MAX Diagnostic Command Reference

This guide provides all available information about the MAX Diagnostic commands. The information is organized for quick reference, and does not include tutorials. All commands are listed alphabetically.

Under most circumstances, Diagnostic commands are not required for correct operation of the MAX, and in some circumstances might produce undesirable results. Please use the following information with caution. Contact Ascend Technical Support with any questions or concerns.

Note: Every attempt has been made to confirm that this chapter correctly describes the functionality and output of the MAX Diagnostic commands. But while Diagnostic mode can be a very valuable troubleshooting tool for anyone, its primary focus is on the requirements of Ascend's development engineers. For this reason, Ascend does not guarantee the completeness of list of commands or the cataloging of functionality from release to release.

Using MAX Diagnostic Commands

To be allowed access to Diagnostic mode, you must set the Field Service privilege to Yes in the active Security Profile. (If you have any questions about how to activate Security Profiles, refer to the *MAX Security Supplement.*)

Use one of two methods to access Diagnostic mode:

- From the MAX VT100 interface, display the DO menu by pressing Ctrl-D. Then press D or select D=Diagnostics.
- From the MAX VT100 interface, type the following key sequence in rapid succession:

  ```
  Esc [ Esc =
  ```

  (Press the Escape key, then the left-bracket, then the Escape key, then the equals sign.)

  You must type all 4 four keys within one second for the MAX to recognize the escape sequence.

To display an abbreviated list of the most commonly used commands, enter a question mark:

```
MAX>?
```

To see a complete listing, type `ascend` after the question mark:

```
MAX>? ascend
```

To exit Diagnostic mode, enter `quit`.

Using combinations of commands

Since most commands are designed to give a developer information about specific portions of MAX functionality, you might find it helpful to use commands in combination to troubleshoot different problems.

For example, if you see problems with the initial connection of remote users, you might want to use a combination of `networki`, `routmgr` and `wantoggle` to obtain a complete view of three functions involved in establishing a call.

When troubleshooting modem-related issues, you might want to use modemdrvstate, modemdiag and mdialout (if modem outdial is supported on your MAX) to get all modem-related information for your calls.

Using several commands simultaneously not only gives you a clearer picture of a given scenario, it also shows you a chronological timeline of the events that are happening.

Command reference

Following are the MAX Diagnostic commands in alphabetic order:

?

Description: Displays an abbreviated list of the most commonly used Diagnostic commands and a brief description of each command. Use the `ascend` modifier to see the complete list of commands.

Usage: **`?`** `[` **`ascend`** `]`

Syntax element:	Description:
ascend	List all commands.

Example:

```
MAX> ?
? -> List all monitor commands
clr-history -> Clear history log
ConnList -> Display connection list information
ether-display -> ether-display <port #> <n>
fatal-history -> List history log
fclear -> clear configuration from flash
FiltUpdate -> Request update of a connection
frestore -> restore configuration from flash
fsave -> save configuration to flash
help -> List all monitor commands
nslookup -> Perform DNS Lookup
priDisplay -> priDisplay <n>
quit -> Exit from monitor to menus
reset -> Reset unit
tloadcode -> load code from tftp host
trestore -> restore configuration from tftp host
tsave -> save configuration to tftp host
```

```
wanDisplay -> wanDisplay <n>
wanDSess -> wandsess <sess <n>> (display per session)
wanNext -> wanNext <n>
wanOpening -> wanOpening <n> (displays packets during opening/negotiation)
```

AddrPool

Description: Displays messages related to dynamic address pooling. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `addrpool` at the MAX prompt.

Example: Following are several examples of output displayed by `addrpool`.

With 18 addresses currently allocated from a pool `addrpool` might display:

```
ADDRPOOL: lanAllocate index 0 inuse 18
```

The address 208.147.145.155 was just allocated:

```
ADDRPOOL: allocate local pool address [208.147.145.155]
```

The address 208.147.145.141 is to be freed because the user of that address has hung up his call. The Max must find the pool to which the pool belonged, then free the address so it is available for another user:

```
ADDRPOOL: found entry by base [208.147.145.141] entry
[208.147.145.129]
ADDRPOOL: free local pool address [208.147.145.141]
```

A new pool is created. Under Ethernet > Mod Config > WAN Options, Pool #1 Start is set to 192.168.8.8, and Pool #1 Count is set to 4:

```
ADDRPOOL: Deleting addrPool
ADDRPOOL: New Addr pool rc = 0
addrPool index 1 ip [192.168.8.8] count 4
```

The parameter Pool #1 Count of existing pool is changed from 4 to 3

```
ADDRPOOL: Deleting addrPool
ADDRPOOL: New Addr pool rc = 0
addrPool index 1 ip [192.168.8.8] count 3
```

A second pool is created. Under Ethernet > Mod Config > WAN Options, Pool#2 Start is set to 192.168.10.8, and Pool #2 Count is set to 10:

```
ADDRPOOL: Deleting addrPool
ADDRPOOL: New Addr pool rc = 0
addrPool index 1 ip [192.168.8.8] count 4
ADDRPOOL: New Addr pool rc = 0
addrPool index 1 ip [192.168.8.8] count 4
addrPool index 2 ip [192.168.10.1] count 10
```

The second pool is deleted:

```
ADDRPOOL: Deleting addrPool
ADDRPOOL: New Addr pool rc = 0
addrPool index 1 ip [192.168.8.8] count 4
```

ARPTable

Description: Displays the Address Resolution Protocol (ARP) table on the MAX. The MAX uses the ARP table to associate known IP addresses with physical hardware addresses.

Usage: Enter `arptable` at the command prompt.

Example:

```
MAX> arptable
ip address   ether addr  if rts pkt   ref  insert
DYN    206.30.33.11 00A0244CCE04   0   0   0     1   281379
DYN   206.30.33.254 00605C4CA220   0   0   0     1   281303
DYN    206.30.33.21 00059A403B47   0   0   0     1   281179
DYN    206.30.33.15 00A0247C2A72   0   0   0     1   281178
```

The ARP table displays the following information:

Column Name:	Description
	First, unnamed column indicates how the address was learned, dynamically (DYN) or by specification of a Bridge Address (STA).
ip address	Network address contained in ARP requests.
ether addr	Media Access Control (MAC) address of the host identified by `ip address`. This is also referred to as the hardware address.
if	Interface on which the MAX received the ARP request.
rts	Routes pointing to the address.
pkt	Number of packets queued.
ref	Number of times that the address was used.
insert	Time at which this entry was inserted into the ARP table.

ARPToggle

Description: Displays messages related to ARP. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `arptoggle` at the command prompt.

Example: Following are two examples of `arptoggle messages.`

As ARP entries are added to the ARP cache, you'll see the following output:

```
ARP-0: update cache for 208.147.154.50/32.
ARP-0: update cache for 206.30.33.252/32.
```

As ARP entries are removed normally from the ARP cache, you see output similar to the following:

```
ARP: delete A@B0524080 208.147.154.39/32.
```

Avm

Description: Displays a report on the status of the availability of modems in the MAX. Each time you type `avm`, you get a snapshot look at current modem states and the recent history for each modem. This command is particularly helpful in troubleshooting modem connection problems, for which you must focus on the ability of individual modems to successfully connect with dial-in users.

A call is noted as successful if modem handshaking (training) and authentication are successful.

A call is noted as bad if modem handshaking fails at any point in the initial call set-up, or if the dial-in user does not successfully log in.

The `dir` parameter indicates the direction of the last call into each modem:

1 - call direction unknown
2 - call was outgoing
3 - call was incoming

A modem is moved to the *suspect* list if its first four calls are bad, or if it experiences 8 bad calls in a row. Modems on the *suspect* list may still be used if all *free* modems are in use. Any subsequent successful call to a *suspect* modem places that modem back on the *free* list.

Note: A call that has been categorized as bad does not necessarily indicate a modem problem with the MAX. Poor line quality, software problems with the calling modem, wrong numbers, and forgotten passwords all can generate calls that display as bad calls that have nothing to do with modems on the MAX.

Usage: Enter `avm` at the command prompt.

Example: In the example below, an 8-mod modem card is located in slot 8 of the MAX. Modems 8:5 and 8:6 are in use. Modems 8:2, 8:3, 8:4, 8:7, and 8:8 are idle and available to accept calls. Modem 1 has been disabled via the V.34 Modem > Modem Diag > Modem #1 parameter.

Looking at Modem 4 on slot 8 (designated 8:4), the eight digit hexadecimal number has to be converted to binary to indicate how many of the last 32 calls were successful.

ffdffbfc = 11111111110111111111101111111100

For modem 8:4, there have been 4 unsuccessful calls. The last two calls were unsuccessful.

dir=3 indicates that the last call was an incoming call.

```
MAX> avm
Modems on free list:
Modem 8:4, 70 calls, 6 bad, last 32 calls = ffdffbfc dir=3
Modem 8:8, 54 calls, 1 bad, last 32 calls = ffffffff dir=3
Modem 8:3, 63 calls, 1 bad, last 32 calls = fffbffff dir=3
Modem 8:2, 74 calls, 1 bad, last 32 calls = ffffffff dir=3
Modem 8:7, 64 calls, 2 bad, last 32 calls = ffbfffbf dir=3
Modems on suspect list:
Modem 8:1, 57 calls, 0 bad, last 32 calls = ffffff00 dir=3
Modems on disabled list:
Modems on dead list:
Modems on busy list:
```

```
Modem 8:5, 65 calls, 2 bad, last 32 calls = fffffffd dir=3
Modem 8:6, 58 calls, 1 bad, last 32 calls = ffffffff dir=3
```

Bridge-Info

Description: Displays the bridge address table. This table can be helpful in troubleshooting particular hosts that are having difficulty in reaching specific destinations. A bridge uses the MAC address to determine the side on which a particular frame should be sent.

The commands displays a snapshot of the bridging table at the current time.

Usage: Enter `bridge-info` at the command prompt.

Example:

```
MAX> bridge-info
Bucket  ---Mac Address---  IF  Prof  LastUsed
7    00:80:5f:5a:25:a7   0    0      d40d3
96    00:c0:7b:61:b7:0d   0    0      d41c9
174    00:c0:7b:6b:c7:b9   0    0      d41ce
175    00:c0:7b:61:58:2d   0    0      d41e2
213    00:40:c7:5a:64:6c   0    0      d41e3
246    00:c0:7b:54:79:60   0    0      d41cc
```

The bridge-info table displays the following information:

Column Name:	**Description**
Bucket	A number used internally.
Mac Address	The Media Access Control (MAC) address of a known host. Also referred to as the hardware address.
IF	The interface number on which this host is located.
Prof	The name of the associated Connection profile.
LastUsed	The last time this entry was referenced.

BriDisplay

Description: Displays messages related to the D-channel signalling for any BRI slot cards installed on the MAX. This command is available only if you have loaded a version of MAX software that supports BRI slot cards.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the BriDisplay command during a period of low throughput.

Usage: bridisplay n

where *n* is the number of octets to display per frame. Specifying a value of zero disables the logging of this data.

Example:

```
MAX> **bridisplay 4**
BRI-XMIT-7:  : 4 octets @ B04EE520
[0000]: 00 B3 01 01
BRI-RCV-7:  : 4 octets @ B0539A80
[0000]: 02 B3 01 01
BRI-XMIT-7:  : 4 octets @ B0529560
[0000]: 02 B3 01 01
BRI-RCV-7:  : 4 octets @ B05608A0
[0000]: 00 B3 01 01
```

BrouterDebug

Description: Displays messages related to the Bridge/Router functionality of the MAX. The command is a toggle that alternately enables and disables the debug display.

You can use this command to give a general view of the load on the MAX.

Usage: Enter `brouterdebug` at the command prompt.

Example:

Typically there are very few messages logged with brouterdebug enabled. The following display appeared over several minutes on a MAX with 40-45 users active:

```
MAX> brouterdebug
BROUTER debug display is ON
BROUTER_LOAD_MSG: time= 0
BROUTER_LOAD_MSG: time= 1
BROUTER_LOAD_MSG: time= 0
MAX> brouterdebug
BROUTER debug display is OFF
```

The `BROUTER_LOAD_MSG` message is an indication of how busy the Bridge/Router function is. A low number, as is illustrated here, indicates the Bridge/Router is not experiencing any problems.

BrouterLoad

Description: Reports Bridge/Router backlog time indicating whether the MAX is experiencing any delay. The time is shown in ticks. Multiply the number of ticks by ten to get the corresponding time in milliseconds.

You can use this command to give a general view of the load on by the MAX.

Usage: Enter `brouterload` at the command prompt.

Example: The following message indicates no delays in the Bridge/Router.

```
MAX> brouterload
BROUTER load time is 0 ticks (x10msec)
```

Callback

Description: Displays messages related to the callback functionality of the MAX. You can use the command to display, for example, sessions queued for callback. The command is a toggle that alternately enables and disables the debug display.

When you enable the callback feature, the MAX hangs up after receiving an incoming call that matches the one specified in the Connection Profile. The MAX then uses the Dial # specified in the Connection Profile to call back the device at the remote end of the link.

You can use the callback command to tighten security, by ensuring that the MAX always makes a connection with a known destination. The command can also help you troubleshoot detailed areas of the callback process.

Usage: Enter `callback` at the command prompt.

Example: Following are several examples of output displayed by `callback`.

```
MAX> callback
CALLBACK debug is now ON
```

The following message appears as the MAX prepares to call back the remote end:

```
CALLBACK: processing entry entry
```

The MAX then dials the remote end:

```
CALLBACK: initiate call to entry
```

When the call has been made and is being negotiated:

```
CALLBACK: new state WAITING
```

If callback failed and will be retried:

```
CALLBACK: new state FAILED
```

If callback is never successful, the call is marked for removal from the callback list and the following message appears:

```
CALLBACK-FAILED: entry marked as failed
```

After the remote end is called back, its entry is removed from the Callback list so that the MAX can reallocate and use the resources. The following message appears:

```
CALLBACK: deleting entry entry
```

To terminate the display:

```
MAX> callback
CALLBACK debug is now OFF
```

CallRoute

Description: Displays messages related to outbound call routing on the MAX. The output is different from that of the callback diagnostic command, in that it displays output from the module that routes call between the various WAN interface types. The command is a toggle that alternately enables and disables the debug display.

This command will not log any messages if your MAX does not support outbound dialing.

Usage: Enter `callroute` at the command prompt.

