO CARRIER result code. Messages may be numeric or verbose depending on the setting of the ATV command.

The following table lists the verbose messages and equivalent numeric codes:

Code	Verbose message
1	Unallocated (unassigned) number
2	No route to specified transit network

Code	Verbose message
3	No route to destination
4	Channel unacceptable
6	Channel unacceptable
7	Call awarded and being delivered in an established channel
16	Normal call clearing
17	User busy
18	No user responding
19	No answer from user (user alerted)
21	Call rejected
22	Number changed
26	Non-selected user clearing
27	Destination out of order
28	Invalid number format
29	Facility rejected
30	Response to STATUS ENQUIRY
31	Normal, unspecified
34	No circuit/channel available
38	Network out of order
41	Temporary failure
42	Switching equipment congestion
43	Access information discarded
44	Requested circuit/channel not available
47	Resources unavailable, unspecified
49	Quality of service unavailable
50	Requested facility not subscribed
57	Bearer capability not authorized
58	Bearer capability not presently available
63	Service or option not available, unspecified
65	Bearer capability not implemented
66	Channel type not implemented
69	Requested facility not implemented
70	Only restricted digital information bearer
79	Service or option not implemented, unspecified
81	Invalid call reference value

Code	Verbose message
82	Identified channel does not exist
83	A suspended call exists, but this call identity does not
84	Call identity in use
85	No call suspended
86	Call having the requested call identity has been cleared
88	Incompatible destination
90	Destination address missing or incomplete
91	Invalid transit network selection
95	Invalid message, unspecified
96	Mandatory information element is missing
97	Message type non-existent or not implemented
98	Message not compatible with call state or message type nonexistent or not implemented
99	Information element non-existent or not implemented
100	Invalid information element contents
101	Message not compatible with call state
102	Recovery on timer expired
111	Protocol error, unspecified
127	Interworking, unspecified
128	General level 2 call control failure (probable network failure)

Note:
Some verbose messages may be abbreviated by the unit.

CLR Clear an X.25 Call

The CLR command is used to clear the current call and release the associated virtual channel for further calls. On completion of call clear the PAD> prompt is re-displayed. A call may also be cleared as a result of a number of other situations. If one of these situations occurs, a message is issued to the PAD in the following format:

CLR <Reason> C:<n> - <text>

where:

<Reason> is a 2/3 character clear down code

<n> is the numeric equivalent of the clear down code

<text> is a description of the reason for clear down

The clear down reason codes supported by the unit are listed in the following table:

Reason Code	Numeric Code	Text
DTE	0	by remote device
OOC	1	number busy
INV	3	invalid facility requested
NC	5	temporary network problem
DER	9	number out of order
NA	11	access to this number is barred
NP	13	number not assigned
RPE	17	remote procedure error
ERR	19	local procedure error
ROO	21	cannot be routed as requested
RNA	25	reverse charging not allowed
ID	33	incompatible destination
FNA	41	fast select not allowed
SA	57	ship cannot be contacted

If an unknown reason code is received, the text field is blank.

ICLR Invitation To CLR

The ICLR command "invites" the remote X.25 service to CLR the current X.25 session.

INT Send Interrupt Packet

INT causes PAD to transmit an interrupt packet. These packets flow "outside" normal buffering/flow control constraints and are used to interrupt the current activity.

LOG Logoff and Disconnect

LOG is used to terminate an X.25 session. It causes the PAD to clear any active X.25 calls, disconnect and return to AT command mode.

PAR? List Local X.3 Parameters

PAR? lists the local X.3 parameters for the current session.

PROF Load/Save PAD Profile

The PROF command is used to store or retrieve a pre-defined set of X.3 PAD parameters (referred to as a PAD profile). The information is stored in system file called X3PROF. There are four pre-defined profiles numbered 50, 51, 90 and 91. Additionally, you may create four "user PAD profiles" numbered 1 to 4.

Profile 50 is automatically loaded when a PAD is first activated. To load one of the other pre-defined profiles use the PROF command followed by the required profile number. For example:

PROF 90

To create a User PAD profile you must use the SET command to configure the various PAD parameters to suit your application and then use the PROF command in the format:

```
PROF &nn
```

where "nn" is the number of the User PAD profile to be stored, e.g. 03. Alternatively, you may use the web interface to edit the parameters directly (***Configuration - Network > Legacy Protocols > X.25 > PADs n-n > PAD n > PAD Settings***).

The pre-defined profiles (50, 51, 90, 91), cannot be overwritten and are permanently configured as shown in the following table:

Parameter	Profile			
	50	51	90	91
1	1	0	1	0
2	0	0	1	0
3	0	0	126	0
4	5	5	0	20
5	0	3	1	0
6	5	5	1	0
7	0	8	2	2
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	15	15	15	15
12	0	3	1	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	8	8	127	127
17	24	24	24	24
18	18	18	18	18
19	2	2	1	1
20	64	64	0	0
21	0	0	0	0
22	0	0	0	0

Stored X.25 PAD profiles are held in non-volatile memory and will not be lost when the unit is switched off.

When used in the format:

```
prof nn
```

the PROF command loads the stored profile specified by "nn".

RESET Send Reset Packet

RESET is used to issue a reset for the current call to the network. It does NOT clear the call but it does return the network level interface to a known state by re-initialising all Level 3 network control variables. All data in transit will be lost.

RPAR? Read Remote X.3 Parameters

RPAR? lists the current X.3 parameter settings for the remote system.

RSET Set Remote X.3 Parameters

RSET is used to set one or more X.3 parameters for the remote system. It is entered in the format:

```
RSET par #:value[,par #:value[,par #:value ...]]
```

SET Set Local X.3 Parameters

SET is used to set one or more of the local X.3 parameters for the duration of the current session. The format of the command is:

```
SET par #:value[,par #:value[,par #:value ...]]
```

STAT Display Channel Status

STAT displays the current status for each logical channel indicating whether it is free or engaged. For example:

```
stat
PAD   STATE
1     ENGAGED
2     FREE
3     FREE
4     FREE
```

PPP OVER ETHERNET

PPP over Ethernet (PPPoE) is a means of establishing a PPP connection over the top of an Ethernet connection. The implementation provided is compliant with RFC 2516, "A Method for Transmitting PPP Over Ethernet". A typical application would be to allow non-PPPoE enabled devices to access Internet services where the connection to the Internet is provided by an ADSL bridge device.

Using the Web Page(s)

There is no dedicated web page for configuring the unit to use PPPoE; rather there are a number of parameters that appear on other web pages that must be used in conjunction with each other to establish a PPPoE connection over the appropriate Ethernet interface.

In particular, the following configuration pages and parameters are important.

On the appropriate ***Configuration - Network > Interfaces > Advanced > PPP n - n*** pages, you should configure the following parameters on the following pages:

Configuration - Network > Interfaces > Advanced > PPP n - n > PPP n

As a minimum requirement the **Username** and **Password** parameters should be initialised.

The parameter **This PPP interface will use x,y** defines the physical Ethernet interface over which the PPPoE session will operate. In most cases this is PPPoE 0 (for Ethernet 0). The fact that you have selected "PPPoE 0" as the physical interface for operation with PPP automatically enables PPPoE mode. If another Ethernet instance is used, Eth 1 for example, this will need to be specified as PPPoE 1 to ensure the correct MAC address is used, this is in the format 0 or blank for port 0, 1 for port 1, 2 for port 2 etc.

If necessary, continue to the page ***Configuration - Network > Interfaces > Advanced > PPP n - n > PPP n > Advanced*** and set the **Enable "Always On" mode of this interface** parameter to "**On**" to configure the unit so that it will attempt to renegotiate the PPP link should it go down for any reason.

Configuration - Network > Interfaces > Advanced > PPP n - n > PPP n > PPP Negotiation

The **advanced PPP options** on this page should be initialised as required by your ISP.

In addition:

Desired Local MRU and **Desired Remote MRU** should be set to "1492".

Request Local ACFC and **Request Remote ACFC** should be set to "No".

Request Local PFC and **Request Remote PFC** should be set to "No".

Desired Local ACCM and **Desired Remote ACCM** should be set to "0xffffffff".

Using Text Commands

There are no specific PPPoE commands available to the user via the text command interface. The appropriate *ppp* CLI commands should be used to set the required options.

IPSEC AND VPNS

What is IPSec?

One inherent problem with the TCP protocol used to carry data over the vast majority of LANs and the Internet is that it provides virtually no security features. This lack of security, and recent publicity about "hackers" and "viruses", prevent many people from even considering using the Internet for any sensitive business application. IPSec provides a remedy for these weaknesses adding a comprehensive security "layer" to protect data carried over IP links.

IPSec (Internet Protocol Security) is a framework for a series of IETF standards designed to authenticate users and data, and to secure data by encrypting it during transit. The protocols defined within IPSec include:

- IKE – Internet Key Exchange protocol
- ISAKMP – Internet Security Association and Key Management Protocol
- AH – Authentication Header protocol
- ESP – Encapsulating Security Payload protocol
- HMAC – Hash Message Authentication Code
- MD5 – Message Digest 5
- SHA-1 – Security Hash Algorithm

and the cryptographic (encryption) techniques include:

- DES – Data Encryption Standard
- 3DES – Triple DES
- AES – Advanced Encryption Standard (also known as Rijndael)

Two key protocols within the framework are AH and ESP. AH is used to authenticate users, and ESP applies cryptographic protection. The combination of these techniques is designed to ensure the integrity and confidentiality of the data transmission. Put simply, IPSec is about ensuring that:

- only authorised users can access a service and
- that no one else can see what data passes between one point and another.

There are two modes of operation for IPSec, transport mode and tunnel mode.

In transport mode, only the payload (i.e. the data content), of the message is encrypted. In tunnel mode, the payload and the header and routing information are all encrypted thereby by providing a higher degree of protection.

Data Encryption Methods

There are several different algorithms available for use in securing data whilst in transit over IP links. Each encryption technique has its own strengths and weaknesses and this is really, a personal selection made with regard to the sensitivity of the data you are trying to protect. Some general statements may be made about the relative merits but users should satisfy themselves as to suitability for any particular purpose.

DES (64-bit key)

This well-known and established protocol has historically been used extensively in the banking and financial world. It is relatively "processor intensive", i.e. to run efficiently at high data rates a powerful processor is required. It is generally considered very difficult for casual hackers to attack but may be susceptible to determined attack by well-equipped and knowledgeable parties.

3-DES (192-bit key)

Again, this is a well-established and accepted protocol but as it involves encrypting the data three times using DES with a different key each time, it has a very high processor overhead. This also renders it almost impossible for casual hackers to attack and very difficult to break in any meaningful time frame, even for well-equipped and knowledgeable parties.

AES (128-bit key)

Also known as Rijndael encryption, AES is the new "de-facto" standard adopted by many USA and European organisations for sensitive applications. It has a relatively low processor overhead compared to DES and it is therefore possible to encrypt at higher data rates. As with 3-DES, it is almost impossible for casual hackers to attack and is very difficult to break in any meaningful time frame, even for well-equipped and knowledgeable parties.

To put these into perspective, common encryption programs that are considered "secure" (such as PGP) and on-line credit authorisation services (such as Web-based credit card ordering) generally use 128-bit encryption.

Note:
Data rates are the maximum that could be achieved but may be lower if other applications are running at the same time or small IP packet sizes are used.

What is a VPN?

VPNs (Virtual Private Networks) are networks that use the IPSec protocols to provide one or more secure routes or "tunnels" between endpoints. Users are issued either a shared "secret" key or "public/ private" key pair that is associated with their identity. When a message is sent from one user to another, it is automatically "signed" with the user's key. The receiver uses the secret key or the sender's public key to decrypt the message. These keys are used during IKE exchanges along with other information to create session keys that only apply for the lifetime of that IKE exchange.

The Benefits of IPSec

IPSec is typically used to attain confidentiality, integrity, and authentication in the transport of data across inherently insecure channels. When properly configured, it provides a highly secure virtual channel across cheap, globally available networks such as the Internet, or creates a "network within a network" for applications such as passing confidential information between two users across a private network.

X.509 Certificates

In the previous section, security between two points was achieved by using a "pre-shared secret" or password. Certificates provide this sort of mechanism but without the need to manually enter or distribute secret keys. This is a complex area but put simply a user's certificate acts a little like a passport providing proof that the user is who they say they are and enclosing details of how to use that certificate to decrypt data encoded with it. Passports however can be forged so there also needs to be proof that the passport has been properly issued and hasn't been changed since it was. On a paper passport this is achieved by covering the photograph with a coating that shows if it has been tampered with, embedding the user's name in code in a long string of numbers, etc. In the same way, for a Security Certificate to be genuine it has to be protected from alteration as well. Like a passport, you also have to trust that the issuer is authorized and competent to create the certificate.

Certificates use something called a "Public/Private Key Pair". This a complex area but the principle is that you can create an encryption key made up from two parts, one private (known only to the user), the other public (known to everyone). Messages encrypted with someone's public key can only be recovered by the person with the Public AND Private key but as encrypting the message to someone in the first place only requires that you know their public key, anyone who knows that can send them an encrypted message, so you can send a secure message to someone knowing only their publicly available key. You can also prove who you are by including in the message your "identity" whereupon they can look up the certified public key for that identity and send a message back that only you can understand. The important principles are that a) your private key cannot be determined from your public key and b) you both need to be able to look up the others certified ID. Once you've established a two way secure link you can use it to establish some rules for further communication.

Before this gets any more complicated we'll assume that Digi International are a competent authority to issue certificates and given that they exist and are valid, see how they are used.

