EMU User Guide

This manual describes the installation, use and interpretation of the displays produced by the EMU system

October 2000

Preface

preface

This manual is divided into four sections:

- Introduction and history
- Installation
- User guide
- Limitations and further information

Contents

Preface v

preface vii

Chapter 1 Introduction

1.1 System Overview 1–2
1.2 A Brief History of EMU 1–2

Chapter 2 Installation

Chapter 3 User Guide

3.1 Starting the system 3–1
3.2 Stopping the system 3–1
3.3 User Interface 3–3
3.3.1 Menu system overview 3–3
3.3.1.1 Menu Security 3–3
3.3.1.2 Getting help 3–3
3.3.2 Main menu 3–3
3.3.3 Device menu 3–3
3.3.4 Device Query menu 3–4
3.3.5 Device display format 3–4
3.3.5.1 Parameter Display 3–5
3.3.5.2 Hidden Menu 3–5
3.3.6 Reporting menu 3–6
3.3.7 Network menu 3–7

3.3.8 System menu . . . 3–7
3.3.8.1 Monitor Listener . . . 3–7
3.3.8.2 Monitor PSRs . . . 3–9
3.3.8.3 Monitor Probes . . . 3–10
3.3.8.4 View Sections . . . 3–10
3.3.8.5 Dump_Database . . . 3–12
3.3.8.6 Trace . . . 3–12
3.3.8.7 Scanner . . . 3–12
3.3.8.8 Translate Tables . . . 3–12
3.4 Alert . . . 3–13
3.4.1 Main Menu . . . 3–13
3.4.2 Alert Format . . . 3–14
3.5 Utilities . . . 3–14
3.5.1 Error Log . . . 3–14
3.5.2 PARAMTBL . . . 3–15
3.5.3 Operations . . . 3–15
3.5.3.1 Parameters . . . 3–16
3.5.3.2 PARAMTBL Summary . . . 3–18
3.5.4 MIB compiling and Registration . . . 3–18
3.5.4.1 Usage Instructions . . . 3–19
3.5.5 MIBWALKER . . . 3–20
3.6 Errors and Omissions . . . 3–21
3.7 Hints . . . 3–22
3.7.1 Further processing of Reports . . . 3–22
3.7.2 Seeding the IP database . . . 3–23
3.7.3 Finding a MOP console user . . . 3–23

Index

Figures

3–1 Menu System Schematic . . . 3–2

Tables

3–1 Report Output . . . 3–6
3–2 PSR Status Bits . . . 3–8
3–3 EMU ASN.1 Private DataTypes . . . 3–18

Chapter 1
Introduction

EMU (Ethernet Monitoring Utility) is a system that listens to the attached Ethernet and records all addresses appearing on the cable. For each address found it also records the protocols enabled on that address and for selected protocols it can extract information useful for network management purposes. It then uses this database to locate other nodes in the wide area network and initiate monitoring functions.

Some of the basic features of the system follow:

- The system is fully automatic. The system builds and formats the database without operator assistance or intervention. All monitoring and control subfunctions start and run without intervention and in fact, no operator assistance is ever required for normal operation.
- An alert screen provides the operator with timely reports on new addresses appearing and for selected protocols, changes to configuration parameters, fault reports and performance reports.
- A number of system management utilities are provided. These utilities are all menu driven and self explanatory.
- A versatile reporting command allows the operator to extract selected information to a user defined and formatted file.This may be useful for input into other management tools and/or reports.
- A comprehensive query facility allows operators to view the database contents. Nodes can be selected by many criteria.

For details on what data is available and instructions on how to extract it see the user guide section.

1.1 System Overview

This is the 5th version of EMU and it's 3rd major rewrite. While all the original goals and much of the code was retained, both the architecture and the engineering have been greatly expanded to reflect the ongoing changes in the networking world. Most importantly, the principles that drove the design remained:

1. Full automation. The system is designed to assist the user in monitoring and managing the network, not the reverse as in many current systems. All data in the database is derived from the network; there is no user input possible at this level. This ensures data integrity and eliminates the burden on the user.
2. Full integration. The system maintains full cross referencing of all addresses where there is sufficient information. While it is necessary to implement each network as a separate entity, the user sees only single devices attached to many networks, or many networks with devices attached. This arrangement allows the system to more effectively monitor any single device using the multiple protocols it supports and allows more accurate and complete reporting of any device.
3. The system itself should require no maintenance. All processes are self configuring, self monitoring and self tuning.
4. The complexity of the network itself should be hidden. The system should be fully usable by non-network personnel to extract useful information; The simple question "How is the network performing?" should be answered simply, completely and with qualification.
5. The system will acquire and store sensitive information. It is essential that this system incorporates controls that protect this information, or even the fact of it's existence from unauthorised disclosure.