Example: Following are several examples of output displayed by `callroute`.

```
MAX> callroute
CALLROUTE debug is now ON
```

When an outbound call is placed, the number is displayed:

```
CALLROUTE:routeOutboundCall: phone number 912135551212
```

Then, another message displays the trunk group being used:

```
CALLROUTE:_matchTrunkGroup: TG 9
```

When the link and channel have been validated, a message similar to the following shows that a call can be placed:

```
CALLROUTE:_matchTrunkGroup: matched T1 slot 1 DSL 0
```

The MAX then routes the outgoing call to an available channel, and the call is placed. For the call in this example, the following message appears:

```
CALLROUTE: found a matching trunk group routing call to DSL 0
```

ClockSource

Description: Displays the source of clocking for the MAX. Clock slips can cause connectivity problems, particularly for analog users. If you use the Net/T1 > Line Config > Line # > Clock Source parameter to move the clock source, you can use this Diagnostic command to validate your changes.

Note: You need to reboot the MAX to enable any changes to Clock Source. Also, if more than one line has Clock Source=Yes, remember that clock source will be derived from the first line that syncs. If you want to ensure that a particular line is the source, make sure it has Clock Source set to Yes, and that all other lines have Clock Source=No.

Usage: Enter `clocksource` at the command prompt.

Example: In the following example, the clock source is taken from the first T1/PRI line, designated dls 0. `Dsl#` indicates the maximum number of possible sources for clock. Source can be on Net/T1 slot cards or Net/BRI slot cards. This MAX has three T1/PRI lines configured, so there are three possible external sources for clock. LstSel is further validation that clock is being derived from `Dsl#0`. After `Now`, a 2 indicates that layer 2 is up for that line and is available as clock source.

```
MAX> clocksource
Clock source is dsl 0
Dsl#    01234567890123456789012345678901234567890123456789
LstSel  a?????????????????????????????????????????????????
Now     222-----------------------------------------------
```

Clr-History

Description: Clears the fatal error history log.

Usage: Enter `clr-history` at the command prompt. To view the log before clearing it, enter the `fatal-history` command.

Example:

```
MAX> fatal-history
OPERATOR RESET:  Index: 99  Load: ti.m40 Revision: 5.0A
Date: 02/13/1997.       Time: 04:22:47
```

```
DEBUG Reset from unknown in security profile 1.
SYSTEM IS UP:  Index: 100  Load: ti.m40 Revision: 5.0A
Date: 02/13/1997.       Time: 04:23:50
MAX> clr-history
```

The log is now empty:

```
MAX> fatal-history
MAX>
```

See Also: Fatal-History

ConnList

Description: Displays the names of any users that have established outgoing MPP Bundle connections.

Usage: Enter `connlist` at the command prompt.

Example:

```
MAX> connlist
Current time: 227010598
Connection list
Connection: TestUser:426   1 filters, pers=0x0,  time=0
Dataname: ,  Callname:
0xb0389108
0x0
0x0
End of Connection list
```

CoreDump

Description: Enables or disables the ability of the MAX to send the contents of its memory (core) to a specified UNIX host. When you use the function, the core file created can be several Megabytes large. Also, the UNIX host must be running the `ascendump` daemon, which is available by contacting Ascend Technical Support.

CoreDump is a particularly useful tool for Ascend's development engineering, and Technical Support occasionally requests its use to help troubleshoot specific issues.

You can include the `now` option to instruct the MAX to dump its core immediately. You can include the `enable` option to direct the MAX to dump its core when it has logged an entry to the fatal error log.

Warning: This command causes active connections to be disconnected and the MAX to reboot after its memory (core) has been dumped. Do not use the command unless specifically requested to do so by an Ascend representative.

Usage: **coredump** [**enable**] [**disable**] [**now**] *ip address*

where:

- `enable` instructs the MAX to dump its core to the specified ip address when and entry is logged to the fatal error log.
- `disable` cancels the command if it has been enabled.

- `now` instructs the MAX to dump its core immediately to the specified ip address.

Example: Following are examples of use of the coredump command with possible response messages.

```
MAX> coredump enable 1.1.1.1
coredump over UDP is enabled locally only with server 1.1.1.1
```

```
MAX> coredump disable 1.1.1.1
coredump over UDP is disabled locally only with server 1.1.1.1
```

```
MAX> coredump
coredump over UDP is disabled locally only with server 1.1.1.1
```

```
MAX> coredump enable 200.200.28.193
coreDump: Sending arp request...
coreDump: Sending arp request...
coreDump: Sending arp request...
coreDump aborted: Can't find ether address for first hop to
200.200.28.193
```

CtCheck

Description: Analyzes the CIDR tree. The command displays general statistics about the quantity of nodes and levels in the CIDR tree.

Usage: Enter `ctcheck` at the command prompt.

Example:

```
MAX> ctcheck
active nodes: 101
total nodes:  201
active%:      50%
max level:    15
ave level:    11.09
```

CtDebug

Description: Displays messages related to CIDR routing. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `ctdebug` at the command prompt.

Example:

```
MAX> ctdebug
CIDR tree debug is 0
```

DHCPToggle

Description: Displays messages related to DHCP functionality. This command provides little information, and is designed for developmental use only. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `dhcptoggle` at the command prompt.

Example: Following is sample output from `dhcptoggle`.

If the MAX receives a bad DHCP packet:

```
DHCP: bad magic cookie
```

WHen the MAX receives a good DHCP packet:

```
DHCP: packet received\n"));
```

DumpStackInfo

Description: Displays a summarized report relating to MP+ stacking calls into the MAX.

Usage: Enter `dumpstackinfo` at the command prompt.

Example: The following example is from a MAX with Stacking disabled:

```
MAX> dumpstackinfo
_stackInfoList = 0
total number = 0
```

Ether-Display

Description: Displays the contents of Ethernet packets.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the Ether-Display command during a period of low throughput.

Usage: **`ether-display`** *`port_0-# n`*

Syntax element:	Description:
port 0-#	The range of Ethernet ports on which received or transmitted packets should be displayed. Use ***`0`*** only to indicate that Ethernet packets for all ports should be displayed.
n	The number of octets to display from each Ethernet packet.

Example: To display the first 12 octets of each Ethernet packet for all ports:

```
MAX> ether-display 0 12
Display the first 12 bytes of ETHER messages
ETHER XMIT: 105 octets @ B07BE920
[0000]: 00 40 C7 5A 64 6C 00 C0 7B 0C 01 59
ETHER RECV: 64 octets @ B077EE70
[0000]: 00 C0 7B 0C 01 59 00 40 C7 5A 64 6C
ETHER XMIT: 219 octets @ B07BE920
[0000]: 00 40 C7 5A 64 6C 00 C0 7B 0C 01 59
ETHER RECV: 64 octets @ B077F4C0
[0000]: 00 C0 7B 0C 01 59 00 40 C7 5A 64 6C
MAX> ether-display 0 0
ETHER message display terminated
```

Ether-Stats

Description: Displays all statistics and error counters maintained by the Ethernet driver.

Usage: etherstats *port 0-#*

Example:

```
MAX> ether-stats 0
Tx unicast:     20695704
non-unicast: 12841
octets:      1958053023
collisions:  78108
dma under:   0
cts loss:    0
no carrier:  0
late coll:   0
drops:       0
Rx unicast:     20414347
non-unicast: 224846
octets:      1401213008
collisions:  0
short frame: 0
dma over:    0
no resource: 0
Alignment:   0
Unaligns:    30
Length Errs: 0
Restarts:    53
```

```
MAX> ether-stats 100
Port # must be in [0 - 0] range
```

```
MAX> ether-stats 10
Port # must be in [0 - 0] range
```

```
MAX> ether-stats 0 - 10
Tx unicast:     20696397
non-unicast: 12844
octets:      1958105927
collisions:  78111
dma under:   0
cts loss:    0
no carrier:  0
late coll:   0
drops:       0
Rx unicast:     20415065
non-unicast: 224913
octets:      1401828909
collisions:  0
short frame: 0
dma over:    0
no resource: 0
Alignment:   0
Unaligns:    30
```

```
Length Errs: 0
Restarts:    53
MAX> ether-stats 10 - 100
Port # must be in [0 - 0] range
MAX> ether-stats 0 - 2000
Tx unicast:     20696941
non-unicast: 12844
octets:      1958146763
collisions:  78113
dma under:   0
cts loss:    0
no carrier:  0
late coll:   0
drops:       0
Rx unicast:     20415579
non-unicast: 224956
octets:      1402266360
collisions:  0
short frame: 0
dma over:    0
no resource: 0
Alignment:   0
Unaligns:    30
Length Errs: 0
Restarts:    53
```

Fatal-History

Description: Displays the MAX fatal error log. Each time the MAX reboots, it logs a fatal-error message to the fatal-error history log. The fatal-error log also includes Warnings, for which the MAX did not reset. Development engineers use Warnings for troubleshooting purposes. A Warning indicates that the MAX detected an error condition but recovered from it. The number of entries in this log is limited by available flash space, and the errors rotate on a First-in, First-out (FIFO) basis. The log can be cleared with the Clr-History command.

Note: If your MAX experiences a fatal reset or warning, contact Ascend Technical Support immediately.

Definitions of fatal errors:

The following reset is caused when an Assert is placed in the code. This problem can be either hardware or software related. Contact Ascend Technical Support if you experience an FE1 reset.

```
FATAL_ASSERT =              1
```

The following reset is an out-of-memory condition, sometimes termed a memory leak.

```
FATAL_POOLS_NO_BUFFER =     2
FATAL_PROFILE_BAD =         3
FATAL_SWITCH_TYPE_BAD =     4
FATAL_LIF_FATAL =           5
FATAL_LCD_ERROR =           6
FATAL_ISAC_TIMEOUT =        7
```

The following reset is caused by a processor exception error.

```
FATAL_SCC_SPURIOUS_INT =    8
FATAL_EXEC_INVALID_SWITCH = 9
```

The following reset occurs because the MAX tried to allocate a mail message, and there were none left. A reset of this type is usually due to a memory leak.

```
FATAL_EXEC_NO_MAIL_DESC =   10
FATAL_EXEC_NO_MAIL_POOL =   11
FATAL_EXEC_NO_TASK =        12
FATAL_EXEC_NO_TIMER =       13
FATAL_EXEC_NO_TIMER_POOL =  14
FATAL_EXEC_WAIT_IN_CS =     15
FATAL_DSP_DEAD =            16
FATAL_DSP_PROTOCOL_ERROR =  17
FATAL_DSP_INTERNAL_ERROR =  18
FATAL_DSP_LOSS_OF_SYNC =    19
FATAL_DSP_UNUSED =          20
FATAL_DDD_DEAD =            21
FATAL_DDD_PROTOCOL_ERROR =  22
FATAL_X25_BUFFERS =         23
FATAL_X25_INIT =            24
FATAL_X25_STACK =           25
FATAL_ZERO_MEMALLOC =       27
FATAL_NEG_MEMALLOC =        28
```

The following reset is caused by a software loop.

```
FATAL_TASK_LOOP =           29
FATAL_MEMCPY_TOO_LARGE =    30
FATAL_MEMCPY_NO_MAGIC =     31
FATAL_MEMCPY_WRONG_MAGIC =  32
FATAL_MEMCPY_BAD_START =    33
FATAL_IDEC_TIMEOUT =        34
FATAL_EXEC_RESTRICTED =     35
FATAL_STACK_OVERFLOW =      36
```

The following entry is logged to the fatal-error table when the MAX has been manually reset, either in Diagnostic mode (with the RESET or NVRAMCLEAR commands), through the user interface, or through MIF.

Instead of a standard stack backtrace, the message includes the active security-profile index. The numbering is one-based, with 0 indicating an unknown security profile. On the MAX the Default profile is number 1, and the Full Access profile is number 9.

This reset is logged immediately before the MAX goes down.

```
FATAL_OPERATOR_RESET =      99
```

As a complement to entry 99, the following entry is logged as the MAX is coming up. For a normal, manual reset, you should see a fatal error 99 followed by a fatal error 100.

```
FATAL_SYSTEM_UP =           100
```

Warning messages:

Note: Warnings are not the result of reset conditions. The MAX logs Warnings when it detects a problem and recovers.

```
ERROR_BUFFER_IN_USE             101
ERROR_BUFFER_WRONG_POOL         102
ERROR_BUFFER_WRONG_HEAP         103
```

The following warning can be logged under different conditions. For example, double freeing memory or low memory conditions can both generate a Warning 104.

```
ERROR_BUFFER_NOT_MEMALLOC       104
ERROR_BUFFER_BAD_MEMALLOC       105
ERROR_BUFFER_BOGUS_POOL         106
```

Memory management code (or other modules) detected that the buffer header of what should have been a free buffer had been corrupted by the previous overwrite.

```
ERROR_BUFFER_BOGUS_HEAP         107
```

The following warning is logged when a negative length request is made to the memory allocation code.

```
ERROR_BUFFER_NEG_MEMALLOC       108
```

The following warning is similar to Warning 108, except that the a zero length request is made to the memory allocation code.

```
ERROR_BUFFER_ZERO_MEMALLOC      109
ERROR_BUFFER_BOUNDARY           110
```

The following Warning occurs when a software routine has tried to allocate a block of memory greater than 64Kbytes.

```
ERROR_BUFFER_TOO_BIG            111
ERROR_BUFFER_NULL               112
ERROR_BUFFER_SEGCOUNT_ZERO      113
ERROR_BUFFER_TRAILER_MAGIC      114
ERROR_BUFFER_TRAILER_BUFFER     115
ERROR_BUFFER_TRAILER_LENGTH     116
ERROR_BUFFER_TRAILER_USER_MAGIC 117
ERROR_BUFFER_WRITE_AFTER_FREE   118
ERROR_BUFFER_NOT_IN_USE         119
ERROR_BUFFER_MEMCPY_MAGIC       120
ERROR_BUFFER_MEMCPY_MAGIC_NEXT  121
ERROR_BUFFER_MIN                101
ERROR_BUFFER_MAX                121
```

The following Warning is caused when a memory-copy routine was called, but the source buffer was much larger than expected.

```
ERROR_LCD_ALLOC_FAILURE         145
ERROR_MEMCPY_TOO_LARGE          150
ERROR_MEMCPY_NO_MAGIC           151
ERROR_MEMCPY_WRONG_MAGIC        152
ERROR_MEMCPY_BAD_START          153
```

The following warning is caused by and error in the WAN driver.

```
ERROR_WAN_BUFFER_LEAK           154
ERROR_TERMSRV_STATE             160
ERROR_TERMSRV_SEMA4             161
ERROR_STAC_TIMEOUT              170
```

The following Warning is caused kernel temporarily does not have available memory to spawn a task.

```
ERROR_EXEC_FAILURE              175
```

The following warning is caused by a missing channel on a T1/PRI line.

```
ERROR_CHAN_MAP_STUCK            180
ERROR_CHAN_DISPLAY_STUCK        181
```

The following warning indicates that a Disconnect message to the Central Office (CO) was not sent. The problem can be caused by conditions on the MAX or at the CO. When the MAX encounters the condition, it assumes the CO is correct, and answers the call.

```
ERROR_NEW_CALL_NO_DISC_REQ      182
ERROR_NEW_CALL_NO_DISC_RESP     183
ERROR_DISC_REQ_DROPPED          184
ERROR_SPYDER_BUFFER             185
ERROR_SPYDER_DESC               186
ERROR_TCP_SBCONT_TOO_BIG        190
ERROR_TCP_SEQUENCE_GAP          191
ERROR_TCP_TOO_MUCH_DATA         192
ERROR_TCP_TOO_MUCH_WRITE        193
ERROR_TCP_BAD_OPTIONS           194
ERROR_OSPF_BASE                 200
```

Usage: Enter `fatal-history` at the command prompt.