Generally, the issuing and management of certificates will be provided as a managed service by Digi or its partners, but some general information is provided here for system administrators.

Certificates are held in non-volatile files on the unit. Any private files are named privxxxx.xxx and cannot be copied, moved, renamed, uploaded or typed. This is to protect the contents. They can be overwritten by another file, or deleted.

Two file formats for certificates are supported:

- PEM – Privacy Enhanced MIME
- DER – Distinguished Encoding Rules

Certificate and key files should be in one of these two formats, and should have an extension of ".pem" or ".der" respectively.

Note:
The equivalent filename extension for .PEM files in Microsoft Windows is ".CER". By renaming ".PEM" certificate files to ".CER", it is possible to view their makeup under Windows.

The unit maintains two lists of certificate files. The first is a list of "Certificate Authorities" or CAs. Files in this list are used to validate public certificates sent by remote users. Public certificates must be signed by one of the certificates in the CA list before the unit can validate them. Certificates with the filename CA*.PEM and CA*.DER are loaded into this list at start-up time. In the absence of any CA certificates, a public certificate cannot be validated.

The second list is a list of public certificates that the unit can use to obtain public keys for decrypting signatures sent during IKE exchanges. Certificates with a filename CERT*.PEM and CERT*.DER are loaded into this list when the unit is powered on or rebooted. Certificates in this list will be used in cases where the remote unit does not send a certificate during IKE exchanges. If the list does not contain a valid certificate communication with the remote unit cannot take place.

Both the host and remote units must have a copy of a file called CASAR.PEM. This file is required to validate the certificates of the remote units.

In addition, the host unit should have copies of the files CERT02.PEM (which allows it to send this certificate to remote units) and PRIVRSA.PEM. Note that before it can send this certificate, the "Remote ID" parameter in the Configuration - Network > Virtual Private Networking (VPN) > IPsec > IPsec Tunnels > IPsec n - n > IPsec n page must be set to "host@Digi.co.uk".

The remote unit must have copies of CERT01.PEM and PRIVRSA.PEM. In addition, any Eroutes that are going to use certificates for authentication should be configured as follows:

Our ID

Should be set to "info@Digi.co.uk". This is the same as the subject "Altname" in certificate CERT01.PEM which makes it possible for the router to locate the correct certificate to send to the host.

Authentication Method

Should be set to RSA Signatures. This indicates to IKE that RSA signatures (certificates) are to be used for authentication.

When IKE receives a signature from a remote unit, it needs to be able to retrieve the correct public key so that it can decrypt the signature, and confirm that the signature is correct. The certificate must either be on the FLASH file system, or be provided by the remote unit as part of the IKE negotiation. The ID provided by the remote unit is used to find the correct certificate to use. If the correct certificate is found, the code then checks that it has been signed by one of the certificate authority certificates (CA*.PEM) that exist on the unit. The code first checks the local certificates, and then the certificate provided by the remote (if any). IKE will send a certificate during negotiations if it is able to find one that has subject "AltName" that matches the ID being used. If not able to locate the certificate, then the remote must have local access to the file so that the public key can be retrieved.

A typical set-up may be that the host unit has a copy of all certificates. This means that the remote units only require the private key, and the certificate authority certificate. This eases administration as any changes to certificates need only be made on the host. Because they do not have a copy of their certificate, remote units rely on the host having a copy of the certificate. An alternative is that the remote units all have a copy of the certificate, as well as the private key and certificate authority certificate, and the host only has its own certificate. This scenario requires that the remote unit send its certificate during negotiations. It can validate the certificate because it has the certificate authority certificate.

FIREWALL SCRIPTS

Introduction

A "firewall" is a protection system designed to prevent access to your local area network by unauthorised "external" parties, i.e. other users of the internet or another wide area network. It may also limit the degree of access local users have to external network resources. A firewall does not provide a complete security solution; it provides only one element of a fully secure system. Consideration should also be given to the use of user authentication and data encryption. Refer to the IPSec section for further information.

In simple terms, a firewall is a packet filtering system that allows or prevents the transmission of data (in either direction) based on a set of rules. These rules can allow filtering based on the following criteria:

- source and destination IP addresses
- source and destination IP port or port ranges
- type of protocol in use
- direction of the data (in or out)
- interface type
- the eroute the packet is on
- if an interface is OOS (out of service)
- ICMP message type
- TCP flags (SYN, ACK, URG, RESET, PUSH, FIN)
- TOS field
- status of a link and/or data packets on UDP/TCP and ICMP protocols

In addition to providing comprehensive filtering facilities, Digi routers also allow you to specify rules relating to the logging of information for audit/debugging purposes. This information can be logged to a pseudo-file on the unit called FWLOG.TXT, the EVENTLOG.TXT pseudo-file or to a syslog server. It can also be used to generate SNMP traps.

Firewall Script Syntax

A firewall must be individually configured to match the needs of authorised users and their applications. On Digi routers the rules governing firewall behaviour are defined in a script file called FW.TXT. Each line in this file consists of a label definition, a comment or a filter rule.

Labels

A label definition is a string of up to 12 characters followed by a colon. Labels can only include letters, digits and the underscore character and are used in conjunction with the break option to cause the processing of the script to jump to a new location.

Comments

Any line starting with the hash character ("#") is deemed to be a comment and ignored.

Filter Rules

The syntax for a filter rule is:

```
[action] [in-out] [options] [tos] [proto] [dnslist] [ip-range] [inspect-state]
```

When the firewall is active, the script is processed one line at a time as each packet is received or transmitted. Even when a packet matches a filter-rule, processing still continues and all the other filter rules are checked until the end of the script is reached. The action taken with respect to a particular packet is that specified by the last matching rule. With the break option however the script processing can be redirected to a new location or to the end of the script if required. The default action that the firewall assigns to a packet is to block. This means that if the packet does not match any of the rules it will be blocked.

The various fields of a script rule are described below:

[action]

The `[action]` field may be specified as `block`, `pass`, `pass-ifup`, `dscp`, `vdscp` or `debug`. These operate as follows:

block

The `block` action prevents a packet from being allowed through the firewall. When block is specified an optional field can be included that will cause an ICMP packet to be returned to the interface from which that packet was received. This technique is sometimes used to confuse hackers by having different responses to different packets or for fooling an attacker into thinking a service is not present on a network.

The syntax for specifying the return of an ICMP packet is:

```
"return-icmp" [icmp-type [icmp-code]]
```

where `[icmp_type]` is a decimal number representing the ICMP type or can be one of the predefined text codes listed in the following table:

ICMP type value	ICMP type
1	Unreach
2	Echo
3	Echorep
4	squench
5	redir
6	timex
7	paraprob
8	timest
9	timestrap
10	inforeq
11	inforep
12	maskreg
13	maskrep
14	routerad

ICMP type value	ICMP type
15	routersol

The optional `[icmp-code]` field can also be a decimal number representing the ICMP code of the return ICMP packet but if the `[icmp-type]` is `[unreach]` then the code can also be one of the following pre-defined text codes:

ICMP code	Meaning
net-unr	Network unreachable
host-unr	Host unreachable
proto-unr	Protocol unrecognised
port-unr	Port unreachable
needfrag	Needs fragmentation
srcfail	Source route fail

For example:

```
block return-icmp unreach in break end on ppp 0
```

This rule would cause the unit to return an ICMP Unreachable packet in response to all packets received on PPP 0.

Instead of using the `return-icmp` option to return an ICMP packet, `return-rst` can be used to return a TCP reset packet instead. This would only be applicable for a TCP packet. For example:

```
block return-rst in break end on eth 0 proto tcp from any to 10.1.2.0/24
```

This would return a TCP reset packet when the firewall receives a TCP packet on the Ethernet interface 0 with destination address 10.1.2.*.

pass

The pass action allows packets that match the rule to pass through the firewall.

pass-ifup

The `pass-ifup` action allows outbound packets that match the rule to pass through the firewall but only if the link is already active.

debug

The `debug` action causes the unit to tag any packets matching the rule for debug. This means that for every matching rule that is encountered from this point in the script onwards, an entry will be placed in the pseudo-file FWLOG.TXT.

dscp

The `dscp` action causes any packets matching this rule to have its DSCP value adjusted according to this rule. The DSCP value of a packet indicates the type of service required and is used in conjunction with QOS (Quality of Service) functions. A decimal or hex number must follow the `dscp` keyword to indicate the value that should be set.

vdscp

The vdscp action is very similar to the dscp action as described above in that it adjusts the DSCP value in a packet. The difference however is that this is a virtual change only which means that the actual packet is not changed, and that the packet is processed as if it had the DSCP value as indicated. Like the dscp action, a decimal or hex number must follow.

[in-out]

The [in-out] field can be in or out and is used to specify whether the action applies to inbound or outbound packets. When the field is left blank the rule is applied to any packet irrespective of its direction.

[options]

The [options] field is used to define a number of options that may be applied to packets matching the rule. These are:

log

When the log option is specified, the unit will place an entry in the FWLOG.TXT file each time it processes a packet that matches the rule. This log will normally detail the rule that was matched along with a summary of the packet contents. If the log option is followed by the body sub-option, the complete IP packet is entered into the log file so that when the log file is displayed, a more detailed decode of the IP packet is shown.

The log field may also be followed by a further sub-option that specifies a different type of log output. This may either be snmp, syslog or event.

If snmp is specified an SNMP trap (containing similar information to the normal log entry), is generated when a packet matches the rule.

If syslog is specified, a syslog message is sent to the configured syslog manager IP address. This message will contain the same information as that entered into the log file, but in a different format.

If the body option has also been specified, some of the IP packet information is also included.

Note that the size of the syslog message is limited to the maximum of 1024 bytes. The syslog message is sent with default priority value of 14, which expands out to facility of USER, and priority INFO.

If event is specified the log output will be copied to the EVENTLOG.TXT pseudo-file as well as the FWLOG.TXT file. The event log entry will contain the line number and hit count for the rule that caused the packet to be logged.

Example:

Say your local network is on subnet 192.168.*.* and you want to block any packets received on PPP 0 that were "pretending" to be on the local network and log the receipt of any such packets to the FWLOG.TXT file and to a syslog server. The filter rule would be constructed as follows:

```
block in log syslog break end on ppp 0 from 192.168.0.0/16 to any
```

break

When the break option is specified it must be followed by a user-defined label name or the predefined end keyword. When followed by a label, the rule processor will "jump" to that label to continue processing. When followed by the end keyword rule processing will be terminated and the packet will be treated according to the last matching rule.

Example:

```
break ppp_label on ppp 0
# insert rule processing here for packets that are not on ppp 0
break end
ppp_label
# insert rule processing here for packets that are on ppp 0
```

on

The on option is used to specify the interface to which the rule applies and must be followed by a valid interface name. For example, if you were only interested in applying a particular rule to packets being transmitted or received by PPP 0, you would include on ppp 0 in the rule. Valid interface-names are either eth n, tun n or ppp n, where n is the instance number.

oneroute

The oneroute option is used to specify that a rule will only match packets associated with the specified eroute. For example, including the option oneroute 2 would cause the rule to only match on packets transmitted or received over Eroute 2. The oneroute option can be followed with the keyword any, which will match if the packet is on any eroute.

routeto

When the routeto option is specified and the firewall is processing a received packet, if the rule is the last matching rule, then the packet is tagged as being required to be routed to the specified interface.

For example:

```
pass in break end routeto eth 1 from 10.1.0.0/16 to 1.2.3.4 port=telnet
```

would ensure that all packets from 10.1.*.* to 1.2.3.4 on the telnet port are all routed to ETH 1.

oosed

The oosed option is used to check the out of service status of an interface. For example, including the option oosed ppp 1 would cause the rule to match only if interface PPP 1 is out of service.

[tos]

The [tos] field may be used to specify the Type of Service (TOS) to match. If included, the [tos] field consists of the keyword tos followed by a decimal or hexadecimal code identifying the TOS to match. For example, to block any inbound packet on PPP 0 with a TOS of 0 you would use a rule such as:

```
block in on ppp 0 tos 0
```

[proto]

The [proto] field is used to specify a protocol to match and consists of the proto keyword followed by one of the following protocol identifiers:

Identifier	Meaning
tcp	udp TCP or UDP packet
udp	UDP packet

Identifier	Meaning
tcp	TCP packet
ftp	FTP packets regardless of port number
icmp	ICMP packet
decimal number	decimal number matched to protocol type in IP header

The `[proto]` field is also important when "stateful" inspection is enabled for a rule (using the `[inspect-state]` field), as it describes the protocol to inspect (see `[inspect-state]` below).

[dnslist]

The `[dnslist]` field is used to match packets that contain DNS names that are in a given dnslist. Following `dnslist` there needs to be a name of a dnslist as specified by the `#dns` command. For example, say we have the following dnslist.

```
#dns gglist www.Digi.co.*,www.*.co.nz
```

Then the following firewall rule will block all dns lockups to DNS names matching the above list.

```
block out break end on ppp 1 proto udp dnslist gglist from any to any port=dns
```

[ip-range]

The `[ip-range]` field is used to describe the range of IP addresses and ports to match upon and may be specified in one of several ways. The basic syntax is:

```
ip-range = "all" | "from" ip-object "to" ip-object [flags] [icmp]
```

where `ip-object` is an IP address specification. Full details of the syntax with examples are given under the heading "Specifying IP Addresses and Address Ranges" below.

[inspect-state]

The `[inspect-state]` field is used in create rules for "stateful inspection". This is a powerful option in which the firewall script includes rules that allow the unit to keep track of a TCP/UDP or ICMP session and therefore to only pass packets that match the state of a connection.