1.2 A Brief History of EMU

EMU started life as a project to automate an early DECnet management system provided by Digital. Once this was completed, it was expanded out to include some of the other Digital tools then available and the result was an unwieldy mass of DCL procedures that none the less proved the possibility of unburdening the network manager from the task of managing the system that was intended to save time and effort. This set the first principle of design: A network management system should tell the network manager about the network, not the reverse. One of the early problems found on all systems, including EMU was the need to poll the managed devices constantly in order to acquire the information necessary to manage it. In extreme cases substantial resources are used just to manage and often the tool intended to solve problems became the problem. In order to alleviate this the concept of listening was developed. All major network schemes implement a protocol that is used to maintain the integrity of the network. In most cases they pass information about nodes or networks that can be reached, nodes that have stopped responding and in some cases, service names and other useful information. By listening to this conversation, a fairly complete picture of any network can be gained without adding data to the network. This, though much more difficult to implement became the second principle: A network management system should be part of the solution, not part of the problem.

The next stage was to recode the logic into programmes and it was about that time that the major systems that we use today came out (SunNet manager, HP Openview, Netview and DECMCC) and the EMU effort would have ended there had any of these systems adhered to these (to us!) most basic of principles. The time saving achieved by using the system to monitor the network was used up by having to monitor the system itself. A further problem was that while all the manufacturers claimed multivendor capability the claims in some cases were a bit thin and in no case was integration achieved. For example DECMCC had a DECnet module, a TSM (Terminal Servers) and an SNMP module all which worked but internally did not intercommunicate. Thus should a node fail all the individual modules reacted independently without regard for each other. It was apparent at the time that this failure not only complicated the interface and the network manager's task but had this integration been included, the system could then use it's understanding of the various protocols present to enhance it's usefulness. For example, should the DECnet module report a failure on a node, the first action the manager would normally take is to access the node on an alternate protocol to determine if the node is down or if DECnet has failed. As this is a routine effort, this routine should be built into the system so that failures can be reported more completely and in context. This became the third design principle: EMU would take 2 views of the network equally. The network as a collection of devices cooperating on 1 or more networks and as a protocol in which many devices cooperate. It is a subtle but essential distinction.

The first 3 versions became experimental and proof of concept and for version 4 an architecture was drawn up to guide a more formal effort at implementing what was learned so far. Version 4 was successful in that it ran without major change for about 3 years and it even managed some commercial success but changes in the networking world began to overtake it. The most basic flaw was that EMU assumed (accurately at the time) that each device on the network had exactly 1 LAN address and this was used as the key in all databases. As this became less and less true and EMU changed to accommodate, it was clear that the original design was inadequate. A decision to re-engineer the system became an opportunity to re-architect it and remove some of the built in restrictions. Most notable among these restrictions was the over adherence to listening and not talking on the network. It was clear that listening was a good mechanism to build the initial picture but that polling was needed to complete it.

To shorten this narrative to it's essentials, suffice to say that in the 3 years since this decision was taken, 18 months was spend exclusively on the architecture document with the remaining time spend on coding and testing the implementation. The current version implements about 60% of the architecture.

Chapter 2
Installation

1. Create the root directory for the EMU system. This can be on any disk at any level. The system resides in various directories under this to a level of 3 max so make sure you do not create the root at a depth greater than 4.
   $CREATE/DIR disk:[emu]
2. The files created in the next section will consume about 75,000 blocks of disk space. The running system will (very dependant on the size of your network) will consume a further 50,000 - 100,000 blocks.
3. Set default to this directory and backup the saveset:
   $BACKUP/LO Tape:[...]*.*;*
4. Edit [.SRC]EMU_LOGICALS.COM and replace the first executable line with the device and directory you created in step 1. Remember to include the final '.' in the directory spec as this will become a rooted logical.
5. Execute EMU_LOGICALS.COM:
   $ @[.SRC]EMU_LOGICALS
6. Ensure you got the expected effect:
   $SHO LOG EMU*
   You should see a list of directories under the directory you created in step 1.
   $DIR EMU5_SRC:
   You should see a list of .MAR files
7. All being well set default to EMU5_SRC: and execute BUILD:
   @BUILD
   Answer 'YES' to the question 'Compile all subroutines (YES/n)'
8. The system will then build the necessary libraries
9. A number of warning messages will be displayed during compilation. These can be ignored as long as the process continues to completion.