Example:

```
MAX> fatal-history
OPERATOR RESET:  Index: 99  Load: mhpe1bip Revision: 4.6Cp22
Date: 02/24/1997.       Time: 16:08:43
DEBUG Reset from unknown in security profile 1.
OPERATOR RESET:  Index: 99  Load: ebiom.m40 Revision: 5.0A
Date: 02/24/1997.       Time: 16:09:35
NVRAM was rebuilt
SYSTEM IS UP:  Index: 100  Load: ebiom.m40 Revision: 5.0A
Date: 02/24/1997.       Time: 16:10:04
```

See Also: Clr-History

Fcat

Description: Performs a UNIX `cat` on the configuration stored in Flash memory.

Usage: Enter `fcat` at the command prompt.

Example:

```
MAX> fcat
ReadProfile returned cnt = 512
0x53 ‘S’ 0x54 ‘T’ 0x41 ‘A’ 0x52 ‘R’ 0x54 ‘T’ 0x3d ‘=’ 0x53 ‘S’ 0x41 ‘A’ 0x50
‘P’
0x46 ‘F’ 0x49 ‘I’ 0x4c ‘L’ 0x54 ‘T’ 0x3d ‘=’ 0x39 ‘9’ 0x30 ‘0’ 0x30 ‘0’ 0x3d
‘=
‘ 0xa ‘’0’ 0xd ‘
‘ 0x45 ‘E’ 0x4e ‘N’ 0x44 ‘D’ 0x3d ‘=’ 0x53 ‘S’ 0x41 ‘A’ 0x50 ‘P’ 0x46 ‘F’ 0x49
‘
‘ 0xa ‘ ‘L’ 0x54 ‘T’ 0x3d ‘=’ 0x39 ‘9’ 0x30 ‘0’ 0x30 ‘0’ 0x3d ‘=’ 0x30 ‘0’ 0xd
```

```
’
’ 0x53 ‘S’ 0x54 ‘T’ 0x41 ‘A’ 0x52 ‘R’ 0x54 ‘T’ 0x3d ‘=’ 0x53 ‘S’ 0x41 ‘A’ 0x50
‘
P’ 0x46 ‘F’ 0x49 ‘I’ 0x4c ‘L’ 0x54 ‘T’ 0x3d ‘=’ 0x39 ‘9’ 0x30 ‘0’ 0x30 ‘0’
0x3d
’ 0xa ‘1 ‘1’ 0xd ‘
```

FClear

Description: Clears Flash memory on the MAX. When the MAX boots, it loads the code and configuration from Flash memory into Dynamic Random Access Memory (DRAM). If you want to return your MAX to its factory-set defaults, you’ll need to perform an FClear.

Usage: Enter `fclear` at the command prompt.

Example:

```
MAX> fclear
.
```

See Also: FSave

FRDLState

Description: Displays information regarding the state of the Frame Relay connections, focusing mostly on Data Link information. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `frdlstate` at the command prompt.

Example:

```
MAX> frdlstate
FRDLCALL state display is now ON
```

In the following example, an outgoing call is to be placed. A route to the destination is available over a Frame Relay link.

```
FRDLCALL: Clear Call for route: 136
```

The following message indicates an outgoing call is connected.

```
FRDLCALL-136: call complete, status 1, 0 channels
```

Either the MAX or the far end device has destroyed a route. The MAX updates its table to reflect this routing change.

```
FRDLCALL-136: dead call
FRDLCALL-136: route destroyed
```

FRDump

Description: Displays a snapshot of the Frame Relay Interface table. The display shows data for each DLCI assigned to a Frame Relay link. The command The command has little practical use other than as a tool for developmental engineering. Most practical information is available through the Terminal Server SHOW commands.

Usage: Enter `frdump` at the command prompt.

Example:

```
MAX> frdump
* Frname  State  DLinkAddr  routeID.id frmgrLink dlIfNum dlIfSpeed
FR-1    CONNECTED  b04c0480  20  b04fbd40 6 0
*dlci Addr  ifNum routeID enaps   dataLink   state
b04dc760    19   19       b05102e0      b04c0480     ACTIVE
Encaps state mru  fragOffset frag  dlciAddr  xmit seq  recv seq
ACTIVE    1530     0      NO FRAG    b04dc8e0       0        0
Ip Routing
```

FRInARP

Description: Performs an Inverse ARP test over the specified Frame Relay link and DLCI. frinarp can be used to help troubleshoot connectivity and routing problems over a Frame Relay link.

Usage: **frinarp** *Frame_Relay_Profile_Name DLCI_#*

Example:

```
MAX> frinarp FR-1 38
frInArp: frinarp  frname  dlci
Inverse Arp op 2304 hw type 3840 prot type 8 hw len 2 prot len 4
Source Hw address 0401 Target Hw address 0000
Source Protocol address cd933401 Target Protocol address cd930005
```

FRestore

Description: Restores a configuration from Flash memory and loads it into DRAM on the MAX.

Note: The MAX performs an FRestore when it boots. This command needs to be performed if you've made changes to the current configuration and want to restore to the configuration stored in Flash memory.

Usage: Enter `frestore` at the command prompt.

FRLinkState

Description: Displays Frame Relay control messages. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `frlinkstate` at the command prompt.

Example:

```
MAX> frlinkstate
FR control msg display is ON
```

The following message indicates the MAX sent a Frame Relay Status Enquiry. The Send sequence number is 135. The Receive sequence number is 134.

```
FRMAIN: time 67192300, send status enquiry (135,134)
```

DLCI 16 is being processed. This is a normal message, and you should see one `process` message for each DLCI.

```
process pvc dlci 16
```

FRLMI

Description: Displays Frame Relay Link Management Interface (LMI) information. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `frlmi` at the command prompt.

Example:

```
MAX> frlmi
FRMAIN: Lmi display is ON
FRMAIN: Setting timer DTE
```

The following message validates the consistency of sequence numbers in LMI messages. The 144 after `want` indicates the original sequence number the MAX sent. The two numbers after `got` indicate the Switch's Send sequence number and the Switch's report of the last sequence number it received from the MAX, respectively. The original sequence number should match the Switch's report of the last sequence number it received.

```
FRMAIN: Time 67201400, got link report: want (*,144), got (144,144)
```

FRMgrDump

Description: Displays the Frame Relay link and DLCI information including states and counters. This information is also available through the Terminal Server SHOW commands.

Usage: Enter `frmgrdump` at the command prompt.

Example:

```
MAX> frmgrdump
Data Link Info
Status
B04FBD40 ACTIVE   B04C0480 1532   19759603   19530429
Status
enq sent =      66710              rsp rcvd =      66763
upd rcvd =         53              timeouts =          1
Errors
UI field =          0              PD field =          0
CR field =          0              msg type =          0
stat rsp =          0              lock shf =          0
inv info =          0              rpt type =          0
Last Error
type = 5
time =       6100
Fr Type 0      value: 20 octets @ B04FBE26
[0000]: 04 91 03 CC 45 00 00 3A 4B 0E 00 00 7F 11 54 D7
[0010]: CD 93 08 07
LMI type =  AnnexD
DTE Monitor  n391 = 6, t391 = 10, n392 = 3, n393 = 4
Event: recv seq 155 send Seq 155 Index = 0, cycles left = 4
OK OK OK OK OK OK OK OK OK OK
DCE Monitor  t392 = 15,n392 = 3, n393 = 4
```

```
Event:  dce send seq 0  index = 0
OK OK OK OK OK OK OK OK OK OK
DLCI info
--addr-- dlci --state- userHndl n201 --check- -pkt xmit- -pkt recv-
B04C09A0    0 ACTIVE          0 1532 NO CHECK      66710      66763
---DE--- --FECN-- --BECN-- -crTime-  chgTime  pending
0        0        0      100      100 FALSE
```

FRSCert

Description: Toggles between Sprint and Frame Relay Forum LMI checks. The default is the Sprint certification policy. In most cases, the default setting is correct and should not be changed.

The command is a toggle, so each time you type frcert will switch between the two possible selections.

Usage: Enter `frscert` at the command prompt.

Example:

```
MAX> frscert
frSCert is FRFCert
MAX > frscert
frSCert is SCert
```

FRState

Description: Displays messages related to Frame Relay state changes. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `frstate` at the command prompt.

Example: Below, data is received from the Frame Relay interface and processed:

```
MAX> frstate
FRrly state display is ON
FRRLYIF: Calling frifRecv routeId 20
FR1490 dataFrom wan entry state 2
FRRLYIF: Send up stack ifnum 1
FRRLYIF: Calling frifRecv routeId 20
FR1490 dataFrom wan entry state 2
FRRLYIF: Send up stack ifnum 7
FRRLYIF: frIfSend ifNum 1
FR1490 data to  wan entry state 2
FRRLYIF: datatoWan datalink B04C0480
```

FSave

Description: Stores the current configuration into its Flash memory.

Note: When you load code with the `tloadcode` command, an `FSave` is performed automatically before the code is uploaded. When the box boots after the upload, the MAX will load the configuration stored in Flash rather than be reset to factory default settings.

Usage: Enter `fsave` at the command prompt.

Example:

```
MAX> fsave
........................................
.
MAX>
```

Heartbeat

Description: Displays information related to multicast heartbeat functionality. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `heartbeat` at the command prompt.

Example: Following are several examples of output displayed by `heartbeat`.

```
HB: Sending SNMP Alarm count
HB: Checking Number of HeartBeats received
HB: HeartBeats received x
HB: Changing to Alarm Mode, HeartBeats Received x Expected y
HB: HeartBeat group address changed
HB: Heart beat received with invalid UDP port
HB: Heart beat received from invalid source
HB: Received HeartBeat packet
```

Help

Description: Displays a list of the most commonly used Diagnostic commands and a brief description of each command. You can use the `[ ascend ]` modifier to see the complete list of commands.

Usage: **`help`** `[` **`ascend`** `]`

Syntax element:	Description:
`ascend`	List all commands.

Example:

```
MAX> help
? -> List all monitor commands
clr-history -> Clear history log
ConnList -> Display connection list information
ether-display -> ether-display <port #> <n>
fatal-history -> List history log
fclear -> clear configuration from flash
FiltUpdate -> Request update of a connection
frestore -> restore configuration from flash
fsave -> save configuration to flash
help -> List all monitor commands
nslookup -> Perform DNS Lookup
priDisplay -> priDisplay <n>
quit -> Exit from monitor to menus
reset -> Reset unit
tloadcode -> load code from tftp host
```

```
trestore -> restore configuration from tftp host
tsave -> save configuration to tftp host
wanDisplay -> wanDisplay <n>
wanDSess -> wandsess <sess <n>> (display per session)
wanNext -> wanNext <n>
wanOpening -> wanOpening <n> (displays packets during opening/
negotiation)
```

See Also: ?

IPAdrCfg

Description: Displays messages from a routine on the MAX that is used for initial IP address assignment on an unconfigured MAX.

The command is a toggle that alternately enables and disables the debug display.

Example: Following are examples of output you'll receive when using ipadrcfg:

```
IPADRCFG: sending reply via ip
IPADRCFG: sending reply direct ethernet
```

The following message indicates a request is received for this MAX from another TCP/IP host on the ethernet LAN:

```
IPADRCFG: Probe Req from 1.1.1.1
```

IPXRipDebug

Description: Displays incoming and outgoing IPX RIP traffic. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `ipxripdebug` at the command prompt.

Example:

```
MAX> ipxripdebug
IPX-RIP state display is ON
```

The following message appears as the MAX sends an IPX RIP packet announcing its route

```
IPXRIP: 10000a17 announced 0 routes on interface 1000:
```

A P50 has dialed this MAX. The MAX receives a RIP route from the P50.

```
IPXRIP: received response from ac1b0001:00c07b5e04c0 (1 nets).
```

The following message indicates the MAX is delaying sending a RIP packet to prevent the interpacket arrival time from being closer than busy/slow boxes can handle. An IPX router should not ever violate the minimum broadcast delay.

```
IPX-RIP: too soon to send on interface 1000.

IPXRIP: 10000a81 announced 0 routes on interface 1000:
IPXRIP: received response from ac1b0001:00c07b6204c0 (1 nets).
IPXRIP: 10000aa6 announced 0 routes on interface 1000:
IPXRIP: received response from ac1b0001:00c07b5504c0 (1 nets).
IPXRIP: 10000abc announced 0 routes on interface 1000:
```

IPXSpoof

Description: Displays messages related to the IPX spoofing functionality of the MAX. When a WAN link goes down and the parameter Ethernet > Connections > IPX Options > Netware t/o has a value other than 0.

Usage: Enter `ipxspoof` at the command prompt.

Example: Following are several examples of output displayed by `IPXSpoof`.

In the following message, a watchdog packet was received by the MAX from its Ethernet interface:

```
IPX_SPOOF: watchdog packet detected:
```

```
IPX_SPOOF: timeout for record
IPX_SPOOF: CONN:
IPX_SPOOF: already have a connection record
IPX_SPOOF: creating a connection record.
IPX_SPOOF: don’t have a record of this connection.
IPX_SPOOF: deleting connection record.
```

LANVal

Description: Displays messages relating to external validation requests. You can use this command in conjunction with `radif` to troubleshoot authentication issues.

Usage: Enter `lanval` at the command prompt.

Example:

```
MAX> lanval
LANVAL state display is ON
LANVAL: radius auth, id B054AD60
LANVAL: radius callback, id B054AD60, auth SUCCESS
LANVAL:_lanvFreeInfo: freeing iprof@B05A9360
```

LifDebug

Description: Displays ISDN layer 2 and layer 3 information. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `lifdebug` at the command prompt.