Additionally, the `[inspect state]` field can specify an optional OOS (Out Of Service) parameter. This parameter allows the unit to mark any route as being out-of-service for a given period of time in the event that the stateful inspect engine has detected an error.

A full description of how the `[inspect state]` field works is given below under the heading "Stateful Inspection".

Specifying IP Addresses and Ranges

The `ip-range` field of a firewall script rule identifies the IP address or range of addresses to which the rule applies. The syntax for specifying an IP address range is:

```
ip-range = "all" | "from" ip-object "to" ip-object [ flags ] [ icmp ]
```

where:

```
ip-object = addr [port-comp | port-range]
flags = "flags" { flags } [ !{ flags } ]
icmp = "icmp-type" icmp-type [ "code" decnum ]
```



```
addr = "any" | ip-addr[ "/"decnum ] [ "mask" ip-addr | "mask" hexnum ]
port-comp = "port" compare port-num
port-range = "port" port-num "<>" | "><" port-num
ip-addr = IP address in format nnn.nnn.nnn.nnn
decnum = a decimal number
hexnum = a hexadecimal number
compare = "=" | "!=" | "<" | "<=" | ">" | ">="
port-num = service-name | decnum
service-name = "http" | "telnet" | "ftpdat" | "ftpcnt" | "pop3" | "ike" | "xot"|
"sntp" | "smtp"
```

In the above syntax definition:

- items in quotes are keywords
- items in square brackets are optional
- items in curly braces are optional and can be repeated
- the vertical bar symbol ("|") means "or"

An `ip-object` therefore consists of an IP address and an IP port specification, preceded by the keyword `from` or `to` to define whether it is the source or destination address. The most basic form for an `ip-object` is simply an IP address preceded by `from` or `to`. For example, to block all packets destined for address 10.1.2.98 the script rule would be:

```
block out from any to 10.1.2.98
```

An `ip-object` can also be specified using an address mask. This is a way of describing which bits of the IP address are relevant when matching. The script processor supports two formats for specifying masks.

Method 1: The IP address is followed by a forward slash and a decimal number. The decimal number specifies the number of significant bits in the IP address. For example, if you wanted to block all packets in the range 10.1.2.* the rule would be:

```
block from any to 10.1.2.0/24
```

i.e. only the first 24 bits of the address are significant.

Method 2: This same rule could be described another way using the mask keyword:

```
block from any to 10.1.2.0 mask 255.255.255.0
```

The IP address can also contain either "`addr-ppp n`" or "`addr-eth n`" where "`n`" is the eth or ppp instance number. In this case the rule is specifying that the IP address is that allocated to the PPP interface or to the Ethernet interface. This is useful in the situation were IP addresses are obtained automatically and therefore are not known by the author of the filtering rules. For example:

```
block in break end on ppp 0 from addr-eth 0 to any
```

Address/Port Translation

One further option that may be used when specifying addresses is to use address translation. The syntax for this is:

```
srcdst = "all | fromto [-> [ip-object] "to" object]
```

I.e. directly after the IP addresses and port are specified an optional "->" can follow indicating that the addresses/ports should be translated. The first source object is optional and is unlikely to be used as it is more normal to translate the destination address. The following example will reroute packets originally destined for 10.10.10.12 to 10.1.2.3:

```
pass out break end from any to 10.10.10.12 -> to 10.1.2.3
```

Additionally to this complete subnets can have NAT applied, the address bits not covered by the subnet mask are taken from the original IP address, so for example to NAT the destination subnet of 192.168.0.0/24 to be 192.168.1.0/24 the firewall rule is:

```
pass out break end from any to 192.168.0.0/24 -> to 192.168.1.0/24
```

Filtering on Port Numbers

Now let us say there is a Telnet server running on a machine on IP address 10.1.2.63 and you wish to make this accessible. Using the filter from the previous example would block all packets to 10.1.2.*. To make the Telnet server available on 10.1.2.63 we need to add the following line in front of the blocking rule:

```
pass break end from any to 10.1.2.63 port=23
```

So, a packet being sent to the Telnet server (port 23) on IP address 10.1.2.63 will match this rule and further checking is prevented by the break end option.

The above example illustrates the "=" comparison. Other comparison methods supported are:

Symbol	Meaning
!=	not equal
>	greater than
<	less than
<=	less than or equal to
>=	greater than or equal to

It is also possible to specify a port in range or a port out of range with the "><" or "<>" symbols. For example, to pass all packets to addresses in the range 23 to 28, the rule would be specified as:

```
pass break end from any to 10.1.2.63 port 23><28
```

To simplify references to ports, some commonly used port numbers are associated with the predefined strings listed in the table below. For instance, in the example above we could substitute the number 23 with the string telnet. This would make the rule:

```
pass break end from any to 10.1.2.63 port=telnet
```

The other port keywords that are defined are:

Keyword	Std. Port	Service
Ftpdat	20	File Transfer Protocol data port
Ftpcnt	21	File Transfer Protocol control port

Keyword	Std. Port	Service
telnet	23	Telnet server port
smtp	25	SMTP server port
http	80	Web server port
pop3	110	Mail server port
sntp	123	NTP server port
ike	500	Source/destination port for IKE key
xot	1998	Destination port for XOT packets

Note:
The above service keywords are pre-defined based on "standard" port numbers. It is possible that these may have been defined differently on your system in which case you should use the port numbers explicitly (not the defined names).

Filtering on TCP Flags

An `ip-object` can be followed by an optional `[flags]` field. This field allows the script to filter based on any combination of TCP flags. The `[flags]` field is used to specify the flags to check and consists of the flags keyword followed by a string specifying the flags themselves. Each letter in this string represents a particular flag type as listed below:

Code	Flag
f	FIN Flag
r	RESET Flag
s	SYN Flag
p	PUSH Flag
u	URG Flag
a	ACK Flag

These flag codes allow the filter to check any combination of flags.

Following on from the previous example, to block packets that have all the flags set you would need to precede the pass rule with the following block rule:

```
block break end from any to 10.1.2.0/24 port=telnet flags frspua
```

Here, the list of flags causes the unit to check that those flags are set. This list may be optionally followed by an exclamation mark ("!") and a second list of flags that the unit should check for being clear.

For example:

```
flags s!a
```

would test for the `s` flag being on and the `a` flag being off with all other flags ignored.

As a further example, let us say we want to allow outward connections from a machine on 10.1.2.33 to a Telnet server. We have to define a filter rule to pass outbound connections and the inbound response packets. Because this is an outbound Telnet service we can make use of the fact that all incoming packets will have their ACK bits set. Only the first packet establishing the connection will have the ACK bit off. The filter rules to do this would look like this:

```
pass out break end from 10.1.2.33 port>1023 to any port=telnet
pass in break end from any port=telnet to 10.1.2.33 port>1023 flags !a
```

The first rule allows the outward connections, and the second rule allows the response packets back in which the ACK flag must always be on. This second rule will filter out any packets that do not have the ACK flag on. This will bar any attackers from trying to open connections onto the private network by simply specifying the source port as the Telnet port (note that there is a simpler way to achieve the same effect using the inspect state option described below).

Filtering on ICMP Codes

An `ip-object` can be followed by an optional `[icmp]` field. This allows the script to filter packets based on ICMP codes. ICMP packets are normally used to debug and diagnose a network and can be extremely useful. However they form part of a low-level protocol and are frequently exploited by hackers for attacking networks. For this reason most network administrators will want to restrict the use of ICMP packets.

The syntax for including ICMP filtering is:

```
icmp = "icmp-type" icmp-type ["code" decnum]
```

The `icmp-type` can be one of the pre-defined strings listed in the following table or the equivalent decimal numeric value:

ICMP Type	ICMP Value
Unreach	3
Echo	8
Echorep	0
Squench	4
Redir	5
Timex	11
Paramprob	12
Timest	13
Timestrep	14
Inforeq	15
Inforep	16
Maskreq	17
Maskrep	18
Routerad	9
Routersol	10

The following two rules are therefore equivalent:

```
pass in break end on ppp 0 proto icmp from any to 10.1.2.0/24 icmp-type 0
pass in break end on ppp 0 proto icmp from any to 10.1.2.0/24 icmp-type echorep
```

Both of these rules allow echo replies to come in from interface `ppp 0` if they are addressed to our example local network address (10.1.2.*).

In addition to having a type, ICMP packets also include an ICMP code field. The filter syntax allows for the specification of an optional code field after the ICMP type. When specified the code field must also match. The ICMP code field is specified with a decimal number.

For example, suppose we wish to allow only echo replies and ICMP unreachable type ICMP packets from interface PPP 0. Then the rules would look something like this:

```
pass in break end on ppp 0 proto icmp from any to 10.1.2.0/24 icmp-type echorep code 0
pass in break end on ppp 0 proto icmp from any to 10.1.2.0/24 icmp-type unreach code 0
block in break end on ppp 0 proto icmp
```

The first two rules in this set allow in the ICMP packets that we are willing to permit and the third rule denies all other ICMP packets in from this interface. Now if we ever expect to see echo replies in on `ppp 0` we should allow echo requests out on that interface too. To do that we would have the rule:

```
pass out break end on ppp 0 proto icmp icmp-type echo
```

Stateful Inspection

The Digi routing code stack contains a sophisticated scripted "Stateful Firewall" and "Route Inspection" engine. Stateful inspection is a powerful tool that allows the unit to keep track of a TCP/UDP or ICMP session and match packets based on the state of the connection on which they are being carried. In addition to providing sophisticated Firewall functionality the SF/RI engine also provides a number of facilities for tracking the "health" of routes, marking "dead" routes as being Out Of Service (OOS) and creating rules for the automatic status checking of routes previously marked as OOS (for use in multilevel backup/restore scenarios).

The firewall may be used to place interface into an OOS state and also control how the interfaces return to service. When an interface goes OOS, all routes configured to use that interface will have their route metric set to 16 (the maximum value), meaning that some other route with a lower metric will be selected.

When a firewall stateful inspection rule expires, a decision is made as to whether the traffic being allowed to pass by this rule completed successfully or not. For example, if the stateful rule monitors SYN and FIN packets in both directions for a TCP socket then that rule will expire successfully. However, if SYNs are seen to pass in one direction but no SYNs pass in the other direction, the stateful rule will expire and the unit will tag this as a failure.

The following conditions tag a stateful rule as a failure:

• packets have only passed in one direction

• 10 packets have passed in one direction with no return packets (for TCP the packets must also be re-transmits) All of these features depend upon the stateful inspection capabilities of the Firewall engine which are explained below.

The `[inspect]` field takes the following format:

```
inspect = ["inspect-state" {"oos" {interface-name¦logical-name} secs {t=secs}
{c=count} {d=count}} {r="ping"¦"tcp"{,secs{secs}}} {rd=x} {dt=secs}{stat}]
```

The field can be used on its own or with an optional `oos` (Out Of Service) parameter.

To understand this better let us look at a simple example in which we want to set up a filter to allow all machines on a local network with addresses in the range 10.1.2.*, to access the Internet on port 80. We will need one rule to filter the outgoing packets and another to filter the responses:

```
pass out break end on ppp 0 from 10.1.2.0/24 to any port=80
pass in break end on ppp 0 from any port=80 to 10.1.2.0/24
```

In this example, the first rule allows outgoing http requests on PPP 0 from any address matching the mask 10.1.2.* providing that the requests are on port 80 (the normal port address for HTTP requests).

The second rule allows http response packets to be received on PPP 0 providing they are on port 80 and they are addressed to an IP address matching the mask 10.1.2.*.

However, rule 2 creates a potential security "hole". The problem with filtering based on the source port is that you can trust the source port only as much as you trust the source machine. For instance an attacker could perform a port scan and provided the source port was set to 80 in each packet, it would get through this filter. Alternatively, on an already compromised system, a "Trojan horse" might be set up listening on port 80.

A more secure firewall can be defined using the "inspect-state" option. The stateful inspection system intelligently creates and manages dynamic filter rules based on the type of connection and the source/destination IP addresses. Applying this to the above example, we can redesign the script to make it both simpler and more effective as described below.

As a consequence of the fact that only the first packet in a TCP handshake will have the SYN flag set, we can use a rule that checks the SYN flag:

```
pass out break end on ppp 0 from 10.1.2.0/24 to any port=80 flags s inspect-state
block in break end on ppp 0
```

The first rule matches only the first outgoing packet because it checks the status of the s (SYN) flag and will only pass the packet if the SYN flag is set. At first glance however, it appears that the second rule blocks all inbound packets on PPP 0. Whilst this may be inherently more secure, it would also mean that users on the network would not be able to receive responses to their HTTP requests and would therefore be of little use!

The reason that this is not a problem is that the stateful inspection system creates temporary filter rules based on the outbound traffic. The first of these temporary rules allows the first response packet to pass because it also will have the SYN flag set. However, once the connection is established, a second temporary rule is created that passes inbound or outbound packets if the IP address and port number match those of the initial rule but does not check the SYN flag. It does however monitor the FIN flag so that the system can tell when the connection has been terminated. Once an outbound packet with the FIN flag has been detected along with a FIN/ACK response, the temporary rule ceases to exist and further packets on that IP address/port are blocked.

In the above example, if a local user on address 10.1.2.34 issues an http request to a host on 100.12.2.9, the outward packet would match and be passed. At the same time a temporary filter rule is automatically created by the firewall that will pass inbound packets from IP address 100.12.2.9 that are addressed to 10.2.1.34 port x (where x is the source port used in the original request from 10.1.2.34).

This use of dynamic filters is more secure because both the source and destination IP addresses/ports are checked. In addition, the firewall will automatically check that the correct flags are being used for each stage of the communication.