10. When finished (Maybe 20 min or so) answer 'YES' to the question
    Compile all programs? (YES/n)
    This will build the system on the current version of VMS on the current architecture.

11. A number of warning messages will be displayed during compilation. These can be ignored as long as the process continues.

12. All EMU processes run under the system account. Ensure this account has the required quotas. A copy of one that works appears at the end of this list.

13. Ensure LAT, DECnet and UCX are all started.

14. Ensure LATCP can see all services:
    $MCR LATCP SET NODE/USER=ENA=0-255
    The LAT processor will see all LAT devices on the net but updates from the local (VMS created) database. If the above is not done, EMU will likely find LAT nodes that VMS knows nothing about.

15. Finally, start the system:
    @EMU_SRC:START_EMU
    You may clear the databases and start fresh each time you start the system by answering 'y' and confirming this to the prompts. For excessive safety, on the first startup, do this. You may clear the log files on each startup. Generally a good idea (after review). There is no mechanism to control the size of these files automatically and while generally well behaved can, on occasion become churlish.

Following is a snapshot for a system account that EMU works with. The most important quota is BYTLM. The listener enables promiscuous mode and saves as many frames as possible in the driver. This requires a LOT of BYTLM. There are undoutably more optimum settings for the others.

Username: SYSTEM Owner: SYSTEM MANAGER
Account: SYSTEM UIC: [1,4] ([SYSTEM])
CLI: DCL Tables: DCLTABLES
Default: SYS$SYSROOT:[SYSMGR]
LGICMD:
Flags: DefCLI
Primary days: Mon Tue Wed Thu Fri
Secondary days: Sat Sun
No access restrictions
Expiration: (none) Pwdminimum: 8 Login Fails: 1
Pwdlifetime: 30 00:00 Pwdchange: (pre-expired)
Last Login: 16-AUG-2000 13:46 (interactive), 26-OCT-2000 11:00 (non-interactive)
Maxjobs: 0 Fillm: 250 Bytlm: 400000
Maxacctjobs: 0 Shrfillm: 0 Pbytlm: 0
Maxdetach: 0 BIOlm: 150 JTquota: 4096
Prclm: 0 DIOlm: 150 WSdef: 2048
Prio: 4 ASTlm: 250 WSquo: 6400
Queprio: 0 TQElm: 20 WSextent: 32000
CPU: (none) Enqlm: 2000 Pgflquo: 150000
Authorized Privileges:
ACNT ALLSPOOL ALTPRI AUDIT BUGCHK BYPASS
CMEXEC CMKRNL DIAGNOSE DOWNGRADE EXQUOTA GROUP

GRPNAM GRPPRV IMPERSONATE IMPORT LOG_IO MOUNT
NETMBX OPER PFNMAP PHY_IO PRMCEB PRMGBL
PRMMBX PSWAPM READALL SECURITY SETPRV SHARE
SHMEM SYSGBL SYSLCK SYSNAM SYSPRV TMPMBX
UPGRADE VOLPRO WORLD
Default Privileges:
ACNT ALLSPOOL ALTPRI AUDIT BUGCHK BYPASS
CMEXEC CMKRNL DIAGNOSE DOWNGRADE EXQUOTA GROUP
GRPNAM GRPPRV IMPERSONATE IMPORT LOG_IO MOUNT
NETMBX OPER PFNMAP PHY_IO PRMCEB PRMGBL
PRMMBX PSWAPM READALL SECURITY SETPRV SHARE
SHMEM SYSGBL SYSLCK SYSNAM SYSPRV TMPMBX
UPGRADE VOLPRO WORLD

Note: EMU enables promiscuous mode on the local Ethernet device and attempts to read every packet on the cable. In most environments this causes very high I/O on the system and often results in significant impact on the node. It is strongly advised to run it on a standalone, dedicated VMS system. Any valid VMS system with a working copy of UCX is adequate.

Chapter 3
User Guide

This chapter describes in detail how to start and stop the system, monitor it's progress and the various utilities used to extract information.