Example: Following are several examples of output displayed by `LifDebug`.

```
MAX> lifdebug
LIF debug is now ON
```

A packet is being sent over the wan. The packet is constructed:

```
LIF_SendPkt(): DSL 0, source 0x400, destination 0x300,
event 0x340, SAPI 0, CES 1, Call_Id 77, Chan_Id 0
```

The following message displays the contents of the packet:

```
PACKET:
Header (4): a0 50 59 b0  Info (9): 08 02 00 00 84 08 02 80 90 01
```

```
L3_Go: source 0x400, event 0x340, DSL 0, call_id 77, ces 1
L3_ProcessUserEvent(): State 0x9, Event 0x84, Index 6,
DSL 0, CallID 77
```

Another packet is sent:

```
LIF_SendPkt(): DSL 0, source 0x300, destination 0x205,
event 0x240, SAPI 0, CES 1, Call_Id 77, Chan_Id 0
PACKET:
Header (4): a0 50 59 b0  Info (9): 08 02 83 fe 45 08 02 80 90 00
L3_Go(): end of L3 task, NLCB State 10
L2_Go(): DSL_Id=0, SAPI=0, CES=1, TEI=0, Event=240
L2_ProcessEvent(): DSL 0, index 13, state 7
L2_ProcessEvent(): DSL 0, index 19, state 7
L2_Go(): DSL_Id=0, SAPI=0, CES=1, TEI=0, Event=1
L2_ProcessEvent(): DSL 0, index 1, state 7
L2_Go() end: DLCB->State 7
```

MdbStr

Description: Modfies the default modem AT command strings used by the modems on the MAX both for incoming and for outgoing calls. Previously, you could not modify the AT command for modems on the MAX. You could affect the string in minor ways by modifying the V42/MNP, Max Baud and MDM Trn Lvl parameters located in Ethernet > Mod Config > TServe Options.

The mdbstr command also allows you to return the string to its factory default settings.

The modem chip in the MAX supports AT commands up to 56 characters in length, so to fully support all possible functionality, it is sent as two separate strings. You can modify one or both strings.

Note: The AT command string initializes the modems it supports. When you change the AT command string, you are changing the functionality of the modems. Please use this command with caution.

Here are the two default strings for the MAX:

1 AT&F0&C1V0W1X4

2 AT%C3\N3S2=255S95=44S91=10+MS=11,1,300,33600A

Usage: mdbstr [0] [1] [2] [AT command string]

Example: Below shows you how to modify the each portion of the AT command string:

This will override the existing first string with a new string:

```
mdbstr 1 AT&F0&C1V1W1
```

This will override the second portion of the AT command string:

```
mdbstr 2 AT%C3\N3S2=255S95=44S91=10+MS=11,1,300,14400A
```

This will return both strings to their factory default settings:

```
mdbstr 0
```

ModemDiag

Description: Displays diagnostic information about each modem as its call is cleared. The command is a toggle that alternately enables and disables the debug display.

The command initiates an AT&V1 call at the end of each modem call.

Usage: Enter `modemdiag` at the command prompt.

Example: modemdiag displays the following information:

Variable:	**Description**
TERMINATION REASON	LINK DISCONNECT (The remote side disconnected the call.) LOCAL REQUEST (The MAX initiated a disconnect because of poor line quality.) CARRIER LOSS GSTN CLEARDOWN (Global Switched telephone network (GSTN) initiated the disconnect) NO ERROR CORRECTION INCOMPATIBLE PROTOCOL EXCESSIVE RETRANSMISSIONS DTR LOSS INACTIVITY TIMEOUT INCOMPATIBLE SPEEDS BREAK DISCONNECT KEY ABORT
LAST TX data rate	The last data rate that the modem on the MAX was transmitting.
HIGHEST TX data rate	The highest data rate that the modem on the MAX was transmitting.
LAST RX data rate	The last data rate that the modem on the MAX was receiving.
HIGHEST RX data rate	The higest data rate that the modem on the MAX was receiving.
Error correction PROTOCOL	The negotiated error correction protocol.
Data COMPRESSION	The negotiated data compression protocol.
Line QUALITY	Probes are sent by each modem to determine the quality of the line and the connection. The range for this variable is 0 to 128. The lower the number, the better the perceived line quality.
Receive LEVEL	This is a representation of the attenuation (weakening) of the modem signal, which is measured in decibels. The decibel rating is translated into a number between 0 and 128 for inclusion in this report. The lower the number, the lower the attenuation of the modem signal.
Highest SPX Receive State	This number relates to an internal DSP state machine in the modem code. There is no practical use of it for most users.
Highest SPX Transmit State	This number relates to an internal DSP state machine in the modem code. There is no practical use of it for most users.

Following is an example of output displayed by `modemdiag`:

```
TERMINATION REASON.......... LINK DISCONNECT
LAST TX data rate........... 26400 BPS
HIGHEST TX data rate........ 26400 BPS
LAST RX data rate........... 24000 BPS
HIGHEST RX data rate........ 24000 BPS
Error correction PROTOCOL... LAPM
Data COMPRESSION............ V42Bis
Line QUALITY................ 032
Receive LEVEL............... 017
Highest SPX Receive State... 67
Highest SPX Transmit State.. 67

TERMINATION REASON.......... LINK DISCONNECT
LAST TX data rate........... 28800 BPS
HIGHEST TX data rate........ 31200 BPS
LAST RX data rate........... 28800 BPS
HIGHEST RX data rate........ 28800 BPS
Error correction PROTOCOL... LAPM
Data COMPRESSION............ V42Bis
Line QUALITY................ 032
Receive LEVEL............... 017
Highest SPX Receive State... 85
Highest SPX Transmit State.. 87
```

MDialout

Description: Displays messages related to modem dialout. It can used in conjunction with the Diagnostic command `modemdrvstate` to get detailed information of outbound modem calls.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `mdialout` at the command prompt.

Example: In the following example, a modem on the MAX prepares to make an outbound modem call, but never receives a dialtone.

```
MDIALOUT-2/4: >> CURR state=Await_Off_Hook, NEW event=Event_Off_Hook
MDIALOUT-2/4: connected to DSP!
MDIALOUT-2/4: rqst tone (14) via channelIndex 0
MDIALOUT-2/4: tone generation started.
MDIALOUT-2/4: >> CURR state=Await_Dial_Tone, NEW event=Event_Dialtone_On
MDIALOUT-2/4: decode timer started.
MDIALOUT-2/4: << NEW state=Await_1st_Digit
MDIALOUT-2/4: enabling tone search, channel index=0, timeslot=0
MDIALOUT-2/4: << NEW state=Await_1st_Digit
MDIALOUT-2/4: >> CURR state=Await_1st_Digit, NEW event=Event_On_Hook
MDIALOUT-2/4: stopping decode timer.
MDIALOUT-2/4: rqst tone (15) via channelIndex 0
MDIALOUT-2/4: disabling tone search, channel index=0
MDIALOUT-2/4: disconnected from DSP.
MDIALOUT-2/4: << NEW state=Await_Off_Hook
MDIALOUT-2/4: >> CURR state=Await_Off_Hook, NEW event=Event_Close_Rqst
MDIALOUT-?/?: << NEW state= <DELETED>
```

MDialSess

Description: Displays all the active modem dialout sessions.

Usage: Enter `mdialsess` at the command prompt.

Example:

```
MAX> mdialsess
```

```
No modem outdial entries.
MAX> mdialsess
entry slot:mdm route port hookDetect DSP:tone:timr:decode state
1    6:4     145   16 pollForOff  n : n  : n  :  n    Await_Off_Hook
```

ModemDrvDump

Description: Displays information about the status of each modem.

Usage: Enter `modemdrvdump` at the command prompt.

Example: Following is a message about modem 0 (the first modem) in the modem card in slot 3 on the MAX. The numbers in brackets indicate number of calls with unexpected open requests, unexpected Rcode events, unexpected release events and unexpected timeouts:

```
MODEMDRV-3/0: Unexp Open/Rcode/Rlsd/TimOut=[0,0,0,0]
```

ModemDrvState

Description: Displays communication to and from the modem driver on the MAX. You can see buffers allocated and what AT command strings are being used to establish modem connections.

You can also see that data is received from the modem in an understandable format. If line quality is poor, the modem driver will attempt to parse incoming data from the modem, but may not be successful.

The command is a toggle that alternately enables and disables the debug display.

Note: Once a connection is negotiated, a series of numerical result codes are passed between the modems. You can see and decipher these result codes to see the negotiated connection rate and error correction/compression protocols. Below is a list of several result codes and their meanings:

```
0 - OK
1 - CONNECT (300 bps)
2 - RING
3 - NO CARRIER
4 - ERROR
5 - CONNECT 1200
6 - NO DIALTONE
7 - BUSY
8 - NO ANSWER
9 - CONNECT 0600
10 - CONNECT 2400
11 - ONNECT 4800
12 - CONNECT 9600
13 - CONNECT 7200
14 - CONNECT 12000
```

```
15 - CONNECT 14400
16 - CONNECT 19200
17 - CONNECT 38400
18 - CONNECT 57600
22 - CONNECT 1200/75 (Models with v.23 support only)
23 - CONNECT 75/1200 (Models with v.23 support only
24 - DELAYED
25 - CONNECT 14400
32 - BLACKLISTED
33 - FAX
34 - FCERROR
35 - DATA
40 - CARRIER 300
43 - CONNECT 16800 (V.34 ONLY)
44 - CARRIER 1200/75 (Models with v.23 support only)
45 - CARRIER 75/1200 (Models with v.23 support only)
46 - CARRIER 1200
47 - CARRIER 2400
48 - CARRIER 4800
49 - CARRIER 7200
50 - CARRIER 9600
51 - CARRIER 12000
52 - CARRIER 14400
66 - COMPRESSION: CLASS 5 (MNP 5)
67 - COMPRESSION: V.42BIS (BTLZ)
69 - COMPRESSION: NONE
70 - PROTOCOL: NONE
77 - PROTOCOL: LAP-M (V.42)
80 - PROTOCOL: ALT (MNP)
81 - PROTOCOL: ALT - CELLULAR (MNP 10) +FC +FCERROR
85 - CONNECT 19200 (V.34 ONLY)
91 - CONNECT 21600 (V.34 ONLY)
99 - CONNECT 24000 (V.34 ONLY)
103 - CONNECT 26400 (V.34 ONLY)
107 - CONNECT 28800 (V.34 ONLY)
151 - CONNECT 31200 (V.34 ONLY)
155* - CONNECT 33600 (V.34 ONLY)
```

Usage: Enter `modemdrvstate` at the command prompt.

Example: Following are examples of a modem call coming into the MAX, and a modem call clearing from the MAX.

```
MAX> modemdrvstate
MODEMDRV debug display is ON
```

Modem 1 on the modem card in slot 3 has been assigned to answer an incoming modem call.

```
MODEMDRV-3/1: modemOpen modemHandle B04E3898, hdlcHandle B026809C, orig 0
```

The modem is idle, so it is available to answer the call.

```
MODEMDRV-3/1: _processOpen/IDLE
```

The next two lines indicate the MAX modem sending the first string.

```
MODEMDRV: Answer String, Part 1 - AT&F0E0
```

A buffer needs to be allocated to send the command out the WAN.

```
MODEMDRV-3/1: _hdlcBufSentFnc: buffer = 2E12EAE0, status = SENT
```

Buffers are allocated for data being received from the WAN.

```
MODEMDRV-3/1: _hdlcBufRcvdFnc: data=2E13ADF0, len=8, parseState[n,v]=[0,0],
status= RCVD
MODEMDRV-3/1: _hdlcBufRcvdFnc: data=2E13BA20, len=5, parseState[n,v]=[0,0],
status= RCVD
```

The MAX modem receives “OK” from the calling modem.

```
MODEMDRV-3/1: data =OK
```

The same process is followed for strings 2 and 3

```
MODEMDRV-3/1: _processTimeout/DIAL_STR2
MODEMDRV: Answer String, Part 2 - AT&C1V0W1X4
MODEMDRV-3/1: _hdlcBufSentFnc: buffer = 2E12EAE0, status = SENT
MODEMDRV-3/1: _hdlcBufRcvdFnc: data=2E13C038, len=2, parseState[n,v]=[0,0],
status= RCVD
MODEMDRV-3/1: data = 0
MODEMDRV-3/1: _processTimeout/DIAL_STR3
MODEMDRV: Answer String, Part 3 -
AT%C3\N3S2=255S95=44S91=10+MS=11,1,300,33600A
```

Now result codes are processed to clarify the characteristics of the connection.

The MAX modem sends a result code of 52, or CARRIER 14400 and the MAX modem receives the same speed from the calling modem.

```
MODEMDRV-3/1: _hdlcBufSentFnc: buffer = 2E12EAE0, status = SENT
MODEMDRV-3/1: data = 5
MODEMDRV-3/1: _hdlcBufRcvdFnc: data=2E13ADF0, len=2, parseState[n,v]=[5,0],
status= RCVD
MODEMDRV-3/1: data = 2
MODEMDRV-3/1: decode= 52
```

Result codes 77 and 67 indicate that V.42 error correction and V.42bis error compression has been successfully negotiated.

```
MODEMDRV-3/1: _hdlcBufRcvdFnc: data=2E13B408, len=1, parseState[n,v]=[2,0],
status= RCVD
MODEMDRV-3/1: data = 7
MODEMDRV-3/1: _hdlcBufRcvdFnc: data=2E13BA20, len=8, parseState[n,v]=[5,0],
status= RCVD
19DEMDRV-3/1: data = 7
MODEMDRV-3/1: decode= 77
MODEMDRV-3/1: decode= 67
```

At this point the modem call is up, and the modem driver is done. The call will be passed to ethernet resources from here.

```
MODEMDRV-3/1: _processRcodeEvent/AWAITING RLSD, mType=5, RLSD=0
MODEMDRV-3/1: _processRlsdChange/AWAITING RLSD = 1
```

Following is the normal sequence of steps for a modem call that is cleared (by either modem). Modem 5 on the modem card in slot 7 of the MAX is freed from the previous call, and it is reinitialized (so it is available for the next call).

```
MODEMDRV-7/5: modemClose modemHandle B04E6F38
MODEMDRV-7/5: _closeConnection:ONLINE, event=3
MODEMDRV-7/5: _processTimeout/INIT
```

MPCMToggle

Description: Displays information related channel addition with Multilink Point-to-Point connections. This does not relate toMP+ or BACP connections. This command displays only information from connections established as MP (RFC1717) connections.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `mpcmtoggle` at the command prompt.

Example:

```
MAX> mpcmtoggle
MPCM debug is now ON
MPCM-432: adding 1 channels
```

MPEntry

Description: Displays information for specified, active MP or MP+ connection. The negotiated options are shown for the connection. This command can be extremely helpful when researching MP or MP+ compatibility issues.

Note: The MpID number that must be entered is an internally generated number. To get a list of all currently assigned MpID numbers on your MAX, go to the Terminal Server and enter

```
SHOW USERS
```

Usage: Enter `mpentry` at the command prompt.