The potential for a security breach has now been virtually eliminated because even if a hacker could time his attack perfectly he would still have to forge a response packet using the correct source address and port (which was randomly created by the sender of the HTTP request) and also has to target the specific IP address that opened the connection.

Another advantage of "`inspect-state`" rules is that they are scalable, i.e. many machines can use the rule simultaneously. In our above example for instance many machines on the local network could all browse the Internet and the inspection engine would be dynamically creating precise inward filters as they are required and closing them when they are finished with.

The `inspect-state` option can be used on TCP, UDP protocols and some ICMP packets. The ICMP types that can be used with the "`inspect-state`" option are "`echo`", "`timest`", "`inforeq`" and "`maskreq`".

Using [inspect-state] with Flags

As can be seen above, the `inspect-state` option can be used with flags. To illustrate this we will refer back to the earlier example of filtering using flags. It is possible to simplify the script by using the `inspect-state` option. The original script was:

```
pass out break end from 10.1.2.33 port>1023 to any port=telnet
pass in break end from any port=telnet to 10.1.2.33 port>1023 flags a!
```

Using the inspect state option this can be replaced with a single filter rule:

```
pass out break end from 10.1.2.33 port>1023 to any port=telnet flags s!a inspect-state
```

No rule is needed for the return packets because a temporary filter will be created that will only allow inbound packets to pass if they match sessions set up by this stateful inspection rule.

A further point to note about the new rule is that the "`flags s!a`" specification ensures that it only matches the first packet in a connection. This is because the first packet in a TCP connection has the SYN flag on and the ACK flag off and so we only match on that combination. The stateful inspection engine will take care of matching the rest of the packets for this connection.

Using [inspect-state] with ICMP

The `[inspect-state]` option can be also used with ICMP codes. To allow the use of echo request and to allow echo replies you would have just the one rule:

```
pass out break end on ppp 0 proto icmp icmp-type echo inspect-state
```

The advantage of using `inspect-state`, other than just needing one rule, is that it leads to a more secure firewall. For instance with the `inspect-state` option the echo replies are not allowed in all the time; they will only be allowed in once an echo request has been sent out on that interface. The moment that a valid echo reply comes back (or there is a timeout), echo replies will again be blocked. Furthermore, the full IP address is checked; the IP source and destination must exactly match the IP destination and source of the echo request. If you compare this to the rule to allow echo replies in without using `inspect-state` it would not be possible to check the source address at all and the destination address would match any IP address on our network.

The `inspect-state` option can be used with the following ICMP packet types:

ICMP Type	Matching ICMP Type
Echo	Echo reply
Timest	Timestrep
Inforeq	Inforep
Maskreq	Maskrep

Using [inspect-state] with the Out Of Service Option

The `inspect-state` field can be used with an optional `oos` parameter. This parameter allows the stateful inspect engine to mark as "out of service" any routes that are associated with the specified interface and also to control how and the interfaces are returned to service. Such routes will only be marked as out of service if the specified oos option parameters are met. The `oos` parameter takes the format:

```
oos {interface-name¦logical-name} secs {t=secs} {c=count} {d=count}
{r="ping"|"tcp"{,secs}}
```

where:

`interface-name` or `logical-name` specifies the interface with which the firewall rule is associated, e.g. PPP 1. This can also be a logical interface name which is simply a name that can be created (e.g. "waffle"). When a logical interface name is specified then this name can become oos (out of service) and can be tested in other firewall rules with the `oosed` keyword.

`secs` specifies the length of time in seconds for which the routes that are using the specified interface are marked as out of service.

`{t=secs}` is an optional parameter that specifies the length of time in seconds the unit will wait for a response the packet that matched the rule.

`{c=count}` is an optional parameter that specifies the number of times that the stateful inspection engine must trigger on the rule before the route is marked as out of service.

`{d=count}` is an optional parameter that specifies the number of times that the stateful inspection engine must trigger on the rule before the interface is deactivated (only applies to PPP interfaces).

`{r="ping"|"tcp"{,secs{,secs}}}` is an optional parameter that specifies a recovery procedure. When a recovery procedure is specified then after the oos timeout has expired instead of bringing the interface back into service immediately the link is tested first. It is tested by either sending a TCP SYN packet or a ping packet to the address/port that caused the oos condition. The "secs" field specifies the retry time when checking for recovery. Only when the recovery succeeds will interface become in service again.

UDP Example

```
pass in

pass out

pass out on ppp 1 proto udp from any to 156.15.0.0/16 port=1234 inspect-state oos ppp
1 300 t=10 c=2 d=2
```

The first two rules simply configure the unit to allow any type of packets to be transmitted or received (the default action of the firewall is to block all traffic).

The third rule is more complex. What it does is to configure the stateful inspection engine to watch for UDP packets (with any source address) being routed via the PPP 1 interface to any address that begins with 156.15 on port 1234. If a hit occurs on this rule but the unit does not detect a reply within 10 seconds (as specified by the `t=` parameter), it will increment an internal counter. When this counter reaches the value set by the `c=` parameter, the stateful inspection engine will mark the PPP 1 interface (and therefore any routes using it), as being out of service for 300 seconds. Similarly, if this counter matches the `d=` parameter the stateful inspection engine will deactivate PPP 1. So in the above example, the stateful inspection engine will mark any routes that use PPP 1 as out of service AND deactivate PPP 1 if no reply is detected within 10 seconds for two packets in a row.

Routes will come back into service when either the specified timeout expires or if there are no other routes with a higher metric in service.

PPP interfaces will be re-activated when either the routes using them are back in service and there is a packet to route and the AODI mode parameter is set to "On".

TCP Example

```
pass out log break end on ppp 3 proto tcp from any to 192.168.0.1 flags S!A inspect-
state oos 30 t=10 c=2 d=2

pass in

pass out
```

This rule will specifically trace attempts to open a TCP connection on PPP 3 to the 192.168.0.1 IP address and if it fails within 10 seconds twice in a row, will cause the PPP 3 interface to be flagged as out of service (i.e. its metric will be set to 16), for 30 seconds. The optional `d=2` entry will also cause the PPP link to be deactivated. Deactivating the link can be useful in scenarios where renegotiating the PPP connection is likely to resolve the problem. Again, if a matching route with a higher metric has been defined it will be used whilst PPP 3 routes are out of service thus providing a powerful route backup mechanism.

Using [inspect-state] with the Stat Option

The `inspect-state` option can be used with the `stat` option. The `stat` option will cause this firewall rule to record statistics associated with this firewall rule. Transaction times, counts and errors are recorded under the PPP statistics with this option.

Assigning DSCP Values

When using QOS, packet priorities will be determined by the DSCP values in their TOS fields. These priorities may have already been assigned but if necessary, the router can be configured to assign them by inserting the appropriate rules in the firewall. This is done by using the `dscp` command.

For example:

```
dscp 46 in on eth 0 from 100.100.100.25 to 1.2.3.4 port=4000
```

would set the DSCP value to 46 for almost any type of packet received on ETH 0 from IP address 100.100.100.25 addressed to 1.2.3.4 on port 4000. This allows you to set the DSCP value for almost any type of packet.

As a further example:

```
dscp 46 in on eth 0 proto smtp from any to any
```

would cause outgoing mail traffic to the same top priority queue (46 is by default a very high priority code in the DSCP mappings).

The FWLOG.TXT File

When the log option is specified within a firewall script rule, an entry is created in the FWLOG.TXT pseudo-file each time an IP packet matches the rule. Each log entry will in turn contain the following information:

Parameter	Description
Timestamp	The time when the log entry is created.
Short Description	Usually "FW LOG" but could be "FW DEBUG" for packets that hit rules with the "debug" action set.
Dir	Either "IN" or "OUT". Indicates the direction the packet is travelling.
Line	The line number of the rule that cause the packet to be logged.
Hits	The number of matches for the rule that caused this packet to be logged.
Iface	The Interface the packet was to be transmitted/received on.
Source IP	The source IP address in the IP packet.
Dest. IP	The destination IP address in the IP packet.
ID	The value of the ID field in the IP packet.
TTL	The value of the TTL field in the IP packet.
PROTO	The value of the protocol field in the IP packet. This will be expanded to text as well for the well-known protocols.
Src Port	The value of the source port field in the TCP/UDP header.
Dst Port	The value of the source port field in the TCP/UDP header.
Rule Text	The rule that caused the packet to be logged is also entered into the log file.

In addition, port numbers will be expanded to text pre-defined port numbers.

Log File Examples

Example: log entry **without** the `body` option:

```
----- 15-8-2002 16:25:50 ------
FW LOG Dir: IN Line: 11 Hits: 1 IFACE: ETH 0
Source IP: 100.100.100.25 Dest IP: 100.100.100.50 ID: 39311 TTL: 128
PROTO: TCP (6)
Src Port: 4232 Dst Port: WEB (80)
pass in log break end on eth 0 proto tcp from 100.100.100.25 to addr-
eth 0
flags S/SA inspect-state
----------
```

Example: Log entry **with** the `body` option:

```
----- 15-8-2002 16:27:56 ------
FW LOG Dir: IN Line: 7 Hits: 1 IFACE: ETH 0
Source IP: 100.100.100.25 Dest IP: 100.100.100.50 ID: 40140 TTL: 128
PROTO: ICMP (1)
```



```
block return-icmp echorep log body break end proto icmp icmp-type echo
From REM TO LOCIFACE: ETH 0
45 IP Ver: 4
Hdr Len: 20
00 TOS: Routine
Delay: Normal
Throughput: Normal
Reliability: Normal
00 3C Length: 60
9C CC ID: 40140
00 00 Frag Offset: 0
Congestion: Normal
May Fragment
Last Fragment
80 TTL: 128
01 Proto: ICMP
0C E1 Checksum: 3297
64 64 64 19 Src IP: 100.100.100.25
64 64 64 32 Dst IP: 100.100.100.50
ICMP:
08 Type: ECHO REQ
00 Code: 0
04 5C Checksum: 1116
----------
```

Example: Text included in the EVENTLOG.TXT pseudo-file when the `event` sub-option is specified:

```
16:26:32, 15 Aug 2002,Firewall Log Event: Line: 10, Hits: 3
```

Example: Syslog message where the `body` option is **not** specified:

```
2002-09-04 16:30:06 User.Info100.100.100.50Aug 15 16:31:59 arm.1140
IP Filter -
Filter Rule: block return-icmp unreach host-unr in log syslog break
end on eth 0 proto tcp from any to 100.100.100.50 port=telnet
Line: 10
Hits: 4
```

Example: Syslog message with the `body` option **is** specified:

```
2002-08-30 16:19:59 User.Info100.100.100.50Aug 10 16:21:56 arm.1140
IP Filter - Filter Rule: block return-icmp unreach port-unr in log
body syslog break end on eth 0 proto tcp from any to 100.100.100.50
port=telnet
Line: 9
Hits: 3
PKT:
Source IP: 100.100.100.25
Dest IP: 100.100.100.50
ID: 13317
TTL: 128
Protocol: TCP
Source Port: 1441
```

```
Dest Port: 23
TCP Flags: S
```

Further [inspect-state] Examples

Here is a basic `inspect-state` rule with no OOS options:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state
```

This rule will allow TCP packets from 10.1.1.1 to 10.1.2.1 port 23 with the SYN flag set to pass out on PPP 2. Because the `inspect-state` option is used, a stateful rule will also be set up which allows other packets for that TCP socket to also pass.

Next, we will modify the rule to mark an interface OOS if a stateful rule identifies a failed connection:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60
```

The addition of `oos 60` means that if the stateful rule sees a failure, interface PPP 2 will be set OOS for 60 seconds. If no interface is specified after the `oos` keyword, the interface set to OOS will be the one the packet is currently passing on. It is possible to OOS a different interface by specifying the interface after the `oos` keyword, e.g. `oos ppp 1 60` to put PPP 1 out of service for 60 seconds.

The default time allowed by the stateful rule for a connection to open may be overridden by using the `{t=secs}` option. E.g. To override the default TCP opening time of 60 seconds to 10 seconds:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60 t=10
```

A socket will now only have 10 seconds to become established (i.e. exchange SYNs) before the stateful rule will expire and be tagged as a failure.

It is possible to configure the firewall so that the interface is only set to OOS after a number of consecutive failures occur. To do this, use the `{c=count}` option. For example:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60 t=10 c=5
```

PPP 2 will now only be set OOS after 5 consecutive failures.

It is possible to deactivate the interface after a number of consecutive failures. This is useful for WWAN interfaces, which may get into a state where the PPP connection appears to be operational, but in fact no packets are passing. In this case, deactivating and reactivating the interface will sometimes fix the problem.

For example:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60 t=10 c=5 d=10
```

Now, PPP 2 will be deactivated after 10 consecutive failures.

Keeping a route out of service and using recovery

It may be that the user wants to keep the interface OOS until he is sure that a future connection will work. To help achieve this, one or more recovery options may be specified. These options get the unit to test connectivity between the unit and the destination IP address of the packet that established the stateful rule. The recovery can be in the form of a PING or a TCP socket connection. An interval between recovery checks must also be specified. For example:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60 t=10 c=5 d=10 r=tcp,120
```

Now the interface will be set to OOS for 60 seconds after 5 consecutive failures. After the 60 seconds elapses, the recovery procedure will be initiated. In this example the recovery will consist of TCP connection attempts executed at 2 minute intervals. The interface will remain OOS until the recovery procedure completes successfully. The destination IP address in this case will be 10.1.2.1.