3.1 Starting the system

To start the system issue the command:

$@EMU5_SRC:START_EMU

The procedure provides an option to restore all saved files or clear them and build a new database. The default is to reload all saved files.

EMU will then start all the processes required and after a few seconds it will exit back to DCL.

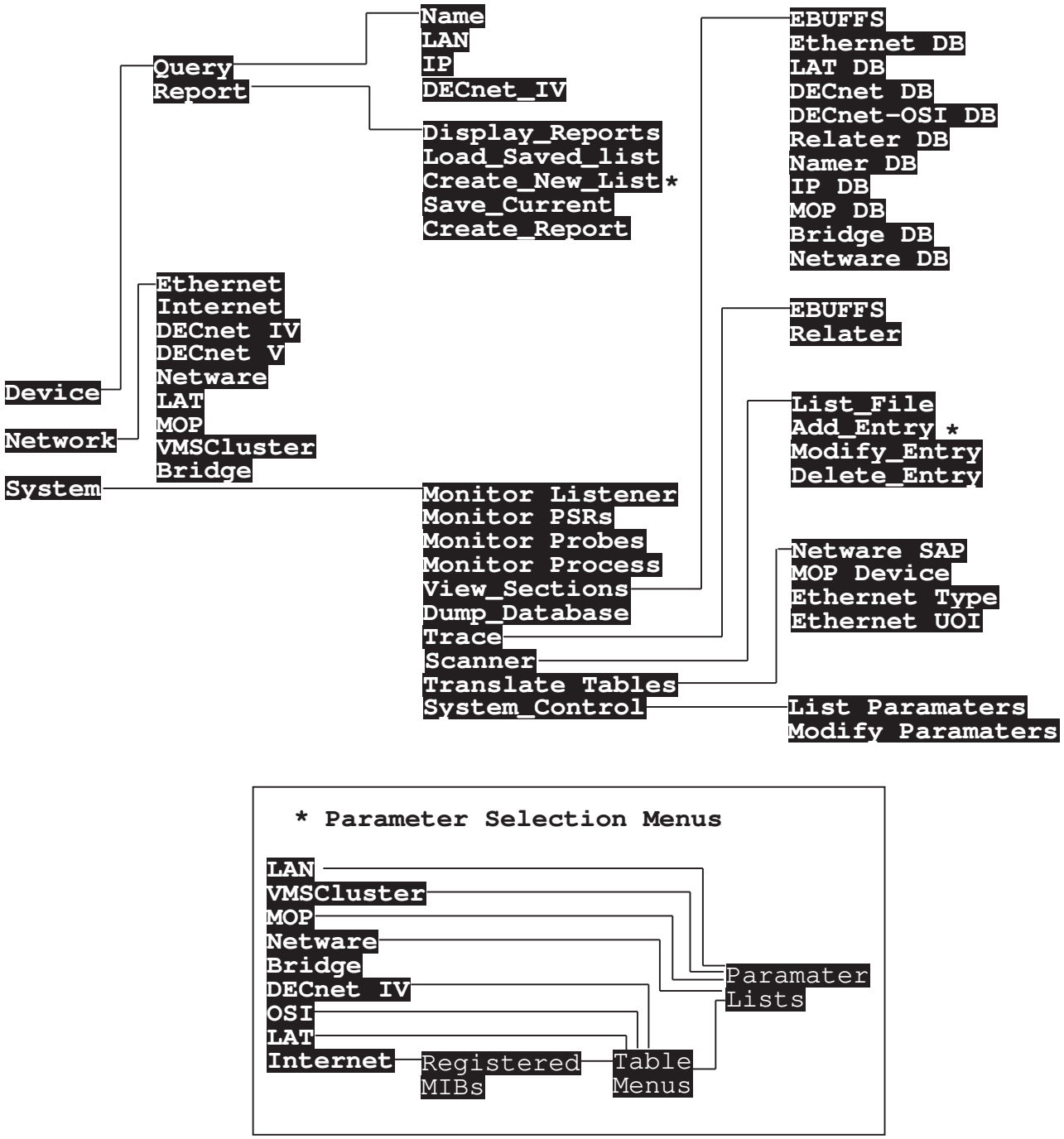
3.2 Stopping the system

To stop the system:

$ @EMU5_SRC:STOP_EMU

This deletes the EMU_CONTROL process which in turn triggers all other EMU process to shutdown. All database files and sections are saved to disk files in EMU5_DAT:

Figure 3–1: Menu System Schematic



3.3 User Interface

The user interface is entirely menu driven - there are no commands to learn. The menus provide comprehensive access to all EMU facilities and data. In addition to the menus there are a number of utilities (described in the Utilities section) that are run as separate procedures. Each menu item is described in the following sections.