Example: The following example shows an MP+ call (noted as MPP). The End Point Discriminator (used to bundle the channels together) is shown under “bundle id”. In this case, it is the hardware MAC address of the calling device.

```
MAX > mpentry
MpID required
MAX> mpentry 28
MP entry 28 @ B055DE60
MpID  28, Flags: delete No, remote No, ncp Yes, mpp Yes bacp No
bundle id: 15 octets @ B0558BE0
[0000]: 03 00 C0 7B 53 97 07 73 65 63 61 2D 68 73 76
vjInfo @ B0562060
startTime 227521989, mrru: local 1524, peer 1524
send: ifIx 1, count 0, seq 77268 / recv: seq 75046
IF 50, send idle 0, recv idle 1, last seq 75045 mode 0 #chans 1
Head:
Tail
Reassembe packet cnt 0 bad lrg pkts 0
```

MPPCM

Description: Displays MP+ call management information. You can use it in conjunction with `mptoggle`, since each command logs debug from a different place in code, but both display information based on multi-channel connections. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `mppcm` at the command prompt.

Example:

```
MAX> mppcm
MPPCM debug is now ON
```

The following 8 messages indicate a second channel is added to a 1-channel MP+ connection:

```
MPP-5: Event = Utilization, CurrentState = Idle/A
MPP-5: check dynamic says: current = 1, recommended = 2
MPP-5: requesting 1 additional channel(s)
MPP-5: 1 call(s) posssible.
MPP-5: new state is: Add/C
MPP-5: Event = RxAddComplete, CurrentState = Add/C
MPP-5: enterIdleA, AddLock = Yes, RemoveLock = No
MPP-5: new state is: Idle/A
```

The following 12 messages indicate a remote management session is brought up for the MP+ user with MpID 28. You can open a remote session to an MP+ user from the Terminal Server.

```
MPP-28: Event = StartRM, CurrentState = Idle/A
MPP-28: start remote management
MPP-28: new state is: Idle/A
MPP-28: Event = RxRmRsp, CurrentState = Idle/A
MPP-28: remote management response (0)
MPP-28: new state is: Idle/A
MPP-28: Event = RxRmTxReq, CurrentState = Idle/A
MPP-28: new state is: Idle/A
MPP-28: Event = RecvRMM, CurrentState = Idle/A
MPP-28: new state is: Idle/A
MPP-28: Event = StopRM, CurrentState = Idle/A
MPP-28: stop remote management
MAX> mppcm
MPPCM debug is now OFF
```

MPToggle

Description: Displays information about MP and MP+ connections. You can use it in conjunction with `mppcm`, since each command logs debug from a different place in code, but both display information based on multi-channel connections. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `mptoggle` at the command prompt.

Example:

```
MAX> mptoggle
MP debug is now ON
MP-26: sending control message 191
```

```
MP-5: sending control message 76
MAX> mptoggle
MP debug is now OFF
```

NetworkI

Description: Displays information about calls that are first presented to the MAX. From here, the calls will be assigned numerical tags to monitor the connection. After a call passes through this section of code, it typically moves to a call route manager, monitored with the routmgr diagnostic command.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `metworki` at the command prompt.

Example:

```
MAX> networki
NETWORKI debug is now ON
```

The following messages indicate a new call comes into the MAX. This is a normal string of messages for most calls:

```
** CALL 30  RINGING      globDsl  0, channel 23, session 999
```

The call is assigned a callID of 23 and a routeID of 123:

```
NETWORKI: cached callID 30, routeID 123
```

Resources have been allocated for the call. The MAX then begins the process of answering the call:

```
NETWORKI: answering incoming call for route 123
```

A check is made in the call route table to verify the MAX has an entry for this new call:

```
NETWORKI: found callID 30 for route
NETWORKI: found session for route 123
NETWORKI: clearSessionData
NETWORKI: answerCallRequest( 30, 123 )
```

With this next message, the call has been answered. The MAX determines where to route the call from here:

```
** CALL 30  CONNECTED    globDsl  0, channel 23, session  26
NETWORKI: call state connected, callid: 30
networki::pending call, checking for session completeness
NETWORKI: completeTransaction, route 123
NETWORKI: First call completed.  Got base profile, service 1, type 0
NETWORKI: activateChannelList for route 123
clear enter NETWORKI: clearSessionData
clear exit clear done
```

At this point, the call is passed to another function, and networki is no longer in use by this call.

In the following messages, a call is cleared. This is a normal string of messages:

```
NETWORKI: clearSession
NETWORKI: Aborting transaction, route 102
NETWORKI: clearing retries
NETWORKI: callid 6 added to pending clear list
** CALL 6   INACTIVE     globDsl  0, channel 23, session 999
```

NSLookup

Description: Similar to the UNIX nslookup command. When you specify a host name, a DNS request is forwarded, and if the host is found, the corresponding ip address is displayed.

Usage: **nslookup** *host_name*

Example:

```
MAX> nslookup host1
Resolving host host1.
IP address for host drawbridge is 1.1.1.1.
MAX> nslookup 198.4.92.1
Resolving host 198.4.92.1.
MAX> nslookup
Missing host name.
MAX> nslookup nohost
Resolving host nohost.
Unable to resolve nohost!
```

NVRAMClear

Description: Clears non-volatile random access memory (NVRAM). The current system configuration is stored in NVRAM.

Note: A copy of the configuration may also be stored in Flash memory. If you clear NVRAM, the MAX will reset and initialize itself with the configuration it detects in Flash memory. To return your MAX to its factory default settings, you must first clear the configuration in Flash, by using the fclear command, then use nvramclear.

Usage: Enter `nvramclear` at the command prompt.

See Also: FClear

OSFPAvlTree

Description: Displays the entire OSPF AVL tree.

Note: Previously, issuing this command caused the MAX to reset. Do not use this command unless your MAX is running 5.0Ap1 or later software.

Usage: Enter `ospfavltree` at the command prompt.

Example:

```
MAX> ospfavltree
dest             mask            Lptr            Rptr            Myaddr
mrkedDel
0x  1c88c0       0x  ffffff      0x       0      0x       0      0xb0582dc0
0x1
```

Pools

Description: Displays a snapshot of a large selection of memory pools, the size of each pool and specific status about each pool. At the end of the list is a summarization of the total memory allocation in the MAX.

Memory is dynamically allocated to support various tasks. Memory should be freed when it a particular task has been completed. Taking pools snapshots over an extended period of time may help troubleshoot a problem with a memory leak. That is, memory is allocated for a task but never freed.

You should never see the entire quantity of allocated memory (or even any single pool) increase over several snapshots over an extended period of time.

Usage: Enter `pools` at the command prompt.

Example: The number of pools displayed is very large. Size is indicated in Kilobytes. Limit, inUse and hiWat parameters are listed in bytes. The following list gives you sample data. `Size` is the size of each buffer (in bytes). `Limit` is the maximum number of buffers that may be allocated to a pool. `inUse` is the current state of the pool. `hiWat` is the highest amount of pools that have been allocated to a pool since the MAX was brought up.

```
MAX> pools
Pool Name                         size  limit  inUse  hiWat  heapAdrs
AIM-6 Lcd Menu                      8     0       0      0   B044EB00
AIM-6 Status Menu                   2     0       0      0   B044EB00
AcctEvnt                           14     0     160    160   B044EB00
Async V120 Buffer                3056     0       0      2   B030C710
Async X75 Buffer                 3056     0       0      0   B030C710
printf buffers                    172   200     133    133   B044EBC0
total pools:          295
total buffers in use:         6007
total memalloc:       902086
total memfree:       901161
memalloc in use:          925
memalloc failures:            0
memfree failures:           0
memalloc high water:          942
Histogram of memalloc’d memory block sizes:
1088 buffers in range [64,127]
299 buffers in range [128,255]
2 buffers in range [256,511]
11 buffers in range [512,1023]
2 buffers in range [1024,2047]
2 buffers in range [2048,4095]
1 buffers in range [32768,65535]
1 buffers in range [65536,131071]
Total memory in use: 249440 bytes in 1406 buffers
Histogram of free memory block sizes:
14 buffers in range [64,127]
57 buffers in range [128,255]
1 buffers in range [512,1023]
2 buffers in range [2048,4095]
1 buffers in range [8192,16383]
```

```
1 buffers in range [1048576,2097151]
Total free memory: 1515200 bytes in 76 buffers
```

PPPDump

Description: Very similar to the diagnostic command `WANDisplay`. `pppdump` strips out escape characters that are present for asynchronous ppp users (who are dialing in with modems). The escape characters are necessary because of the asynchronous nature of the data stream. Stripping them out simply clarifies the presentation of the data.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the PPPDump command during a period of low throughput.

Usage: pppdump *n*

where *n* is the number of octets to display per frame. Specifying a value of zero will disable the logging of this data.

Example: Following are two examples of display of an asynchronous call, with `wandisplay` and `pppdump`.

The following frames are logged using wandisplay 64. To get the data stream without escape characters, the 0x7D bytes need to be stripped, and the byte following each 0x7D byte needs to be decremented by 0x20

With `wandisplay`, the raw data stream is displayed as:

```
7E FF 7D 23 C0 21 7D 21 7D 21 7D 20 7D 37 7D 22 7D 26 7D 20 7D 2A 7D
20 7D 20 2D 7D 23 7D 26 3A AA 7E
7E FF 7D 23 C0 21 7D 21 7D 21 7D 20 23 7D 20 7D 24 7D 20 7D 20 7D 22
7D 7E
```

Using `pppdump`, the data is automatically converted and displayed:

```
7E FF 03 C0 21 01 01 00 17 02 06 00 0A 00 00 2D 03 06 3A AA 7E 7E
FF 03 C0 21 01 01 00 23 00 24 00 00 02 7E
```

See Also: `wandisplay`, `wannext`, `wanopen`

PPPFSM

Displays changes to the ppp state machine as ppp users connect. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `pppfsm` at the command prompt.

Example: Following is a complete establishment of a ppp session.

```
MAX> pppfsm
PPPFSM state display is ON
PPPFSM-97: Layer 0   State INITIAL     Event OPEN...
PPPFSM-97: ...New State STARTING
PPPFSM-97: Layer 0   State STARTING    Event UP...
PPPFSM-97: ...New State REQSENT
```

```
PPPFSM-97: Layer 1   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 2   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 3   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 4   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 5   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 6   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 7   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 8   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 9   State INITIAL     Event UP...
PPPFSM-97: ...New State CLOSED
PPPFSM-97: Layer 0   State REQSENT     Event RCONFREJ...
PPPFSM: irc_new scr 4
PPPFSM-97: ...New State REQSENT
PPPFSM-97: Layer 0   State REQSENT     Event RCONFACK...
PPPFSM-97: ...New State ACKRECD
PPPFSM-97: Layer 0   State ACKRECD     Event RCONFREQ...
PPPFSM-97: ...New State ACKRECD
PPPFSM-97: Layer 0   State ACKRECD     Event RCONFREQ...
PPPFSM-97: Layer 1   State CLOSED      Event OPEN...
PPPFSM-97: ...New State REQSENT
PPPFSM-97: ...New State OPENED
PPPFSM: PAP Packet
PPPFSM-97: Layer 6   State CLOSED      Event OPEN...
PPPFSM-97: ...New State REQSENT
PPPFSM-97: Layer 4   State CLOSED      Event OPEN...
PPPFSM-97: ...New State REQSENT
PPPFSM-97: Layer 4   State REQSENT     Event RCONFREQ...
PPPFSM-97: ...New State REQSENT
PPPFSM: ccp Packet code 1
PPPFSM-97: Layer 6   State REQSENT     Event RCONFREQ...
PPPFSM-97: ...New State REQSENT
PPPFSM: ccp Packet code 2
PPPFSM-97: Layer 6   State REQSENT     Event RCONFACK...
PPPFSM-97: ...New State ACKRECD
PPPFSM-97: Layer 4   State REQSENT     Event RCONFACK...
PPPFSM-97: ...New State ACKRECD
```

PPPIF

Description: Displays messages relating to each ppp connection. This command is particularly useful in troubleshooting negotiation failures. To help in troublshooting ppp issues, you might want to use `pppif` in conjunction with `pppdump`.

Usage: Enter `pppif` at the command prompt.

Example:

```
MAX> pppif
PPPIF debug is ON
PPPIF: open: routeid 285, incoming YES
```

The following message indicates this is a modem call.

```
PPPIF-110: ASYNC mode
```

Link Compression Protocol (LCP) is negotiated.

```
VJ Header compression is enabled.
PPPIF-110: vj comp on
```

PAP authentication is configured on the MAX and required for access.

```
PPPIF-110: _initAuthentication
PPPIF-110: auth mode 1
PPPIF-110: PAP auth, incoming
PPPIF-110: bypassing async layer
```

LCP has been successfully negotiated and established. Authentication is next.

```
PPPIF-110: Link Is up.
PPPIF-110: pppMpNegUntimeout last 0 layer 0
PPPIF-110: pppMpNegUntimeout last 0 layer 0
PPPIF-110: LCP Opened, local ’Answer’, remote ’’
PPPIF-110: _openAuthentication
PPPIF-110: pppMpNegUntimeout last 0 layer 1
PPPIF-110: Auth Opened
PPPIF-110: Remote hostName is ’my_name’
```

PAP Authentication was successful. Compression Control Protocol (CCP) is negotiated next, along with IP Network Control Protocol (IPNCP).

```
PPPIF-110: opening CCP
PPPIF-110: pppMpSendNeg Pkt
PPPIF-110: pppMpNegTimeout layer 6
```

The user will be given the address 1.1.1.1 from pool 0

```
PPPIF-110: using address from pool 0
PPPIF-110: Allocated address [1.1.1.1]
PPPIF-110: opening IPNCP:
PPPIF-110: pppMpSendNeg Pkt
PPPIF-110: pppMpNegTimeout layer 4
PPPIF-110: pppMpSendNeg Pkt
PPPIF-110: pppMpSendNeg Pkt
PPPIF-110: pppMpNegUntimeout last 0 layer 6
PPPIF-110: pppMpNegUntimeout last 0 layer 4
PPPIF-110: pppMpSendNeg Pkt
PPPIF-110: pppMpSendNeg Pkt
PPPIF-110: pppMpNegUntimeout last 0 layer 4
PPPIF-110: IPNCP Opened to
PPPIF-110: pppMpSendNeg Pkt
PPPIF-110: pppMpNegUntimeout last 0 layer 6
PPPIF-110: CCP Opened
```

IPNCP and CCP have been successfully negotiated. The ppp session has been completely established.

PPPInfo

Description: Displays information about established ppp sessions. The command has little practical use other than as a tool for developmental engineering.