To override the default socket connection time, it is possible to specify an additional recovery option. For example:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60 t=10 c=5 d=10 r=tcp,120,10
```

Now, 10 seconds is allowed for each recovery attempt. If the socket connects within that time, the recovery is successful, else the recovery is unsuccessful.

There is also an option `{rd=x}` to disconnect the interface after a recovery attempt completes. This option can be used to deactivate the interface after a recovery failure, success, or either. "`x`" is a bitmask indicating the cases where the interface should be deactivated. Bit 0 is used to deactivate the interface after a recovery failure. Bit one is used to deactivate the interface after a recovery success, i.e.

- rd=1 – means deactivate after a recovery failure
- rd=2 – means deactivate after a recovery success
- rd=3 – means deactivate after either recovery success or recovery failure

Extending our firewall rule to include this option gives:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60 t=10 c=5 d=10 r=tcp,120,10 rd=3
```

Now the interface will be deactivated after a recovery success or failure.

If the `{rd=x}` option is not used, the interface will remain up until its inactivity timer expires, or it is deactivated by some other means.

The `{dt=secs}` option may be used to indicate that the interface is to remain OOS when it is disconnected, and that it should be reactivated some time after it last disconnected. Recovery procedures will take place after the interface connects.

Extending our firewall rule to include this option gives:

```
pass out break end on PPP 2 proto TCP from 10.1.1.1 to 10.1.2.1 port=telnet flags S!A
inspect-state oos 60 t=10 c=5 d=10 r=tcp,120,10 rd=3 dt=60
```

Now the interface will be reconnected 60 seconds after it disconnects and recovery procedures will start after the interface connects. This option would normally be used with the `{rd=x}` option so that recovery has control over when the interface connects and disconnects.

Keeping a route out of service and using recovery with a list of addresses

This expands on the functionality above and gives the ability to check connectivity to a range of addresses using a ping. It is possible to specify an address list that the recovery mechanism will ping in turn to see if any respond. This will help ensure that even when 1 or maybe 2 or 3 destinations cant be reached due to an outage on the remote network, the connection will be made available again if at least one of the addresses in the list responds.

The address lists are created using the following syntax:

```
#addrs <list-name> <address1,address2,address3,address4>
```

Address lists can span multiple lines if required, for example:

```
#addrs <list-name> <address1,address2>
```

```
#addrs <list-name> <address3,address4>
```

The address list is called using the recovery option pingl. An example firewall rule would be:

```
pass out break end on PPP 1 proto ICMP from 10.1.1.1 to 10.1.2.1 inspect-state oos 60
t=10 c=5 d=10 r=pingl listA ,120,10 rd=3 dt=60
```

This rule would allow pings outbound and on detecting a communication failure it will use pings to a address list named listA. The address list named listA could look like this:

```
#addrs listA 10.1.2.1,10.1.3.1,10.1.4.1,10.1.5.1
```

```
#addrs listA 10.1.6.1,10.2.1.1,10.2.2.1
```

This causes the recovery to ping the range of address shown in the list above.

Debugging a Firewall

During the creation and management of firewall scripts, firewall scripts may need debugging to ensure that packets are being processed correctly. To assist in this, a rule with the debug action may be used.

If a rule with the `debug` action is encountered, an entry is made in the FWLOG.TXT pseudo-file each time the packet in question matches a rule from that point on. This gives the administrator the ability to follow a packet through a rule set, and can help determine what, if any, changes are required to the rule set. Rules that specify the `debug` action would typically be placed near the top of the rule set, so that all matching rules from that point on are entered into the log file.

Entries the FWLOG.TXT file created as the result of a `debug` rule may be identified by the short description "FW_DEBUG" at the top of the log entry.

An example rule set using a `debug` rule:

```
debug in on ppp 2 proto tcp from any to any port=http
```

```
pass in break end proto tcp from any to any port=http flags s/sa inspect state
```

```
pass out break end proto udp
```

If placed at the top of the rule set, any packet received on interface PPP 2 to destination port 80 will generate a debug entry in the log file for each subsequent rule that it matches. In the example rule set above, a packet that matched the second rule would also match the first rule, and would therefore create two log entries. The same packet would not match the third rule, and so no log entry would be made for this rule.

Because of the extra processor time required to add all of these additional log entries, debug rules should be removed (or commented out) once the rule set is operating as desired.

REMOTE MANAGEMENT

Digi products equipped with ISDN BRIs can be accessed and controlled remotely via the ISDN network by using:

- a V.120 connection to access the text command interface
- PPP to access the Web Interface
- PPP to access the text command interface using Telnet
- the X.25 remote command channel

Remote access via any one of these methods can be used to reconfigure the unit, upload/download files or upgrade the software, examine the event log or protocol analyser traces or to view statistics.

Using V.120

To establish a remote access session using V.120, initiate a V.120 call as normal using the ATD command. Enter "%%%" within 5 seconds of the remote unit answering and you will be prompted to enter your username and password. Correct entry of these will allow access to the text command interface. If the remote unit has been programmed with a **Router Identity** string on the ***Configuration - System > Device Identity*** page, the **Router Identity** will appear as the command line prompt. Three login attempts are permitted before the connection is reset.

Using Telnet

If you have created a PPP DUN (Dial-up Networking) entry for the remote unit that you wish to access, any terminal program that supports Telnet may be used to establish a remote connection.

To initiate the connection, launch the DUN. If the remote unit is configured correctly with one of the PPP instances enabled for answering, it will connect and the linked computers icon will appear in the Windows system tray. You may then load your Telnet software.

To configure your Telnet software you must first specify that you require a TCP/IP connection and then enter the appropriate IP address or hostname (e.g. 1.2.3.4, 192.168.1.1 or digi.router by default). After ensuring that your software is configured to connect to TCP port number 23 you may then initiate a new connection.

If the connection is successful you will see a connect confirmation message and you will be prompted to enter your username and password. Correct entry of these will allow access to the text command interface. If the remote unit has been programmed with a **Router Identity** string on the ***Configuration - System > Device Identity*** page the **Router Identity** will appear as the command line prompt.

Three login attempts are permitted before the connection is reset.

Using FTP

TransPort routers incorporate an FTP server. FTP allows users to log on to remote hosts for the purpose of inspecting file directories, retrieving or uploading files, etc. For PC users, MS-DOS includes FTP support and there are a number of Windows-based specialist FTP client programs such as CuteFTP™ and Ws_ftp™. Many browsers also incorporate FTP support.

To initiate remote access to a unit using FTP, first establish a PPP DUN connection to the unit and then run your FTP software.

Note:
If your unit has a USB storage device attached, it will show up as a sub-folder named "usb".

FTP under Windows

Once the connection has been established, enter the Web address for the unit. By default this will be:

1.2.3.4, 192.168.1.1 or digi.router

If you are using a browser, as opposed to a specific FTP program, you will need to precede the address with "ftp://". For example:

ftp://digi.router

This will give you an anonymous FTP login to the remote unit and you should see a listing of the file directory (the format of this will depend on the FTP client software that you are using). With an anonymous login you will be able to view and retrieve files, but NOT upload, rename or delete them.

For full file access, you will need to log in with your correct username and password. To do this, enter the address in the following format:

ftp://username:password@digi.router

This will give you full access and will allow you to copy, delete, rename, view and transfer files.

When using a browser CUT, COPY, DELETE and PASTE may be used for manipulating files as if they were in a normal Windows directory. If you are using a specific FTP client program, these operations may be carried out using menu options or buttons.

FTP under DOS

To use FTP under DOS, use Windows DUN to establish the connection and then run the MSDOS prompt program. At the DOS prompt type:

ftp digi.router

or

ftp 192.168.1.1

When the connection has been established you will be prompted to enter your username and password. Following a valid login the ftp> prompt will be issued and you may proceed to use the various ftp commands as appropriate. To obtain a list of available commands enter "?" at the prompt.

Using X.25

Remote access to your unit may also be carried out over an X.25 connection. The remote unit must first have the parameter **Allow CLI access from X.25 address** set to an appropriate value (see ***Configuration - System > General***). If the unit then receives an incoming X.25 call where the trailing digits of the NUA match the specified sub-address, the calling user will receive the standard login prompt. On entry of a valid username and password, they will be given access to the command line as if they were connected locally.

AT COMMANDS

D Dial

The ATD command causes the unit to initiate an ISDN call. The format of the command depends on the mode of operation.

When using the unit to make data calls on one of the ISDN B-channels, enter the ATD command followed by the telephone number. For example, to dial 01234 567890 enter the command:

```
atd01234567890
```

Spaces in the number are ignored. If the call is successful the unit will issue the CONNECT result code and switch to on-line mode.

Dialling with a Specified Sub-Address

The ATD command may also be used to route a call to an ISDN sub-address by following the telephone with the letter S and the required sub-address. The sub-address may be up to 15 digits long. For example:

atd01234567890s003

Dialling Stored Numbers

To dial numbers that have previously been stored within the unit using the AT&Z command, insert the S= modifier within the dial string. For example, to dial stored number 3 use the command:

```
atds=3
```

Combining ISDN and X.25 Calls

A further option for the ATD command for X.25 applications is to combine the ISDN call and the subsequent X.25 CALL in the same command. To do this, follow the telephone number with the "=" symbol and the X.25 call string. For example:

```
atd01234 567890=123456789
```

Pressing any key while the ATD command is being executed will abort the call attempt.

H Hang-up

The ATH command is used to terminate an ISDN call. If the unit is still on-line you must first switch back to command mode by entering the escape sequence, i.e. +++, wait 1 second and then enter an AT command or just AT<CR>.

After entering the ATH command the call will be disconnected and the NO CARRIER result will be issued.

Z Reset

The ATZ command is used to load one of the stored profiles for the active ASY port. The command is issued in the format ATZn where n is the number (0 or 1) of the ASY port profile you wish to load.

&C DCD Control

The AT&C command is used to configure the way in which the unit controls the DCD signal to the terminal. There are three options:

&C0 DCD is always On

&C1 DCD is On only when an ISDN connection has been established (Layer 2 is UP)

&C2 DCD is always Off

&C3 DCD is normally On but pulses low for a time in 10 msec units determined by S register 10.

&F Load Factory Settings

The AT&F command is used to load a pre-defined default set of S-register and AT command settings (the default profile). These are:

E1, V1, &C1, &K1, &D2, S0=0, S2=43

All other values are set to 0.

&R CTS Control

The AT&R command is used to configure the way in which the unit controls the CTS signal to the terminal. There are three options:

&R0 CTS is always On

&R1 CTS follows RTS. The delay between RTS changing and CTS changing is set in AT register 56 in multiples of 10msec

&R2 CTS is always Off

&V View Profiles

The AT&V command displays a list of the current AT command and S register values, and the settings for the two stored profiles. For example:

```
at&v
CURRENT PROFILE:
&c1 &d2 &k1 &s1 &r0 e1 q0 v1 &y0
S0=0 S2=43 S12=50 S31=3 S45=5
states DTR:1 RTS:1

STORED PROFILE 0:
&c1 &d2 &k1 &s1 &r0 e1 q0 v1
S0=0 S2=43 S12=50 S31=3 S45=5

STORED PROFILE 1:
&c1 &d2 &k1 &s1 &r0 e1 q0 v1
S0=0 S2=43 S12=50 S31=3 S45=5
OK
```

&W Write SREGS.DAT

The AT&W command is used to save the current command and S registers settings (for the active port), to the file SREGS.DAT. The settings contained in this file can be reloaded at any time using the ATZ command.

The AT&W command may be immediately followed by a profile number, either 0 or 1, to store the settings in the specified profile, for example:

```
at&w1
```

would store the current settings as profile 1. If no profile number is specified, profile 0 is assumed.

All S register values and the following command settings are written by AT&W:

```
e, &c, &d, &k
```

&Y Set Default Profile

The AT&Y command is used to select the power-up profile (0 or 1). For example, to ensure that the unit boots up using stored profile 1, enter the command:

```
at&y1
```

&Z Store Phone Number

The AT&Z command is used to store "default" telephone numbers within the unit that may subsequently dialled when DTR dialling is enabled or by using the S= modifier in the ATD dial command. One telephone number may be stored for each ASY port. For example, to store the phone number 0800 123456 as the default number to be associated with ASY 2, use the command:

```
at&z2=0800123456
```

If the number of the ASY port is not specified, the number will be stored against the port from which the command was entered, i.e. entering the command:

```
at&z=0800123456
```

from ASY 3 has the same effect as:

```
at&z3=0800123456
```

from any port. Once a number has been stored it may be dialled from the command line using the ATD command with the S= modifier:

```
atds=3
```

This means that any stored number can be dialled from any port. If DTR dialling has been enabled by setting S33=1 for the port, the number associated with that port will be dialled when the DTR signal for that port changes from Off to On, i.e. DTR dialling can only be used with the number associated with the port to which the terminal is connected.

\AT Ignore Invalid AT Commands

This command is a work-around for use with terminals that generate large amounts of extraneous text. If not ignored, this text can cause many error messages to be generated by the router, and may result in a communications failure. To turn on this feature, type the following command:

```
at\at=1
```

To turn off the feature, type the following command:

```
at\at=0
```

When this feature is turned on, the ASY port ignores all commands except real AT commands. As with other ASY modes this can be saved by AT&W but is not included in the AT&V status display. To determine whether or not this mode is enabled type:

```
at\at ?
```

The unit will display 0 if the feature is Off, 1 if it is On.

\LS Lock Speed

The AT\LS command is used to lock the speed and data format of the port at which it is entered to the current settings so that the non-AT application commands may be used.

\PORT Set Active Port

Text commands which affect the settings associated with the serial ports normally operate on the port at which they are entered, i.e. entering the AT&K command from a terminal connected to ASY 1 will affect only the flow control settings for port 1.

The AT\PORT command is used to select a different "active" port from that at which the commands are entered. For example, if your terminal is connected to port 0 and you need to reconfigure the settings for port 2, you would first enter the command:

```
at\port=2
PORT 2
OK
```

Port 2 is now the active port and any AT commands or changes to S registers settings which affect the serial ports will now be applied to port 2 only. This includes:

Commands: Z, &D, &F, &K, &V, &Y, &W

S registers: S31, S45

The AT\PORT? command will display the port to which you are connected and the active port for command/ S register settings. For example:

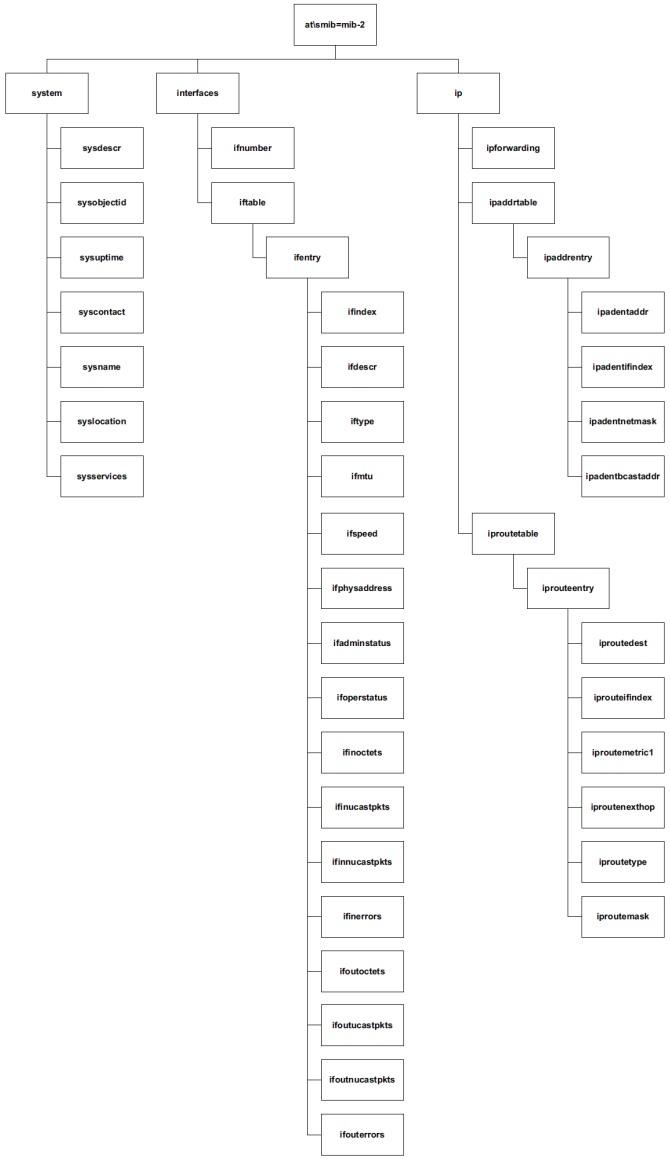
```
at\port?
PORT 2
ASY0
OK
```

Here, ASY2 is the active port and ASY0 is the port at which the command was entered. If the default port and the port to which you are connected are the same, only one entry will be listed.

To reset the default port to the one to which you are connected use the AT\PORT command without a parameter.

\smib Commands

The at\smib command allows you to view a single standard MIB variable. To view the variable use the at\smib=<mib_name> command, where <mib_name> is the variable to be displayed. The variables are sorted according to the hierarchy shown below.



System

The System hierarchy consists of the following:

at\smib=mib-2.system.sysdescr

This variable shows the software version information (equivalent to what is shown on the 'ati5' CLI command output).

```
mib-2.system.sysdescr =
Software Build Ver5121.  Jan 31 2011 12:26:04  9W
```

at\smib=mib-2.system.sysobjectid

The authoritative identification of the network management subsystem. The Digi does not support outputting OID variables. Instead, "*oid*" is output.

```
mib-2.system.sysobjectid = oid
```

at\smib=mib-2.system.sysuptime

The time the unit has been running in 10msec units (hundredths of a second).

```
mib-2.system.sysuptime = 1806718
```

The above example shows that the unit has been running for 5 hours, 1 minute and 7.18 seconds.

at\smib=mib-2.system.syscontact

A description of the contact person for the unit. For the Digi, this is always a zero-length string.

at\smib=mib-2.system.sysname

The name of the unit (the name set in the **Router Identity** parameter on the ***Configuration - System > Device Identity*** page).

```
mib-2.system.sysname = digi.router
```

at\smib=mib-2.system.syslocation

The physical location of the unit. For the Digi, this is always a zero-length string.

at\smib=mib-2.system.sysservices

This variable displays a value that represents the set of services the unit provides. For each OSI layer the unit provides services for, 2(L-1) is added to the value, where L is the layer. The layers are shown below:

Layer	Functionality
1	Physical
2	Data Link
3	Network
4	Transport
5	Session
6	Presentation
7	Application

For the Digi, this value is always 7 (Physical layer (21-1) + Data Link layer (22-1) + Network layer (23-1)).

Interfaces

The Interfaces hierarchy consists of the ifnumber variable and the iftable node:

at\smib=mib-2.interfaces.ifnumber

The total number of interfaces on the unit. This includes Ethernet, PPP and virtual interfaces (i.e. IPSec tunnels) and SYNC ports.

```
mib-2.interfaces.ifnumber = 52
```

at\smib=mib-2.interfaces.iftable

The iftable node contains ifentry nodes for each interface. For each table entry, an index specifier must be appended to the end of each variable (e.g. for PPP0, 1 must be appended).

at\smib=mib-2.interfaces.iftable.ifentry

at\smib=mib-2.interfaces.iftable.ifentry.ifindex

The unique index number of the interface.

at\smib=mib-2.interfaces.iftable.ifentry.ifdescr

This variable displays information about the interface. This information is displayed in the format *<interface type>-<instance>*, where:

<interface type> can be one of *PPP*, *ETH*, *TUN* (for IPSec tunnels), *SNAIP* (for SNAIP links) or *SYNC*, and

<instance> is the instance.

For example:

```
mib-2.interfaces.iftable.ifentry.ifdescr.1 = PPP-0
```

at\smib=mib-2.interfaces.iftable.ifentry.iftype

The type of interface, as described by the physical/link protocol below the network layer in the protocol stack. Values can be one of the following:

PPP	23
ETH	6
IPSec Tunnel	131
SNAIP	17
SYNC port	118

For example:

```
mib-2.interfaces.iftable.ifentry.iftype.1 = 23
```

at\smib=mib-2.interfaces.iftable.ifentry.ifmtu

The size of the largest datagram (in octets) which can be sent on the interface. SNAIP and SYNC ports always return 0. IPSec tunnel interfaces will return the underlying interface if it can be located, otherwise 0 is returned. PPP interfaces will return the negotiated MTU if the link is connected, otherwise 0 is returned.

For example:

```
mib-2.interfaces.iftable.ifentry.ifmtu.21 = 1504
```

at\smib=mib-2.interfaces.iftable.ifentry.ifspeed

This variable displays an estimate of the interface's current bandwidth in bits per second. SNAIP and SYNC ports will always return 0. PPP ports will always return 64000.

For example:
`mib-2.interfaces.iftable.ifentry.ifspeed.1 = 64000`

at\smib=mib-2.interfaces.iftable.ifentry.ifphysaddress

The interface's address at the protocol layer immediately below the network layer in the protocol stack. For interfaces without such an address, a zero-length octet string is returned. For PPP, SNAIP and SYNC ports, a 0 length string is returned.

at\smib=mib-2.interfaces.iftable.ifentry.ifadminstatus

The desired state of the interface. The testing state (3) indicates no operational packets can be passed.

at\smib=mib-2.interfaces.iftable.ifentry.ifoperstatus

The current operational state of the interface. The testing state (3) indicates no operational packets can be passed.

at\smib=mib-2.interfaces.iftable.ifentry.ifinoctets

The total number of octets received on this interface, including framing characters.

at\smib=mib-2.interfaces.iftable.ifentry.ifinucastpkts

The number of subnetwork-unicast packets delivered by this interface to a higher-layer protocol.

at\smib=mib-2.interfaces.iftable.ifentry.ifinnucastpkts

The number of non-unicast (i.e. broadcast or multicast) packets delivered by this interface to a higher-layer protocol.

at\smib=mib-2.interfaces.iftable.ifentry.ifinerrors

The number of inbound packets received by this interface that contained errors preventing them from being delivered to a higher-level protocol.

at\smib=mib-2.interfaces.iftable.ifentry.ifoutoctets

The total number of octets transmitted by this interface, including framing characters.

at\smib=mib-2.interfaces.iftable.ifentry.ifoutucastpkts

The total number of packets that higher-level protocols requested this interface to transmit to a subnetwork-unicast address, including those that were discarded or not sent.

at\smib=mib-2.interfaces.iftable.ifentry.ifoutnucastpkts

The total number of packets that higher-level protocols requested this interface to transmit to a non-unicast (i.e. broadcast or multicast) address, including those that were discarded or not sent.

at\smib=mib-2.interfaces.iftable.ifentry.ifouterrors

The number of outbound packets that this interface could not transmit because of errors.

IP

The IP node consists of the ipforwarding variable and the ipaddrtable and iproutetable nodes.

at\smib=mib-2.ip.ipforwarding

This variable indicates whether the unit is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, the unit. IP gateways forward datagrams, IP hosts do not. For the Digi, this value is always 1.

at\smib=mib-2.ip.ipaddrtable

The ipaddrtable node contains ipaddrentry nodes for each IP address assigned to each interface of the unit. For each table entry, an index specifier must be appended to the end of each variable that specifies the interface (e.g. for PPP0, 1 must be appended).

at\smib=mib-2.ip.ipaddrtable.ipaddrentry

at\smib=mib-2.ip.ipaddrtable.ipaddrentry.ipadentaddr

The IP address to which this entry's addressing information pertains.

at\smib=mib-2.ip.ipaddrtable.ipaddrentry.ipadentifindex

The index identifier for the interface associated with this IP address.

at\smib=mib-2.ip.ipaddrtable.ipaddrentry.ipadentnetmask

The subnet mask associated with the IP address.

at\smib=mib-2.ip.ipaddrtable.ipaddrentry.ipadentbcastaddr

The value of the least-significant bit in the IP broadcast address used for sending datagrams on the IP address of this interface.

at\smib=mib-2.ip.iproutetable

The iproutetable node contains iprouteentry nodes for each route defined on the unit.

at\smib=mib-2.ip.iproutetable.iprouteentry

at\smib=mib-2.ip.iproutetable.iprouteentry.iproutedest

The destination IP address for the route. An entry with a value of 0.0.0.0 is considered the default route. Multiple routes to a single destination can appear in the routing table, but access to such multiple entries is dependant on the table-access mechanisms defined by the network management protocol in use.

at\smib=mib-2.ip.iproutetable.iprouteentry.iprouteifindex

The index value which uniquely identifies the local interface through which the next hop of the route should be reached.

at\smib=mib-2.ip.iproutetable.iprouteentry.iproutemetric1

The primary routing metric for the route.

at\smib=mib-2.ip.iproutetable.iprouteentry.iproutenexthop

The IP address of the next hop of the route.

at\smib=mib-2.ip.iproutetable.iprouteentry.iproutetype

The type of route. Valid values are:

1	Valid
2	Invalid
3	Direct
4	Indirect

at\smib=mib-2.ip.iproutetable.iprouteentry.iproutemask

The netmask for the route.

"S" REGISTERS

In addition to the AT commands there are a number of Special ("S") registers. These registers contain numeric values that may represent time intervals, ASCII characters or operational flags.

To display the contents of a particular "S" register, the ATS command is used in the form ATSn? where n is the number of the register whose contents are to be shown.

To store a new value into a register, use the S command in the form ATSN=X where N is the number of the register to be changed and X is the new value. For example, ATS31=4 would store the value 4 in S31.

The unit maintains one set of registers for each ASY port. By default, the S command operates ONLY on the S register set for the active port. To select an alternative default port, use the AT\PORT command first.

Each register can only be set to a limited range of values as shown in the table below:

Reg.	Description	Units	Default	Range
S0	V.120 Answer enable	Rings	0	0-255
S1	Ring count	Rings	n/a	n/a
S2	Escape character	ASCII	43	0-255
S9	DCD on delay	ms x 20	0	0-255
S10	Pulse time for DCD Low	ms x 10	0	0-255
S12	Escape delay	ms	50	0-255
S15	Data forwarding timer	ms	2	0-255
S23	Parity	N/A	0	0-2 5 6
S31	ASY interface speed	refer to full description	n/a	0-11
S33	DTR dialling	N/A	0	0 1
S45	DTR loss de-bounce	0.05 seconds	(0.25s)	1-255

S0 V.120 Answer Enabled

Units: Rings
Default: 0
Range: 0-255

S0 is used only in V.120 mode to enable or disable automatic answering of incoming ISDN calls. Auto answering is disabled when S0 is set to the default value of 0. Setting S0 to a non-zero value enables auto-answering.

The actual value stored determines the number of "rings" that the unit will wait before answering. For example, the command ATS0=2 enables auto-answering after two incoming rings have been detected.

With each ring the RING result code is issued and the value stored in S1 is incremented. When the value in S1 equals the value in S0 the call is answered.