Usage: ppinfo *index* [**all**]

Example:

Syntax element:	**Description:**
index	Selects a particular ppp information table.
all	Displays information on embedded structures

index selects a particular entry ppp information table

all displays information on embedded structures

Example:

```
MAX> pppinfo 1
Ncp[LCP]        = B02B396C
Ncp[AUTH]       = B02B39BC
Ncp[CHAP]       = B02B3A0C
Ncp[LQM]        = B02B3A5C
Ncp[IPNCP]      = B02B3AAC
Ncp[BNCP]       = B02B3AFC
Ncp[CCP]        = B02B3B4C
Ncp[IPXNCP]     = B02B3B9C
Ncp[ATNCP]      = B02B3BEC
Ncp[UNKNOWN]    = B02B3C3C
Mode            = async
nOpen pending   = 0
LocalAsyncMap   = 0
RemoteAsyncMap  = 0
Peer Name       = N/A
Rmt Auth State  = RMT_NONE
aibuf           = 0
ipcp            = B03E502C
vJinfo          = 0
localVjInfo     = 0
bncpInfo        = B03E559C
ipxInfo         = B03E55DC
remote          = no
Bad FCS         = a
```

PPPState

Displays the state of a ppp connection. PPP calls can route (call routing, as opposed to IP or IPX routing) through a MAX differently. The command is a toggle that alternately enables and disables the debug display.

The command has little practical use other than as a tool for developmental engineering.

Usage: Enter `pppstate` at the command prompt.

Example: The following message indicate data is moved directly from the WAN to the Ethernet segment. WAN data can be redirected to other resources (X.75 handler or V.120 handler) before it is ready to be sent to the Ethernet segment.

```
PPP-116: Redirect async wan direct
```

PPTPCM

Description: Displays messages relating to the call management layer of PPTP. You will see messages as calls are routed to the PPTP server by the MAX. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `pptpcm` at the command prompt.

Example: Following are messages from a successful connection.

```
PPTPCM: Connecting to host [1.1.1.1]
PPTPCM-[1.1.1.1]: Event = Local-Start-Request
PPTPCM-[1.1.1.1]: Starting local session
```

In the following message, `status = 0` indicates this was a successful connection.

```
PPTPCM-[1.1.1.1]: Started local session; status = 0
PPTPCM-[1.1.1.1]: _receiveFunc called
PPTPCM-[1.1.1.1]: Event = Remote-Start-Reply
PPTPCM-[1.1.1.1]: Session state changed from Local-Start to Up
```

Following are messages from an unsuccessful connection:

```
PPTPCM-[2.2.2.2]: Event = Local-Start-Request
PPTPCM-[2.2.2.2]: Starting local session
PPTPCM-[0.0.0.0]: Started local session; status = -4
PPTPCM-[0.0.0.0]: EC Start failed
```

PPTPData

Displays the data flowing between the PPTP client and PPTP server. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `pptpdata` at the command prompt.

Example: Following are some samples of pptpdata output:

In the following message, the MAX received a positive acknowledgement from the NT server:

```
PPTPDATA-[1.1.1.1]: Received GRE ACK
```

The MAX received data from the NT server that needs to be forwarded out the WAN port.

```
PPTPDATA-[1.1.1.1]: _dataFromLan
```

The MAX receives a packet from the wan with a good Frame Check Sequence. It is sent to the PPTP server to be processed.

```
PPTPDATA-[1.1.1.1]: Good FCS.  Sending packet to peer
```

This message is a result of an unsuccessful attempt at connecting to an NT PPTP server.

```
PPTPDATA-[2.2.2.2]: pptpDataSessionDown, Session not found
```

PPTPEC

Description: Displays control link messages between the PPTP client and the PPTP server. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `pptpec` at the command prompt.

Example: Following are messages from a successful connection and an unsuccessful attempt.

Successsful connection:

```
PPTPEC-[1.1.1.1]: pptpECSend called
PPTPEC-[1.1.1.1]: New state = Running
PPTPEC-[1.1.1.1]: Event = Send, current state = Running
PPTPEC-[1.1.1.1]: New state = Running
PPTPEC-[1.1.1.1]: Receive callback called
PPTPEC-[1.1.1.1]: Event = Receive, current state = Running
PPTPEC-[1.1.1.1]: New state = Running
```

Unsuccessful attempt:

```
PPTPEC-[2.2.2.2]: pptpECStart called-
PPTPEC-[2.2.2.2]: Event = Start, current state = Stopped
```

PPTPOutCalls

Description: Enables the MAX to make outbound PPTP calls. The command is a toggle that alternately enables and disables the feature.

The default value is ENABLED.

Usage: Enter `pptpoutcalls` at the command prompt.

Example:

```
MAX> pptpoutcalls
PPTPCM outgoing calls are now DISABLED
MAX> pptpoutcalls
PPTPCM outgoing calls are now ENABLED
```

PPTPSend

Description: Sends an Echo Request to the specified NT PPTP server.

Usage: **pptpsend** *ip_address_of_PPTP_server*

Example:

```
MAX> pptpsend 1.1.1.1
PPTPCM: Sending Echo Request to host [1.1.1.1]
```

PPTPStart

Description: Starts a PPTP session with an NT PPTP server with this command. Typically, you will not need to use this command, since sessions are set up and torn down automatically through the MAX configuration menus.

Usage: pptpstart *ip_address_of_NT_PPTP_server*

Example:

```
MAX> pptpstart 1.1.1.1
PPTPCM: Connecting to host [1.1.1.1]
```

PPTPStop

Description: Stops a PPTP session with an NT PPTP server with this command. Typically, you will not need to use this command, because sessions are set up and torn down automatically through the MAX configuration menus.

Usage: pptpstop *ip_address_of_NT_PPTP_server*

```
MAX> pptpstop 1.1.1.1
PPTPCM: Disconnecting from host [1.1.1.1]
```

PRIDisplay

Description: Displays the contents of WAN packets.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the PRIDisplay command during a period of low throughput.

Usage: pridisplay *n*

where ***n*** is the number of octets to display from each WAN packet.

Example: The following sample displays the first 64 bytes from each packet sent to or received from the WAN.

```
MAX> pridisplay 64
Display the first 64 bytes of PRI messages
PRI-RCV-0(task: B0479C00, time: 83251.39) 4 octets @ B0539620
[0000]: 02 01 01 61
PRI-XMIT-0(task: B04B3A40, time: 83251.39) 4 octets @ B050C340
[0000]: 02 01 01 49
PRI-RCV-0(task: B0479C00, time: 83261.64) 4 octets @ B052AF60
[0000]: 02 01 01 61
PRI-XMIT-0(task: B04B3A40, time: 83261.65) 4 octets @ B051EFA0
[0000]: 02 01 01 49
PRI-RCV-0(task: B0479C00, time: 83269.98) 27 octets @ B0539620
[0000]: 02 01 48 60 08 02 1A 7B 05 04 03 80 90 A2 18 04
[0010]: E9 82 83 88 70 05 C1 34 39 39 30
```

```
pridisplay 0
PRI message display terminated
```

Quit

Description: Exits you from Diagnostic mode.

Usage: Enter `quit` at the command prompt.

RadAcct

Description: Displays RADIUS accounting information. `RadAcct` displays very few messages if RADIUS Accounting is functioning correctly. The command is a toggle that alternately enables and disables the debug display.

`RADIF` displays more detailed information for troubleshooting RADIUS-related issues.

Usage: Enter `radacct` at the command prompt.

Example:

```
MAX> radacct
RADACCT debug display is ON
```

A user hangs up and a stop record is generated.

```
RADACCT-147:stopRadAcct
```

The following message indicates there is some load on the network, and the sending of a stop record is delayed. This is not necessarily an indication of a problem.

```
RADACCT-147:_endRadAcct: STOP was delayed
```

RadIF

Description: Displays RADIUS-related messages. This is a powerful diagnostic command, since it displays RADIUS messages the MAX receives as well as sends. Output from `RadIF` in conjunction with running your RADIUS daemon in debug mode (using the `-x` option) gives you virtually all the information you need to focus issues relating to user authentication.

You can also validate IP port that you have configured (or think you have configured), as well as the user name that is being sent by the client.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `radif` at the command prompt.

Example: Following are messages you might see for a successful RADIUS authentication:

```
RADIF: authenticating <8:my_name> with PAP
RADIF: _radiusRequest: id 41, user name <9:my_name>
RADIF: _radiusRequest: challenge len = <0>
```

The IP address and RADIUS DaemonAuthentication port are displayed:

```
RADIF: _radiusRequest: socket 5 len 89 ipaddr 01010101 port 65534-
>1645
```

```
RADIF: _radCallback
RADIF: _radCallback, buf = B05BBFA0
```

The response is sent back from RADIUS. In this case, the user my_name has successfully authenticated. Below is a list of the most common responses:

```
1 - Authentication Request
2 - Positive acknowledgement
3 - Rejection
4 - Accounting request
5 - Accounting response
7 - Password change request
8 - Password change positive acknowledgement
9 - Password change rejection
11 - Access challenge
29 - Password - next code
30 - Password New PIN
31 - Password Terminate Session
32 - Password Expired
```

```
RADIF: _radCallback, authcode = 2
RADIF: Authentication Ack
```

After a user has successfully authenticated, the RADIUS daemon sends the attributes from the user profile to the MAX. The MAX creates the user’s connection profile from these attributes. See the *MAX RADIUS Supplement* for a complete list of attribute numbers.

```
RADIF: attribute 6, len 6, 00 00 00 02
RADIF: attribute 7, len 6, 00 00 00 01
RADIF: attribute 8, len 6, ff ff ff fe
RADIF: attribute 9, len 6, ff ff ff 00
RADIF: attribute 11, len 12, 73 74 64 2e
RADIF: attribute 12, len 6, 00 00 05 dc
RADIF: attribute 10, len 6, 00 00 00 00
RADIF: attribute 13, len 6, 00 00 00 01
RADIF: attribute 244, len 6, 00 00 11 94
RADIF: attribute 169, len 6, 00 00 11 94
RADIF: attribute 170, len 6, 00 00 00 02
RADIF: attribute 245, len 6, 00 00 00 00
RADIF: attribute 235, len 6, 00 00 00 01
```

A RADIUS Accounting Start packet is sent to the RADIUS Accounting Server (using port 1646).

```
RADIF: _radiusAcctRequest: id 42, user name <9:my_name>
RADIF: _radiusAcctRequest: socket 6 len 82 IP cf9e400b port 1646,
ID=42
RADIF: _radCallback
RADIF: _radCallback, buf = B05433C0
RADIF: _radProcAcctRsp: user:<9:my_name>, ID=42
```

RadServDump

Description: Displays the configuration you have configured on the MAX in the Ethernet > Mod Config > RADIUS Server menu.

Usage: Enter `radservdump` at the command prompt.

This does not display any information relating to your configuration of either your RADIUS Authentication server or RADIUS Accounting server.

Example: For the following example, the MAX has been configured with two RADIUS servers, 1.1.1.1 and 2.2.2.2. The port has not been changed from its default of 1700.

```
MAX> radservdump
Rad serv vars: port=1700,sockId=8
0) clients=1010101
1) clients=2020202
2) clients=0
3) clients=0
4) clients=0
5) clients=0
6) clients=0
7) clients=0
8) clients=0
```

RadSessDump

Description: Displays the state of all RADIUS Accounting sessions.

Usage: Enter `radsessdump` at the command prompt.

Example: The `RadSessDump` command displays the following information:

Column Name:	**Description**
`START`	RADIUS Start record. Stop records are not displayed.
`route`	Internal route ID.
`sessID`	Session ID. This is tied with route ID.
`nasPort`	This indicates statistics about the call. The first two digits indicate the type of call. 1 indicates a digital call. 2 indicates an analog call. The next two digits indicate the line on which the call was received. The last two digits indicate the channel on which the call was received.
`authM`	Method of authentication.
`evTime`	Event time. This is a time stamp.

```
MAX> radsessdump
RadActSess:  state route sessID   nasPort authM  evTime
START 00033 227745759 020106 RADIUS 47030
```

RadStats

Description: Displays a compilation of RADIUS Authentication and Accounting statistics.

Usage: Enter `radstats` at the command prompt.

Example:

```
MAX> radstats
RADIUS authen stats:
```

In the following message, `A` denotes *Authentication*. `O` denotes *Other*. There were 612 authentication requests sent and 612 Authentication responses received.

```
0  sent[A,O]=[612,15], rcv[A,O]=[612,8]
```

602 were authenticated successfully and 18 were not.

```
timout[A,O]=[0,6], unexp=0, bad=18, authOK=602
```

IP address of the RADIUS server is 1.1.1.1 and the `curServerFlag` indicates whether or not this RADIUS server is the current authentication server. You can have several configured RADIUS servers, but only one is current at any one time. `0` indicates *no*. `1` indicates *yes*.

```
IpAddress 1.1.1.1, curServerFlag 1
RADIUS accounting stats:
```

The MAX sent 1557 Accounting packets and received 1555 responses (`ACKs` from the Accounting server). This results in `unexp` displaying `2`. This is not necessarily an indication of a problem, but might be the result of the MAX timing out a particular session before receiving an `ACK` from the RADIUS server. Momentary traffic load might cause this condition. *Bad* indicates packets that were formatted incorrectly either by the MAX or by the RADIUS server.

```
0  sent=1557, rcv=1555, timout=0, unexp=2, bad=0
```

Note the Accounting server is different from the Authentication server. The Accounting server and Authentication do not need to be running on the same host, although they can be.

```
IpAddress 2.2.2.2, curServerFlag 1
Local Rad Acct Stats:
```

The following messages can be used to look for traffic congestion problems or badly formatted Accounting packets. Under typical conditions, you may see a few packets whose acknowledgments fail.

The following message indicates if any RADIUS requests have been dropped by the MAX. With this particular message, no requests were dropped. 1557 were sent successfully.

```
nSent[OK,fail]=[1557,0], nRcv=1557, nDrop[QFull,Other]=[0,0]
```

The following message indicates if any RADIUS responses were not received, causing a session timeout. Also, responses that are received by the MAX but do not match any expected responses. The MAX keeps a list of sent requests and expects a response for each request. In the following message, one response was received from the RADIUS server that did not match any of the requests that the MAX had sent out. This might be caused by a corrupted response packet, or because the MAX timed out the session before the response was received.

```
nRsp[TimOut,NoMatch]=[0,1], nBackoff[new,norsp]=[0,0]
```

The following messages display a summarized list of RADIUS server statistics.

```
Local Rad Serv Stats:
unkClient=0
index 0 #Sent = 0, #SendFail=0 badAuthRcv = 0, badPktRcv = 0
```

Reset

Description: Resets the MAX. It restarts and all active connections are terminated. All users are logged out and the default security level is reactivated. In addition, a system reset causes all WAN lines to be temporarily shut down due to loss of signaling or framing information. As the MAX boots, it runs its POST (power-on self tests).

Usage: Enter `reset` at the command prompt.

Example: To reset the unit:

```
MAX> reset
```

See Also: NVRAM

Revision

Description: Displays the serial number of the box.

Usage: Enter `revision` at the command prompt.