S1 Ring count

Units: Rings
Default: n/a
Range: n/a

When ADAPT detects an incoming ISDN call on an ASY port, it will print "RING" to the ASY port at 2 second intervals. It also increments the S1 register, counting how many times "RING" is printed.

S2 Escape Character

Units: ASCII
Default: 43
Range: 0-255

The value stored in S2 defines which ASCII character is used as the Escape character, which by default is the "+" symbol. Entering this character three times followed by a delay of 1-2 seconds and then an AT command will cause the unit to switch from on-line mode to command mode.

S12 Escape Delay

Units: ms
Default: 50
Range: 0-255

The value stored in S12 defines the delay between sending the escape sequence and entering an AT command for the unit to switch from on-line mode to command mode.

S15 Data Forwarding Timer

Units: 10ms
Default: 0
Range: 0-255

S15 is used to set the data forwarding timer for the ASY port in multiples of 10ms. The default data forwarding time is 20ms and in normal use this there should be no need to change this. However, setting S15 to 1 enables a special mode of operation in which data is forwarded as fast as possible for the data rate for which the port is configured (at 115000bps this will typically be 2-3ms).

Note that the default value of 0 is equivalent to setting the register to 2 in order to maintain compatibility with older systems.

S23 Parity

Units: N/A
Default: 0
Range: 0-2,5,6

The value stored in S23 determines whether the parity used for the ASY port is set to None (0), Odd (1), Even (2), 8Data Odd (5) or 8Data Even (6).

S31 ASY Interface Speed

Units: N/A
Default: 0
Range: 0-11

Register S31 is used to set the speed and data format for the ASY port to which you are currently connected.

The default value for ASY 0 is 0, i.e. the port speed/data format is not set to a specific value, it is determined automatically from the AT commands that you enter.

The default value for ASY 1, 2 and 3 is 3, i.e. the ports will only accept AT commands at 115,200bps (8 data bits, no parity and 1 stop bit).

To set the speed of one of the ports to a particular value, the appropriate register should be set to the required value from the following table:

S31	Port Speed (bps)	S31	Port Speed (bps)
0	Auto-detect	6	19200
1	Reserved	7	9600
2	Reserved	8	4800
3	115200	9	2400
4	57600	10	1200
5	38400	11	300

For example, to change the speed of ASY 1 to 38,400bps, connect your terminal to that port with the speed set to 9600bps. Enter the command:

```
ats31=5
```

then change the speed of your terminal to 38,400bps before entering any more AT commands.

The data format used when the ATS31=n command is entered is selected as the data format for all further commands.

The auto-detect option is only available for ASY0 and ASY1.

S33 DTR Dialling

Units: N/A
Default: 0
Range: 0, 1

S33 is used to enable or disable DTR dialling for the port. When DTR dialling is enabled, the unit will dial the number stored for that port (see AT&Z) when the DTR signal from the terminal changes from Off to On.

S45 DTR Loss De-Bounce

Units: 0.05 seconds
Default: 5
Range: 1-255

The value in S45 determines the length of time for which the DTR signal from the terminal device must go off before the unit acts upon any options that are set to trigger on loss of DTR. Increasing or decreasing the value in S45 makes the unit less or more sensitive to "bouncing" of the DTR signal respectively.

GENERAL SYSTEM COMMANDS

The application commands described in this section are basic configuration commands that do not relate to specific types of application or network.

CONFIG Show/Save Configuration

The config command is used for the following purposes to show current or stored configuration settings, to save the current configuration or to specify which configuration is to be used when the unit is powered up or rebooted.

The format of the config command is:

config <0|1|c> <save|show|powerup>

Two separate configurations can be stored, numbered 0 and 1. The first parameter of the config command specifies to which configuration the command applies. The letter "c" denotes the current configuration settings, i.e. those currently in use.

The second parameter is one of the following keywords:

show displays the specified configuration (either 0, 1 or c for the current configuration)

save saves the current settings as the specified configuration (either 0 or 1)

powerup sets the specified configuration (either 0 or 1) to be used at power-up or reboot

For example, to display the current configuration use the command:

config c show

The output will appear similar to the following example:

```
config c show
eth 0 descr "LAN 0"
eth 0 IPaddr "192.168.1.1"
eth 0 mask "255.255.255.0"
eth 0 bridge ON
eth 1 descr "LAN 1"
eth 2 descr "LAN 2"
eth 3 descr "LAN 3"
eth 4 descr "ATM PVC 0"
```

The config files only contain details of those settings that are different from the unit's default settings. If you make a setting that is the same as the default setting, it will not appear in a stored configuration.

To save the current settings to configuration file 1, enter:

config 1 save

To use configuration 1 when the unit is powered up or rebooted, enter:

config 1 powerup

Config changes counter

The config changes command shows the number of changes to the current configuration since the unit has powered up and the initial configuration file run. Also shows the time when the config file was last saved.

REBOOT Reboot Unit

The ***reboot*** command causes the unit to execute a complete hardware reset, loading and running the main image file from cold. It has three modes of operation:

reboot - will reboot the unit after any FLASH write operations have been completed. Also, 1 second each is allowed for the following operations to be completed before reboot will take place:

- IPSec SA delete notifications have been created and sent
- TCP sockets have been closed
- PPP interfaces have been disconnected

reboot <n> - will reboot the unit in <n>minutes where n is 1 to 65,535

reboot cancel - will cancel a timed reboot if entered before the time period has passed.

Reset router to factory defaults

See reference guide section titled "Administration - Factory Default Settings".

Disabling the reset button

Normally when the reset button is held in for 5 seconds the router is reset to factory defaults. The factory reset button functionality can be disabled / enabled if required.

The command to disable the reset button is "*cmd 0 pbreset off*"

To re-enable the reset button functionality "*cmd 0 pbreset on*"

TEMPLOG Temperature monitoring

The on-board temperature sensors are sampled every 60 seconds and any 'interesting' changes in the temperature are logged to a special flash file, 'templog.c1'. Use '*templog 0 status*' to view the last stored record in this file.

There are 2 sensors built in, there is one on the motherboard and one on the modem module. If a temperature is reached that is outside of normal operating limits, an event will be logged in the eventlog.txt
Note: The only transport models that support TEMPLOG are DR64 and VC7400.

Ping and Traceroute

From the CLI, these commands can be used to help troubleshoot connectivity problems.

The syntax of the ping command is:

ping <ip address|FQDN> [n]

Where n (if used) is the number of ICMP echo requests to send. If not specified, only 1 echo request will be sent.

To stop pings when n has been set to a high value use *ping stop*

The syntax of the traceroute command is:

traceroute <ip address|FQDN>

To stop a failed trace if hosts can not be detected, use *traceroute stop*

Clearing the Analyser Trace and Event Log

To clear the analyser trace, the CLI command is *ana 0 anaclr*

To clear the event log, the CLI command is *clear_ev*

Activate and Deactivate interfaces

To manually activate (or raise) an interface, the following CLI command can be used as an activation request.

<entity> <instance> act_rq

To manually deactivate (or lower) an interface, the following CLI command can be used as an activation request.

<entity> <instance> deact_rq

Where *<entity>* can be:

PPP for PPP interfaces
TUN for GRE TUN interfaces
OVPN for OpenVPN interfaces

And *<instance>* is the interface number, such as 0, 1, 2 etc

For example, to activate PPP 1, the CLI command would be:

ppp 1 act_rq

and to deactivate PPP 1:

ppp 1 deact_rq

Special Usernames

There are some special usernames that can also be used for both local and remote authentication, these are:

%s This uses the serial number of the router as the username.

%i This uses the IMEI of the cellular module as the username.

%c This uses the ICCID of the SIM as the username.

If a '%' symbol is part of the username, it must be escaped with another '%' symbol. For example 'user%1' should be entered as 'user%%1'.

GPIO (General Purpose Input Output)

GPIO commands are necessary to configure WR44, which has one Digital Input/Output port and one Digital Input port. This command allows configuration of the I/O port either as an input port or an output port. For example:

Command	Description
gpio inout input	Configures the I/O port as an input.
gpio inout output	Configures the I/O port as an output.
gpio inout ON	Sets the I/O port to ON when configured as an output.
gpio inout OFF	Sets the I/O port to OFF when configured as an output.

The syntax of the command is as follows:

Usage: *gpio [inout ON|OFF|input|output]*

With no parameters, the command will display the current status of the ports. For example:

```
gpio inout on
Input(s):
         in : OFF
Output(s):
      inout : ON
OK
```

The Input and Input/Output connections (pins 2 and 3) are programmed via the command line using the gpio command. The default setting for pins 2 and 3 are OFF as seen in the above example.

Note:

Only one of the power connectors should be used. Never apply power to both the MAIN and AUX connectors at the same time.

Pin	Description
Pin 1	GROUND
Pin 2	INPUT
Pin 3	Input/ Output
Pin 4	Power

TCPPERM AND TCPDIAL

This section describes the operation of the tcpperm and tcpdial commands which are available only as application commands and have no equivalent web pages.

TCPPERM

The tcpperm command is used to establish a permanent "serial to IP" connection between one of the ASY ports and a remote IP host. After the command has been executed, the unit will automatically open a socket connection to the remote peer whenever data is received from a terminal attached to the specified ASY port. When the socket is first opened and the connection has been established, the unit will issue a CONNECT message to the terminal and will subsequently relay data between the socket and the ASY port. The format of the CONNECT message can be modified using the standard AT commands (e.g. ATV, ATE, etc.) or using the Configuration - Network > Interfaces > Serial > Serial Port n web page.

Note:
The serial port should also be pre-configured to use the appropriate word format, speed and flow control.

While the serial-to-IP connection is established, if the attached serial device drops the DTR signal, then the socket connection will be terminated, much as with a standard modem or terminal adapter. Again this behaviour can be modified via the AT&D command or the serial port settings.

The format of the command is:

```
TCPPERM <[ASY 0-1]> <Dest Host> <Dest Port> [UDP] [nodeact] [-
l<listening port>] [-i<inact_timeout>] [-f<fwd_time>] [-e<eth_ip>][-
d(deact link)] [-k<keepalive_time>] [-s<src_port>] [-ok] [-
t<telnet_mode>] [-ho(host only)] [-ssl] [-ao(always open)] [-m<mhome
idx>]
```

The parameters are detailed in the following table:

Parameter	Description
ASY	The number of the ASY port that the link will be made from/to
Dest Host	The IP address (or name) of the remote peer
Dest Port	The port number to use on the remote peer
UDP	Open a UDP connection (the default is TCP)
-ao	Open socket immediately, and reopen if and when the socket is closed
-e	Use the address of ethernet port 'n' for the socket connection rather than the default of the address of the interface over which the socket is opened (i.e ppp 1, ppp 2, etc.)
-d	Deactivate link - if non-zero, when the socket is closed and there are no other sockets using the interface then the interface connection is dropped (switched connections only)
-f	The forwarding time (x10ms) for packetising data from the serial port
-ho	Host - indicates that the socket should only accept connections from the specified host.

Parameter	Description
-i	The inactivity timeout (s) after which the socket will be closed
-k	Keep alive packet timer (s)
-l	Listening port - allows the user to set a new TCP port number to listen on rather than the default value of 4000+ASY port #
-m	Multihome additional consecutive addresses index
-ok	Open socket in 'quiet mode', i.e. there is no 'OK' response to the TCPPERM command.
-s	Source port number
-ssl	Use SSL mode
-t	Use Telnet mode. Opens socket in the corresponding Telnet mode (port 23 default), 0= raw, 1 Telnet Mode, 2 - Telnet Mode with null stuffing. If this is not specified then the mode specified for the associated ASY port in general setup is used. If the -t option is specified then the "ok" option is always used.

The command can also be made to execute automatically on power-up by using the "cmd n autocmd 'cmd'" macro command, i.e.

```
cmd 0 autocmd 'tcpperm asy 0 192.168.0.1 -f3 -s3000 -k10 -e1'
```

Considerations for use with VPN or GRE Tunnels

When the socket used by TCPPERM is opened the default behaviour is to use the address of the interface over which the socket is carried (ETHn or PPPn) as the source address of the socket. If the socket data is to be tunnelled then it may be necessary to use the -en modifier so that the source address of the socket matches the local subnet address specified in the appropriate Eroute. A similar effect can also be achieved by setting the parameter Default source IP address interface: Ethernet n in the Web interface on the Configuration - Network > Advanced Network Settings.

TCPDIAL

TCPDIAL operates in an identical manner to TCPPERM except that establishment of the socket connection is not automatic and must be initiated by the tcpdial command. The simplest method of achieving this is to map a command using the ***Configuration - Network > Interfaces > Serial > Command Mappings***, i.e. Command to Map ATDT0800456789 maps to "tcpdial asy 1 217.36.133.29 -e0". Now, whenever the attached terminal device attempts to dial the number defined the unit will map it to an IP socket connection.

In this way multiple dial commands can be directed to the same or different IP hosts with other simple command mappings.

Aborting TCPDIAL

The tcpdab command can be used to cancel a TCPDIAL connection before the connection has been made. It can also be used from a command session to disconnect an existing TCPDIAL connection on another ASY port.

The format of the command is:

```
tcpdab <instance> ATH
```

where *<instance>* is the number of the ASY port.

SERIAL PORT CONNECTIONS

Depending upon the model, the asynchronous serial ports on may be presented as DB 25 sockets, DB 9 sockets or 8-pin RJ45 sockets. On some models, a combination of the above may be used. The following tables list the pin designations of each type of connector for each Digi model.

The RS-232 port pin-outs are suitable for both Async and Sync port connections. When used in Async mode the pins for TxC, RxC & ETC are not required, these are needed for Sync mode only.

RS-232 Port Pin-Outs

			DB 25	RJ45
Description	**RS232 signal**	**Direction[1]**	**Pin #**	**Pin #**
Transmit Data	TxD	**in**	2	6
Receive Data	RxD	**out**	3	3
Ready To Send	RTS	**in**	4	1
Clear To Send	CTS	**out**	5	8
Data Set Ready	DSR	**out**	6	n/a
Ground	GND	**n/a**	7	5
Data Carrier Detect	DCD	**out**	8	7
Transmitter Clock	TxC	**out**	15	n/a
Receiver Clock	RxC	**out**	17	n/a
Data Terminal Ready	DTR	**in**	20	2
Ring Indicate	RI	**out**	22	n/a
External Transmitter Clock	ETC	**in**	24	n/a

1. With respect to Digi units

X.21 (RS-422)

Note:
In order for the DR64x0(W) to operate in X.21 mode, a kepler daughter card must be fitted.

			DB 25
Description	**X.21 signal**	**Direction[1]**	**Pin #**
Receive Data (A)	RxDA	**out**	3
Receive Data (B)	RxDB	**out**	16
Transmit Data (A)	TxDA	**in**	2
Transmit Data (B)	TxDB	**in**	14
Indication (B)	INDB	**out**	13
Ground	GND	**n/a**	7
Control (B)	CTLB	**in**	19
Clock (A)	CLKA	**in or out[2]**	17
Clock (B)	CLKB	**in or out[2]**	9
Indication (A)	INDA	**out**	5
Control (A)	CTLA	**in**	4

1. With respect to Digi units
2. Direction depends on whether the Digi unit is clock sink or clock source.

X.21 25-Pin to 15-Pin Straight Through Cable – Internal Clock

This is normally the cable to use to connect an X.21 terminal (e.g. an ATM) to the Digi. Use this cable when the Digi is the clock ***source*** or configured as "internal clock".

DB 25- Digi Side		DB 15	
			
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	Shield	Shield	Frame Ground (Case)
RxDA	3	4	RxDA
RxDB	16	11	RxDB
TxDA	2	2	TxDA
TxDB	14	9	TxDB
INDB	13	12	INDB
GND	7	8	GND
CTLB	19	10	CTLB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	5	INDA
CTLA	4	3	CTLA

N.B. Frame Ground is optional.

X.21 25-Pin to 15-Pin Straight Through Cable – External Clock

This is normally the cable to use to connect an X.21 terminal (e.g. an ATM) to the Digi. Use this cable when the Digi is the clock ***sink*** or configured as "external clock".

DB 25- Digi Side		DB 25	
			
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	Shield	Shield	Frame Ground (Case)
RxDA	3	4	RxDA
RxDB	16	11	RxDB
TxDA	2	2	TxDA
TxDB	14	9	TxDB
INDB	13	12	INDB
GND	7	8	GND
CTLB	19	10	CTLB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	5	INDA
CTLA	4	3	CTLA

N.B. Frame Ground is optional.

Note:

When operating an X.21 (RS-422) link Synchronously it is necessary to fit termination resistors to each signal pair at the receiving end. The Digi already has in-built terminating resistors, but terminating resistors will need to be fitted between the RxDA & RxDB pins, CLKA & CLKB pins and INDA & INDB pins at the DTE.

X.21 25-Pin to 15-Pin Crossover Cable – Internal Clock

This is normally the cable to use to connect the Digi to an X.21 leased line. Use this cable when the Digi is the clock ***source*** or configured as "internal clock".

DB 25- Digi Side		DB 15	
			
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	Shield	Shield	Frame Ground (Case)
RxDA	3	2	TxDA
RxDB	16	9	TxDB
TxDA	2	4	RxDA
TxDB	14	11	RxDB
INDB	13	10	CTLB
GND	7	8	GND
CTLB	19	12	INDB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	3	CTLA
CTLA	4	5	INDA

N.B. Frame Ground is optional.

X.21 25-Pin to 15-Pin Crossover Cable – External Clock

This is normally the cable to use to connect the Digi to an X.21 leased line. Use this cable when the Digi is the clock ***sink*** or configured as "external clock".

DB 25- Digi Side		DB 15	
			
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	1	1	Frame Ground (Case)
RxDA	3	2	TxDA
RxDB	16	9	TxDB
TxDA	2	4	RxDA
TxDB	14	11	RxDB
INDB	13	10	CTLB
GND	7	8	GND
CTLB	19	12	INDB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	3	CTLA
CTLA	4	5	INDA

N.B. Frame Ground is optional.

Note:

When operating an X.21 (RS-422) link Synchronously it is necessary to fit termination resistors to each signal pair at the receiving end. The Digi already has in-built terminating resistors, but terminating resistors will need to be fitted between the TxDA & TxDB pins, CLKA & CLKB pins and CTLA & CTLB pins at the DTE.

RS-232 Port Pin-Outs

			DB 25	DB 9	RJ45
Description	**RS232 signal**	**Direction[1]**	**Pin #**	**Pin #**	**Pin #**
Transmit Data	TxD	**in**	2	3	6
Receive Data	RxD	**out**	3	2	3
Ready To Send	RTS	**in**	4	7	1
Clear To Send	CTS	**out**	5	8	8
Data Set Ready	DSR	**out**	6	6	n/a
Ground	GND	**n/a**	7	5	5
Data Carrier Detect	DCD	**out**	8	1	7
Transmitter Clock	TxC	**out**	15	n/a	n/a
Receiver Clock	RxC	**out**	17	n/a	n/a
Data Terminal Ready	DTR	**in**	20	4	2
Ring Indicate	RI	**out**	22	9	n/a
External Transmitter Clock	ETC	**in**	24	n/a	n/a

1. With respect to Digi units

X.21 (RS-422)

Note:
In order for the WR44 to operate in X.21 mode, a Viper daughter card must be fitted.

			DB 25
Description	**X.21 signal**	**Direction[1]**	**Pin #**
Transmit Data (A)	TxDA	**in**	2
Receive Data (A)	RxDA	**out**	3
Control (A)	CTLA	**in**	4
Indication (A)	INDA	**out**	5
Ground	GND	**n/a**	7
Clock (B)	CLKB	**in or out[2]**	9
Indication (B)	INDB	**out**	13
Transmit Data (B)	TxDB	**in**	14
Receive Data (B)	RxDB	**out**	16
Clock (A)	CLKA	**in or out[2]**	17
Control (B)	CTLB	**in**	19

1. With respect to Digi units
2. Direction depends on whether the Digi unit is clock sink or clock source.

X.21 25-Pin to 15-Pin Straight Through Cable – Internal Clock

This is normally the cable to use to connect an X.21 terminal (e.g. an ATM) to the Digi. Use this cable when the Digi is the clock *source* or configured as "internal clock".

DB 25- Digi Side		DB 15	
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	Shield	Shield	Frame Ground (Case)
RxDA	3	4	RxDA
RxDB	16	11	RxDB
TxDA	2	2	TxDA
TxDB	14	9	TxDB
INDB	13	12	INDB
GND	7	8	GND
CTLB	19	10	CTLB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	5	INDA
CTLA	4	3	CTLA

N.B. Frame Ground is optional.

X.21 25-Pin to 15-Pin Straight Through Cable – External Clock

This is normally the cable to use to connect an X.21 terminal (e.g. an ATM) to the Digi. Use this cable when the Digi is the clock *sink* or configured as "external clock".

DB 25- Digi Side		DB 25	
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	Shield	Shield	Frame Ground (Case)
RxDA	3	4	RxDA
RxDB	16	11	RxDB
TxDA	2	2	TxDA
TxDB	14	9	TxDB
INDB	13	12	INDB
GND	7	8	GND
CTLB	19	10	CTLB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	5	INDA
CTLA	4	3	CTLA

N.B. Frame Ground is optional.

Note:
When operating an X.21 (RS-422) link Synchronously it is necessary to fit termination resistors to each signal pair at the receiving end. The Digi already has in-built terminating resistors, but terminating resistors will need to be fitted between the RxDA & RxDB pins, CLKA & CLKB pins and INDA & INDB pins at the DTE.

X.21 25-Pin to 15-Pin Crossover Cable – Internal Clock

This is normally the cable to use to connect the Digi to an X.21 leased line. Use this cable when the Digi is the clock *source* or configured as "internal clock".

DB 25- Digi Side		DB 15	
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	Shield	Shield	Frame Ground (Case)
RxDA	3	2	TxDA
RxDB	16	9	TxDB
TxDA	2	4	RxDA
TxDB	14	11	RxDB
INDB	13	10	CTLB
GND	7	8	GND
CTLB	19	12	INDB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	3	CTLA
CTLA	4	5	INDA

N.B. Frame Ground is optional.

X.21 25-Pin to 15-Pin Crossover Cable – External Clock

This is normally the cable to use to connect the Digi to an X.21 leased line. Use this cable when the Digi is the clock ***sink*** or configured as "external clock".

DB 25- Digi Side		DB 15	
Signal	**Pin # (DCE)**	**Pin # (DTE)**	**Signal**
Frame Ground (Case)	1	1	Frame Ground (Case)
RxDA	3	2	TxDA
RxDB	16	9	TxDB
TxDA	2	4	RxDA
TxDB	14	11	RxDB
INDB	13	10	CTLB
GND	7	8	GND
CTLB	19	12	INDB
CLKB	9	13	CLKB
CLKA	17	6	CLKA
INDA	5	3	CTLA
CTLA	4	5	INDA

N.B. Frame Ground is optional.

Note:

When operating an X.21 (RS-422) link Synchronously it is necessary to fit termination resistors to each signal pair at the receiving end. The Digi already has in-built terminating resistors, but terminating resistors will need to be fitted between the TxDA & TxDB pins, CLKA & CLKB pins and CTLA & CTLB pins at the DTE.

RS-232 Port Pin-Outs

			DB 25	DB 9
Description	**RS232 signal**	**Direction[1]**	**Pin #**	**Pin #**
Transmit Data	TxD	**in**	2	3
Receive Data	RxD	**out**	3	2
Ready To Send	RTS	**in**	4	7
Clear To Send	CTS	**out**	5	8
Data Set Ready	DSR	**out**	6	6
Ground	GND	**n/a**	7	5
Data Carrier Detect	DCD	**out**	8	1
Transmitter Clock	TxC	**out**	15	n/a
Receiver Clock	RxC	**out**	17	n/a
Data Terminal Ready	DTR	**in**	20	4
Ring Indicate	RI	**out**	22	9
External Transmitter Clock	ETC	**in**	24	n/a

1. With respect to Digi units

ER2110, IR2110 & MR2110

RS-232 Port Pin-Outs

			DB 25
Description	**RS232 signal**	**Direction[1]**	**Pin #**
Transmit Data	TxD	**in**	2
Receive Data	RxD	**out**	3
Ready To Send	RTS	**in**	4
Clear To Send	CTS	**out**	5
Data Set Ready	DSR	**out**	6
Ground	GND	**n/a**	7
Data Carrier Detect	DCD	**out**	8
Transmitter Clock	TxC	**out**	15
Receiver Clock	RxC	**out**	17
Data Terminal Ready	DTR	**in**	20
Ring Indicate	RI	**out**	22
External Transmitter Clock	ETC	**in**	24

1. With respect to Digi units

RS-232 Port Pin-Outs

			DB
Description	**RS232 signal**	**Direction[1]**	**Pin #**
Transmit Data	TxD	**in**	2
Receive Data	RxD	**out**	3
Ready To Send	RTS	**in**	4
Clear To Send	CTS	**out**	5
Data Set Ready	DSR	**out**	6
Ground	GND	**n/a**	7
Data Carrier Detect	DCD	**out**	8
Transmitter Clock	TxC	**out**	15
Receiver Clock	RxC	**out**	17
Data Terminal Ready	DTR	**in**	20
Ring Indicate	RI	**out**	22
External Transmitter Clock	ETC	**in**	24

1. With respect to Digi units

Port Pin-Outs

RS-232

			DB 25	RJ45
Description	**RS232 signal**	**Direction[1]**	**Pin #**	**Pin #**
Transmit Data	TxD	**in**	2	6
Receive Data	RxD	**out**	3	3
Ready To Send	RTS	**in**	4	1
Clear To Send	CTS	**out**	5	8
Data Set Ready	DSR	**out**	6	4
Ground	GND	**n/a**	7	5
Data Carrier Detect	DCD	**out**	8	7
Transmitter Clock	TxC	**out**	15	n/a
Receiver Clock	RxC	**out**	17	n/a
Data Terminal Ready	DTR	**in**	20	2
Ring Indicate	RI	**out**	22	n/a
External Transmitter Clock	ETC	**in**	24	n/a

1. With respect to Digi units

X.21 (RS-422)

Note:

In order for the GR2130 to operate in X.21 mode, an X.21 daughter card must be fitted, with the jumpers set correctly. See “Configuring X.21 on Older Models” on page 522.

			DB 25
Description	**X.21 signal**	**Direction[1]**	**Pin #**
Receive Data (A)	RxDA	**out**	2
Receive Data (B)	RxDB	**out**	3
Transmit Data (A)	TxDA	**in**	4
Transmit Data (B)	TxDB	**in**	5
Indication (B)	INDB	**out**	6
Ground	GND	**n/a**	7
Control (B)	CTLB	**in**	8
Clock (B)	CLKB	**in or out[2]**	15
Clock (A)	CLKA	**in or out[2]**	17
Indication (A)	INDA	**out**	20
Control (A)	CTLA	**in**	22

1.

2. Direction depends on whether the Digi unit is clock sink or clock source.