Example: In the following message, `6363077` is the serial number of the MAX.

```
MAX> revision
revision = 0 1 10 6363077
```

RoutMgr

Description: Displays information regarding how incoming calls are routed to the Inverse MUltipleXer (IMUX), ethernet or modem ports. `RoutMgr`, when used in conjunction with `networki` can show valuable call routing information. If you have problems with users not connecting, and the incoming calls disconnect within one or two seconds of being presented to the MAX, use `routmgr` and `networki` to look for possible clues.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `routmgr` at the command prompt.

Example:

```
MAX> routmgr
ROUTMGR debug is now ON
ROUTMGR: buildIncomingRoute, port 0, phone <4990>
ROUTMGR: routMgrTask routeID=106, port=0, phone=4990
ROUTMGR-106: _matchPhoneNumber
```

There are no port limitations configured under Net/T1 > Line Config > Line # parameters.

```
ROUTMGR-106: _matchAnyPort
```

The Bearer Capability in the ISDN setup message for the call indicates it is a *voice* call. The call is routed to an available modem:

```
ROUTMGR-106: voice call
ROUTMGR: giving call to lan/hostif
```

At this point, the call is passed to other MAX functions to continue the connection setup.

Following is output from `routmgr` when a call is cleared.

```
ROUTMGR: destroyRoute routeID = 106, cause = CLEAR
ROUTMGR-106: port is 59
ROUTMGR: deallocateCapabilityrouteID=106, capability=ALL
ROUTMGR: route 106 destroyed
```

SaveArea

Description: Displays a portion of NVRAM. It is used in conjunction with either the fatal-history log or the coredump utility. The command has little practical use other than as a tool for developmental engineering.

Note: To enable the capture to the savearea buffer, you must first type `broutersave`. After a reset, you must type `broutersave` again to continue to capture data.

Usage: Enter `savearea` at the command prompt.

Example:

```
MAX> broutersave
BROUTER save data is ON
```

```
MAX> savearea
Date: 03/21/1997.       Time: 13:53:05
Save Area: 256 octets @ DC047A14
[0000]: 03 00 00 00 00 00 00 00 00 00 00 00 C0 BE 78 B0
[0010]: 40 00 00 00 00 00 00 00 E0 F1 0A B0 00 00 00 00
[0020]: 00 00 00 00 40 00 00 00 00 C0 7B 0C 01 59 00 40
[0030]: C7 5A 64 6C 08 00 45 00 00 2B 4D E9 00 00 3B 06
[0040]: C6 DE CE 41 D4 0A C0 A8 08 11 04 7A 00 17 36 C8
[0050]: 6A 7E 00 00 4F C3 50 18 10 00 E0 CD 00 00 1B 5B
[0060]: 43 00 6A 64 66 61 3B 6C 73 66 6A 00 00 01 00 01
[0070]: 03 65 6E 67 06 61 73 63 65 6E 64 03 63 6F 6D 00
[0080]: 00 06 00 01 00 00 0E 10 00 22 04 77 6F 70 72 C0
[0090]: 1E 04 72 6F 6F 74 C0 38 00 00 00 2B 00 00 0E 10
[00a0]: 00 00 07 08 00 0D 2F 00 00 00 0E 10 00 1B 36 DF
[00b0]: 00 1B 36 BC FF FF FF FF FF FF FF FF FF FF FF FF
[00c0]: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
[00d0]: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
[00e0]: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
[00f0]: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

SlotMod

Description: Reflects changes made to the configuration of individual modems or individual modem slot cards. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `slotmod` at the command prompt.

Example: In the following examples, several changes are configured under V.34 Modem > Modem Diag, and the changes are displayed by the `SlotMod` command.

V.34 Modem > Modem Diag > Modem#5 is set to `dis modem`.

```
SLOTMOD: slotProfileChanged slot 8
SLOTMOD: dis/enableModemBySlotItem slot 8 modem# 5=
```

V.34 Modem > Modem Diag > Modem#5 is set to `dis modem+chan.`

```
SLOTMOD: slotProfileChanged slot 8
SLOTMOD: dis/enableModemBySlotItem slot 8 modem# 5
```

V.34 Modem > Modem Diag > Modem#5 is set to `enable modem.`

```
SLOTMOD: slotProfileChanged slot 8
SLOTMOD: dis/enableModemBySlotItem slot 8 modem# 5
```

V.34 Modem > Modem Diag > ModemSlot is set to `dis slot`

```
SLOTMOD: slotProfileChanged slot 8
```

V.34 Modem > Modem Diag > Modem#5 is set to `enable slot`

```
SLOTMOD: slotProfileChanged slot 8
```

V.34 Modem > Modem Diag > Modem#5 is set to `dis slot+chan`

```
SLOTMOD: slotProfileChanged slot 8
```

SNTP

Description: Displays messages related to Simple Network Time Protocol (SNTP). The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `sntp` at the command prompt.

Example: Below are samples of messages you'll see with sntp enabled.

The MAX accepts time from a configured NTP server. The following message displays when the MAX does not accept a supplied time:

```
Reject:li= x stratum= y tx= z
```

The following message indicates that the MAX accepts the time from a specified NTP server.

```
Server= 0 Time is b6dd82ed d94128e
```

The stored time is off by more than one second, it is adjusted.

```
SNTP: x Diff1= y Diff2= z
```

StackLimit

Description: Enables a checking routine that will log a warning to the Fatal-History log whenever any MAX function usage gets within 128 bytes from the end of the stack. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `stacklimit` at the command prompt.

T1Link

Description: Displays messages for the the call control link management functionality for the t1 interface(s).

Usage: Enter `t1link` at the command prompt.

Example:

```
MAX> t1link
T1LINK debug is now ON
MAX> t1link
T1LINK debug is now OFF
```

TCPBind

Description: Displays TCP-related messages. The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `tcpbind` at the command prompt.

Example: In the following example, a user opens a TCP-Clear connection to a local host address:

```
TCPBIND: bind local port 15004 to 1.1.1.1-23, if 0
```

TelnetDebug

Description: Displays messages as telnet connections are attempted or established. The Telnet protocol negotiates several options as sessions are established, and `TelnetDebug` displays the telnet option negotiations.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `telnetdebug` at the command prompt.

Example: The following session shows a successful Telnet connection from the Terminal Server on the MAX to another UNIX host.

```
MAX> telnetdebug
TELNET debug is now ON
```

The far end unix host has been contacted.

```
TELNET-4: TCP connect
```

For this Telnet session, the MAX will support options 24 and 1. The UNIX host should respond with either *DO* or *WONT*:

```
TELNET-4: send WILL 24
TELNET-4: recv WILL 1
```

The UNIX host will support option 1:

```
TELNET-4: repl DO 1
```

The MAX receives a request to support option 3:

```
TELNET-4: recv WILL 3
```

The MAX will support option 3:

```
TELNET-4: repl DO 3
```

The UNIX host will support option 3:

```
TELNET-4: recv DO 3
```

The UNIX host will not support option 24:

```
TELNET-4: recv DONT 24
```

The MAX will not do option 24:

```
TELNET-4: repl WONT 24
```

The UNIX host will support options 1 and 3:

```
TELNET-4: recv WILL 1
TELNET-4: recv WILL 3
```

TLoadCode

Description: Loads software from a UNIX host into Flash memory of the MAX using Trivial File Transfer Protocol (TFTP). The TFTP host can be accessed from the Ethernet interface or across the WAN. The MAX needs to be reset to load the the uploaded code, since the MAX must load the code from Flash memory into DRAM.

Although the MAX may experience a small performance degradation during the file transfer, it will be fully functional during the file download process.

When using `TLoadCode`, the current configuration of the MAX will be saved to Flash memory . This means that manual reconfiguration should not be necessary, which is a requirement when loading software through the serial connection.

You'll see a sequence of dots printed to the screen indicating progress of the transfer. Each dot indicates approximately 512 bytes of data has transferred.

Note: If the TFTP transfer is interrupted or the checksum of the uploaded file is incorrect, the new code does not load when the MAX is rebooted. The MAX reloads its previous version of code. Also, if the new code *is* uploaded at boot time, an FRESTORE is performed to load the configuration stored in Flash memory. If this occurs, the MAX reboots again to properly initialize the configuration.

Usage: **tloadcode** *name_or_ip_address_of_tftp_server filename*

Example:

```
MAX> tloadcode
usage: loadcode host file
> tloadcode 1.1.1.1 mhpt1.bin
saving config to flash
................................
.
loading code from 1.1.1.1
file mhpt1.bin...
.......................................................................
.....................................................
............................
```

TRestore

Description: Restores a saved configuration from a TFTP host to Flash memory on the MAX. You need to manually reboot the MAX to load the restored configuration from Flash memory into dynamic RAM.

Usage: **trestore** *name_or_ip_address_of_tftp_server filename*

Example:

```
MAX> trestore 1.1.1.1 config.txt
restoring configuration from 1.1.1.1:69
file config.txt...
```

TSave

Description: Saves the MAX configuration in Flash memory to a TFTP server. You need to perform the FSAVE command if you want to save your currently running configuration. FSAVE will save your configuration to Flash memory.

Usage: **tsave** *name_or_ip_address_of_tftp_server filename*

Example:

```
MAX> tsave 1.1.1.1 config.txt
saving configuration to 1.1.1.1:69
file config.txt...
```

TSShow

Description: Displays uptime and revision information of the MAX. The Terminal Server commands `Show uptime` and `Show revision` display the same information.

Usage: **`tsshow`** `[` **`?`** `] [` **`uptime`** `] [` **`revision`** `]`

Syntax element:	Description:
`?`	List all options.
`uptime`	Display system uptime.
`revision`	Display software and version currently running.

Example: Following are some samples of output you’ll see by using tsshow:

```
MAX> tsshow
Show what? Type ’tsshow ?’ for help.
MAX> tsshow ?
tsshow ?            Display help information
tsshow uptime       Display system uptime.
tsshow revision     Display system revision.
MAX> tsshow uptime
system uptime: up 36 days, 9 hours, 59 minutes, 27 seconds
MAX> tsshow revision
max7 system revision: ti.m40 5.0Ap4
```

Update

Description: Modifies optional functionality of the MAX. To enable some options, you need to be given a set of *hash codes* (supplied by an Ascend representative) that will enable the functionality in your MAX. After each string is entered, you see the word *complete* displayed to indicate the hash code was successfully accepted into the MAX

If you enter `update` without a text string modifier, the MAX displays a list of current configuration information.

Usage: **update** [text_string]

Example:

```
MAX> update
Host interfaces: 4
Net interfaces: 4
Port 1 channels: 255
Port 2 channels: 255
Port 3 channels: 255
Port 4 channels: 255
Field features 1: 182
Field features 2: 33
Field features 3: 54
Protocols: 1
```

```
MAX> update 5 1023 12321312312312321
```

The following two messages indicates the text strings were entered incorrectly.

```
update command: invalid arg 3!
update command: disallowed
```

The following message indicates the MAX accepted the update string.

```
update command: command complete.
```

WANDisplay

Description: Displays all packets received from and sent to all WAN interfaces. This information might be very helpful in ppp negotiation problems since `WANDisplay` outputs raw data displaying what the MAX is receiving from the remote device and what the MAX is sending to the remote device.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the WANDisplay command during a period of low throughput or to use `wandsess`, `wanopen` or `wannext` to focus the display and depending on the types of information you need to gather.

Usage: **`wandisplay`** *number_of_octets_to display_from_each_packet*

Enter `wandisplay 0` to disable the logging of this information.

Example: Following are several examples of outpu displayed by `WANDisplay`. Note the bytes are displayed in hexadecimal format.

```
MAX> wandisplay 24
Display the first 24 bytes of WAN messages
> RECV-272:: 1 octets @ 5E138F74
[0000]: 0D
RECV-272:: 13 octets @ 5E13958C
[0000]: 0A 41 63 63 65 70 74 3A 20 69 6D 61 67
XMIT-276:: 1011 octets @ 2E12D8A4
[0000]: 7E 21 45 00 03 EE 54 2B 40 00 37 06 BA 09 CF 2B
[0010]: 00 86 D0 93 91 90 1A 0A
```

```
MAX> wandisplay 0
WAN message display terminated
```

See Also: `wandsess`, `wanopen`, `wannext`

WANDSess

Description: Similar to `WANDisplay`, but `WANDSess` displays only incoming and outgoing packets for a specific user. `WANDSess` is particularly helpful for troubleshooting a MAX with several simultaneous active connections, which makes the volume of output from commands like `WANDisplay` not as effective for troubleshooting issues for particular users. `WANDSess` is a filter to let you focus your troubleshooting.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the WANDSess command during a period of low throughput.

Usage: **wandsess** *user_name_or_profile_name number_of octets_to_display_from each_packet*

Enter `wandsess` *`user_name_or_profile_name`* `0` to disable the logging of this information.

Example: Notice that the only difference in output between `WANDSess` and `WANDisplay` is that the user is displayed in messages when using `WANDSess`. The data is identical in content. `WANDSess` displays no data from any other sessions. In the following example, only data from `gzoller`'s session is displayed.

```
MAX> wandsess gzoller 24
RECV-gzoller:300:: 1 octets @ 3E13403C
[0000]: 7E 21 45 00 00 3E 15 00 00 00 20 7D 31 C2 D2
RECV-gzoller:300:: 15 octets @ 3E133A24
[0000]: D0 7D B3 7D B1 B3 D0 7D B3 90 02 04 03 00 35
XMIT-gzoller:300:: 84 octets @ 3E12D28C
[0000]: 7E 21 45 00 00 4E C4 63 00 00 1C 7D 31 17 5F D0
[0010]: 93 90 02 D0 93 91 B3 00
MAX> wandsess gzoller 0
MAX>
```

WANNext

Description: Similar to `WANDisplay`, but the MAX displays only incoming and outgoing packets for the next user to successfully authenticate after you enter `WANNext`. As with `WANDSess`, the output is the same as WANDisplay, but is filtered to show you only data from a single user.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the WANNext command during a period of low throughput.

Usage: **wannext** *number_of_octets_to_display_from_each_packet*

Enter `WANNext 0` to disable the logging of this information.

WANOpening

Description: Similar to `WANDisplay`, but this command shows only the opening incoming and outgoing packets for all users during the establishment of their ppp sessions. `WANOpening` is particularly helpful if you are troubleshooting connection problems where users seem to connect to the MAX, but are disconnected within a few seconds. Again, the output from this command is very similar to `WANDisplay`, but only displays packets for sessions to the point where the connection has been completely negotiated.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the WANOpening command during a period of low throughput.

Usage: wanopening *number_of_octets_to_display_from_each_packet*

Enter `WANOpening 0` to disable the logging of this information.

WANToggle

Description: Displays messages from the WAN drivers on the MAX. It displays the state of calls that have been passed from the MAX call routing routines. The connection is prepared to be passed to the Ethernet drivers.

If you enter the command while traffic through your MAX is heavy, the resulting amount of output can make it tedious to find the information you're looking for. The screen might even display the message `----- data lost -----`, which just means that not all the output can be displayed on the screen.

You might prefer to use the WANToggle command during a period of low throughput.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `wantoggle` at the command prompt.

Example: Following shows you typical output from a modem call into the MAX. After the incoming call is determined to be an analog call, a modem is directed to answer it.

```
WAN-389: wanOpenAnswer
WAN-389: modem redirected back to wan
WAN-389: Startup frame received
WAN-389: Detected unknown message
WAN-389: Detected ASYNC PPP message
WAN-389: wanRegisterData, I/F 58
```

When the call is cleared, the following two messages are displayed.

```
WAN-389: wanCloseSession, I/F 58
```

The modem clears the call a split second before the software released its resources.The software does a check on the modem, which has already been released. This message is not an indication of a problem.

```
WAN-??: no modem assoc w WanInfo
```

WDDialout

Description: Displays the specific packet that caused the MAX to dial out. This is a particularly helpful if you are seeing the MAX dial out when it shouldn't. You can use `WDDialout` information to design a filter to keep the MAX from dialing out because of this packet.

The command is a toggle that alternately enables and disables the debug display.

Usage: Enter `wddialout` at the command prompt.

Example: In the following message, the output dislays a date/time stamp, the phone number being dialed, as well as the packet that caused the MAX to dial out.

```
Date: 01/01/1990.     Time: 00:51:56
Cause an attempt to place call to 18185551234
WD_DIALOUT_DISP: chunk D7BA6 type OLD-STYLE-PADDED.
: 60 octets @ F3050
[0000]: 09 00 07 ff ff ff 00 05 02 e8 14 0d 00 24 aa aa
[0010]: 03 00 00 00 80 f3 00 01 80 9b 06 04 00 01 00 05
[0020]: 02 e8 14 0d 00 ff 00 f7 00 00 00 00 00 00 00 ff
[0030]: 8e 01 00 00 00 00 00 00 00 00 00 00
MAX> wddialout
WANDATA dialout display is OFF
```

PPP Decoding Primer

Many of the diagnostic commands display raw data. This Primer is designed to assist you in decoding PPP, MP, MP+ and BACP negotiations. The negotiations can be logged with the Diagnostic commands `PPPDump`, `WANDisplay`, `WANDSess`, `WANNext` or `WANOpen`. For more detailed information than this guide provides, refer to specific RFCs. A partial list of pertinent RFCs appears at the end of this guide.

Breaking down the raw data

An important concept to keep in mind is that each device negotiates PPP independently, so the options might be identical for each direction of the session.

During PPP negotiation, frame formats in the various protocols are very similar. THey share the following characteristics:

- `FF 03` indicating it is a PPP frame.
- A two-byte Protocol Identifier.
- A one-byte Packet Format ID number
- A one-byte ID number.
- A two-byte length.
- Options for the protocol.

Below is a table of the most common protocols you'll see in Ascend diagnostic traces:

Identifier:	**Description:**
`C0 21`	Link Control Protocol (LCP)
`C0 23`	Password Authentication Protocol (PAP)
`C2 23`	Challenge Handshake Authentication Protocol (CHAP)
`80 21`	Internet Protocol (IP)
`80 29`	Appletalk Protocol
`80 2B`	Novell's Internetwork Packet Exchange (IPX)
`80 31`	Bridging PDU
`80 FD`	Compression Control Protocol (CCP)

Following are the packet formats:

Packet Format ID	**Description**
`01`	Configure Request
`02`	Configure Acknowledgment
`03`	Configure Non-Acknowledgment
`04`	Configure Reject
`05`	Terminate Request
`06`	Terminate Acknowledgment
`07`	Code Reject
`08`	Protocol Reject
`09`	Echo Request
`0A`	Echo Reply
`0B`	Discard Request

Note: If a packet received from the wan fails the Cyclic Redundancy Check (CRC) the display is similar to the following, where `RBAD` denotes Received BAD:

```
RBAD-27:: 8712 octets @ 26CFE8
[0000]: fe dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd
[0010]: dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd
[0020]: dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd
[0030]: dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd
```

Annotated Traces

Use the following traces as guides to help you decode other traces.

LCP Configure Request - MP+, MRU of 1524, MRRU of 1524 and End Point Discriminator using the device's MAC address:

```
XMIT-3:: 29 octets @ 2C2E94
[0000]: ff 03 c0 21 01 01 00 19 00 04 00 00 01 04 05 f4
[0010]: 11 04 05 f4 13 09 03 00 c0 7b 4c e0 4c
```

This is a second LCP Configure Request from the same device. Everything in the packet is identical to the previous packet, except the ID number has incremented from 01 to 02:

```
XMIT-3:: 29 octets @ 2C2E94
[0000]: ff 03 c0 21 01 02 00 19 00 04 00 00 01 04 05 f4
[0010]: 11 04 05 f4 13 09 03 00 c0 7b 4c e0 4c
```

LCP Configure Request - CHAP authentication, Magic number

```
RECV-3:: 19 octets @ 2BEB8C
[0000]: ff 03 c0 21 01 60 00 0f 03 05 c2 23 05 05 06 4e
[0010]: 36 c9 05
```

LCP Configure Acknowledgment - This device will authenticate using CHAP. The Magic number is also acknowledged:

```
XMIT-3:: 19 octets @ 2C2E94
[0000]: ff 03 c0 21 02 60 00 0f 03 05 c2 23 05 05 06 4e
[0010]: 36 c9 05
```

LCP Configure Reject - MP+, MRU of 1524, MRRU of 1524 and End Point Discriminator.

This rejection shows two things. It shows that the remote side does not support MP+ or MP, since MP+ and the MRRU were rejected. This will have to be a PPP connection. Also, since the MRU of 1524 was rejected, the default of 1500 is assumed. There needs to be an MRU, so a rejection of a given value only means to use the default value.

At this point, this device will need to retransmit another LCP Configure Request, removing all the rejected options.

```
RECV-3:: 29 octets @ 2BF1A4
[0000]: ff 03 c0 21 04 02 00 19 00 04 00 00 01 04 05 f4
[0010]: 11 04 05 f4 13 09 03 00 c0 7b 4c e0 4c
```

LCP Configure Request - Note all values that were previously rejected are no longer in the packet:

```
XMIT-3:: 8 octets @ 2C2E94
[0000]: ff 03 c0 21 01 04 00 04
```

LCP Configure Acknowledgment -

```
RECV-3:: 8 octets @ 2BF7BC
[0000]: ff 03 c0 21 02 04 00 04
```

At this point, since both sides have transmitted LCP Configure Acknowledgments, LCP is up and the negotiation moves to the authentication phase.

This device receives a CHAP challenge from the remote end:

```
RECV-3:: 21 octets @ 2BFDD4
[0000]: ff 03 c2 23 01 01 00 11 04 4e 36 c9 5e 63 6c 63
[0010]: 72 34 30 30 30
```

This device transmits its encrypted user name and password:

```
XMIT-3:: 36 octets @ 2C2E94
[0000]: ff 03 c2 23 02 01 00 20 10 49 b8 e8 54 76 3c 4a
[0010]: 6f 30 16 4e c0 6b 38 ed b9 4c 26 48 5f 53 65 61
[0020]: 74 74 6c 65
```

The remote device sends a CHAP Acknowledgment:

```
RECV-3:: 8 octets @ 2C03EC
[0000]: ff 03 c2 23 03 01 00 04
```

At this point, the negotiation moves from authentication to negotiation of Network Control Protocols (NCPs). Ascend supports Bridging Control Protocol (BCP), IPCP, IPXCP and ATCP.

IPCP Configure Request - Van Jacobsen Header Compression, IP address of 1.1.1.1

```
RECV-3:: 20 octets @ 2C0A04
[0000]: ff 03 80 21 01 e3 00 10 02 06 00 2d 0f 00 03 06
[0010]: 01 01 01 01
```

BCP Configure Request -

```
RECV-3:: 8 octets @ 2C101C
[0000]: ff 03 80 31 01 55 00 04
```

IPCP Configure Request - IP address of 2.2.2.2

```
XMIT-3:: 14 octets @ 2C2E94
[0000]: ff 03 80 21 01 01 00 0a 03 06 02 02 02 02
```

IPCP Configure Reject - Van Jacobsen Header Compression. The remote device should send another IPCP Configure Request and remove the request to do VJ Header Compression:

```
XMIT-3:: 14 octets @ 2C2E94
[0000]: ff 03 80 21 04 e3 00 0a 02 06 00 2d 0f 00
```

BCP - Protocol Reject. This local device is not configured to support bridging.

```
XMIT-3:: 8 octets @ 2C2E94
[0000]: ff 03 80 31 08 55 00 04
```

IPCP Configure Acknowledgment

```
RECV-3:: 14 octets @ 2C1634
[0000]: ff 03 80 21 02 01 00 0a 03 06 01 01 01 01
```

IPCP Configure Request - Note VJ Header Compression is not requested this time.

```
RECV-3:: 14 octets @ 2C1C4C
[0000]: ff 03 80 21 01 e4 00 0a 03 06 02 02 02 02
```

IPCP Configure Acknowledgment

```
XMIT-3:: 14 octets @ 2C2E94
[0000]: ff 03 80 21 02 e4 00 0a 03 06 01 01 01 01
```

At this point, a PPP connection has been successfully negotiated. The caller was successfully authenticated by means of CHAP and IPCP was the only successfully configured NCP. IPX, Appletalk and bridging will not be supported during this session.

Below are two packets used in determining link quality:

LCP Echo request packet

```
RECV-3:: 16 octets @ 2BEB8C
[0000]: ff 03 c0 21 09 01 00 0c 4e 36 c9 05 00 00 00 00
```

LCP Echo Response

```
XMIT-3:: 16 octets @ 2C2E94
[0000]: ff 03 c0 21 0a 01 00 0c 00 00 00 00 00 00 00 00
```

Example of MP+ call negotiation

LCP Configuration Request - MP+, MRU of 1524, MRRU of 1524, End Point Discriminator using the device's MAC address:

```
XMIT-31:: 29 octets @ D803C
[0000]: ff 03 c0 21 01 01 00 19 00 04 00 00 01 04 05 f4
[0010]: 11 04 05 f4 13 09 03 00 c0 7b 5c d3 71
```

LCP Configure Request - MP+, MRU of 1524, PAP authentication is required. MRRU of 1524, End Point Discriminator using the device's MAC address:

```
RECV-31:: 33 octets @ D4FBC
[0000]: ff 03 c0 21 01 01 00 1d 00 04 00 00 01 04 05 f4
[0010]: 03 04 c0 23 11 04 05 f4 13 09 03 00 c0 7b 53 f0
[0020]: 7a
```

LCP Configuration Acknowledgment -

```
RECV-31:: 29 octets @ D55CC
[0000]: ff 03 c0 21 02 01 00 19 00 04 00 00 01 04 05 f4
[0010]: 11 04 05 f4 13 09 03 00 c0 7b 5c d3 71
```

LCP Configuration Acknowledgment -

```
XMIT-31:: 33 octets @ D803C
[0000]: ff 03 c0 21 02 01 00 1d 00 04 00 00 01 04 05 f4
[0010]: 03 04 c0 23 11 04 05 f4 13 09 03 00 c0 7b 53 f0
[0020]: 7a
```

At this point, LCP is up. Next is the authentication phase. The local device agreed to authenticate using PAP, so it should transmit its user name and password. Note that it is not encrypted, and user name and password can be decoded very easily:

PAP Authentication Request - User name is shown in hexadecimal and must be converted to ascii. User name is 0x6a 0x73 0x6d 0x69 0x74 0x68 (jsmith) and password is 0x72 0x65 0x64 (red):

```
XMIT-31:: 20 octets @ D803C
[0000]: ff 03 c0 23 01 01 00 10 06 6a 73 6d 69 74 68 03 72
[0010]: 65 64
```

PAP Authentication Acknowledgment -

```
RECV-31:: 9 octets @ D5BDC
[0000]: ff 03 c0 23 02 01 00 05 00
```

Authentication is successful. Final negotiation determines protocols to be supported over the link.

Note: MP+ was negotiated, and both devices begin sending MP+ packets from here. The data portion of the packet is identical to PPP, but there is an 8-byte MP+ header instead of the 2-byte PPP header:

In the following packet, `00 3d` is the designation for a Multilink packet. The next byte designates whether this packet is fragmented. The next three bytes are the sequence number. You’ll see them increment by one for each packet sent or received.

Next, the 80 31 01 designates this as a BCP Configure Request:

```
RECV-31:: 20 octets @ D61EC
[0000]: ff 03 00 3d c0 00 00 00 80 31 01 01 00 0a 03 03
[0010]: 01 07 03 00
```

BCP Configure Request:

```
XMIT-31:: 20 octets @ D803C
[0000]: ff 03 00 3d c0 00 00 00 80 31 01 01 00 0a 03 03
[0010]: 01 07 03 00
```

BCP Configure Acknowledgment:

```
XMIT-31:: 20 octets @ D864C
[0000]: ff 03 00 3d c0 00 00 01 80 31 02 01 00 0a 03 03
[0010]: 01 07 03 00
```

BCP Configure Acknowledgment:

```
RECV-31:: 20 octets @ D67FC
[0000]: ff 03 00 3d c0 00 00 01 80 31 02 01 00 0a 03 03
[0010]: 01 07 03 00
```

BCP is up and the session begins sending bridged traffic. No routed protocols were negotiated.

The following packets are sent as part of the MP+ protocol. They are sent at one-second intervals. These packets are used by each unit to validate the existence of the link. It gives the devices a secure way to determine whether the link is still up, even if there is no data traffic passing between the devices.

```
RECV-31:: 8 octets @ D5BDC
[0000]: ff 03 00 3d c0 00 00 05
XMIT-31:: 8 octets @ D803C
[0000]: ff 03 00 3d c0 00 00 04
RECV-31:: 8 octets @ D61EC
[0000]: ff 03 00 3d c0 00 00 06
XMIT-31:: 8 octets @ D803C
[0000]: ff 03 00 3d c0 00 00 05
```

The following RFCs provide more detail about the subjects listed in their titles:

Identifier	Title
RFC1378	PPP AppleTalk Control Protocol (ATCP)
RFC1552	PPP Internetwork Packet Exchange Control Protocol (IPXCP)

Identifier	Title
RFC1638	PPP Bridging Control Protocol (BCP)
RFC1661	Point-to-Point Protocol (PPP)
RFC1934	Ascend's Multilink Protocol Plus (MP+)
RFC1962	PPP Compression Control Protocol (CCP)
RFC1974	PPP Stac LZS Compression Protocol
RFC1989	PPP Link Quality Monitoring
RFC1990	PPP Multilink Protocol (MP)
RFC1994	PPP Challenge Handshake Authentication Protocol