3.3.1 Menu system overview

To start the user interface type EMU at the DCL prompt. The main menu then appears.

3.3.1.1 Menu Security

The menu items are controlled by the security system - that is if appropriate privileges are not owned by the user, some items do not appear. Note that the default privileges are used - the privileges the user has currently enabled are ignored, the system only checks the privileges owned. The required privileges are detailed in the menu description sections.

3.3.1.2 Getting help

The system supplies a comprehensive help facility and can be accessed by pressing the help key (on a Digital Keyboard or the equivalent on others) in any menu. Simply select the menu item help is required on and press the help key.

3.3.2 Main menu

The network can be seen as a collection of devices on 1 or more networks or a network and it's collection of devices. EMU supports both views.
There are 3 items in the main menu. In all cases, selecting an item from this menu displays a further menu.

1. Device. This items allows querying and reporting on devices. There are no privileges required to display this item.
2. Network. This items allows querying and reporting on networks. There are no privileges required to display this item. This is not yet implemented.
3. System. This item allows access to the system management/monitoring facilities. If the user does not have OPER privilege, this item is not displayed and access to the underlying facilities is barred.

3.3.3 Device menu

There are no special privileges need to access these menu items. From this menu, the user may select:

1. Query. Allows access to the live database. Using a further menu the user selects individual devices to display.
2. Report. Allows access to the reporting subsystem. Here the user may create, define and view device reports. See reporting menu for details.

3.3.4 Device Query menu

In this menu, the user selects devices to display. Regardless of how the device is found, the display is always the same: all data for the device become accessible. For each of the following items, selection issues a prompt asking for appropriate input. Standard VMS Wildcards are allowed. The appropriate database is searched for matches. Any found are displayed one at a time until no more match. Within the display, menus allowing control of direction and content are presented. There are no special privileges need to access these menu items.

1. Name. A nodename or any part of a nodename with Wildcards is input. A nodename may be up to 400 printable characters in length.
2. LAN. Either an address or protocol type can be input. In the case of an address the user must input in form:
   xx-xx-xx-xx-xx-xx
   where x is either a hexadecimal digit or wildcard.
   The protocol type query is not yet implemented.
3. IP. An IP address, or any part is input in form:
   ddd.ddd.ddd.ddd
   where ddd is a decimal digit in range 0-255
4. DECnet. A DECnet address or any part is input in form:
   aa.nnnn
   where aa is a decimal number in range 1-63 and nnnn is a decimal number in range 1-1023.

3.3.5 Device display format

Each device found (if any) are displayed in common format. The screen is split into 3 parts:

1. Top section displays the device. If a name is known the name is displayed, if not, the address is used.
   The class and device are also displayed. These fields are not yet implemented but in the released version, these will correspond to a user settable field showing relative importance and generic device type such as bridge, host, router etc.
   Once a particular protocol is selected, the time the device was last heard and the time the device was last updated on the selected protocol are displayed.
2. Control menu. 2 fixed items are followed by the list of supported protocols this device runs. The two fixed items are:
   Next. Display the next device matching the search string.
   Previous. Display the previous device matching the search string.
   Protocol list. For each EMU supported protocol this device runs, a menu item is displayed. Selecting any protocol item displays either the parameters for this protocol or a further menu of the tables in this protocol. The number of parameters in some protocols can be quite large so EMU defines each parameter as belonging to a table. Selecting any table displays the parameters associated with it.

3. Main Section. The parameters are formatted and displayed here in standard format: Parameter name : Parameter value

3.3.5.1 Parameter Display

The display of any parameter is controlled by:

- Security. Each parameter has associated with it the VMS privilege required to view it. If the user does not have the requires privilege, the parameter is not displayed.
- Level. Each parameter is defined as a 'level of interest' in range 0-5 (0 being most interesting and 5 being least). A hidden menu (described in next section) controls the level of reporting. Any parameter below the current 'level of interest' is not displayed.
- Parameter name and formatting are predefined but you are free to change this with the PARAMTBL utility. (which see)

3.3.5.2 Hidden Menu

While in the query display, if you press CTRL / a menu appears allowing:

- Logging. A prompt for file name appears at the bottom of the screen and all further data to screen is also written to this file. This is a 'flip-flop' selection; If logging is on and this is selected it is turned off, if it is off and then selected, it is turned on. Note that each time it is turned on, a new file is created.
- Report level. You can set the 'level of interest' described above here. Selecting the 2nd item displays a further menu used to set the level:
  1. Summary. The lowest level. Only those parameters defined as 'most interesting' are displayed. Generally this would include only names and addresses.
  2. Brief. Names, addresses and those considered major parameters are displayed.
  3. Normal. All but the most obscure parameters are displayed.
  4. Full. All parameter that can be formatted are displayed.
  5. Verbose. All parameters are displayed. If there is no formatting available for a parameter, a default translation is applied. Note that the level any parameter is defined in can be changed with the PARAMTBL utility.
- Update. Normally whenever a node is updated, it is then set to update again in 3 days. Once a display is started, selecting this forces this node to be updated on this protocol on the next cycle. In the case of LAN (MOP, LAT, SCS, IPX) protocols this will most likely happen within 10 - 15 seconds. In the case of WAN protocols (IP, DECnet, OSI) this may take as long as 30 - 45 minutes.

3.3.6 Reporting menu

The reporting menu allows the user to define, create and display reports. Reports are generated from report parameter files created by the user and formatted as a 'spreadsheet' where all matching parameters are in columns. For example: A parameter file specifying IP address and DECnet address finds 3 nodes and would format as follows:

Table 3–1: Report Output

IP Address	Decnet Address
1.2.3.4	50.1[1]
1.2.3.5	
1.2.3.6	
1.4.5.6	50.3[2]
1.4.5.7	50.2[3]
1.4.5.8	

1. This node 1 has 3 IP addresses and 1 Decnet address
2. This node 2 has 1 IP addresses and 1 Decnet address
3. This node 3 has 2 IP addresses and 1 Decnet address

The items in the reporting menu are as follows:

1. Display Reports. Displays a menu of previously created reports. Selecting any item causes the selected report to be displayed in the TPU editor. The report may be edited using all standard TPU features. A new version is always created. Upon exit from TPU, the user may elect any combination of keeping or discarding the new version and purging or not all old versions.
   Note: The call to TPU always creates a new file unless the user explicitly 'quits' out of the editor. For this reason the system assumes a new version is always created. If you quit out of an editing session be very careful how you answer the delete and purge questions!
2. Load Saved List. Displays a menu of all previously created report parameter files. These files define the contents of any report.
3. Create new list. Created report parameter files. Menus of EMU supported protocols, their tables and parameters are displayed. Selecting a parameter adds it to the list. Selecting a parameter already in the list removes it from the list.
4. Save Current. Saves the currently loaded report parameter list to a file. The user is asked for a report name. Enter ONLY a file name (no type or extension). These files can then be LOADed.

5. Create Report. A report is created using the currently LOADed parameter file. A parameter file must have been previously created or loaded.

3.3.7 Network menu

Using the items in this menu, an overview of the various networks can be obtained. This section is under construction (hard hats must be worn at all times and is closed to the public).

3.3.8 System menu

This section is used to monitor and debug the system. Each menu item in the system menu shows a specific aspect of the system. Most of these tools will not be useful to a normal user but may be helpful to the system manager should problems occur on your system. Many of the tools presented here were used as development tools and remain useful in quickly isolating problems either within EMU or demonstrating that problems are external.

3.3.8.1 Monitor Listener

The main Listener and PSR interface is presented as a continuously updated real-time monitor. The display appears as follows:

```
CCCI Ltd 1993          Ethernet Monitor Listener Display
 System:
  Event flags  = 80000000 01FE0000
  Queued  Lowest    Error      Cmplt  IOSB Err  Processed
  28       0        72694   15391978    0         50480
 Frame errors:          Mcast Src     Zero Src   Zero Dest   Zero PTY
                         0              0          0          9350
  Name      Status      Limit      Sent      Disc   In Proc      High
 RECORD   00000000         20         0         0         0         0
 ETHNET   00000001         16   4716223    219803         1        16
 IP       00000005         15   1224266     24677         5        15
 MOP      00000005         11     60083         0         0         8
 DECNET   00000005         12    726942      1462         1        12
 LAT      00000005         10    156502         0         0         6
 ARP      00000007         10   1224261         5         0        10
 LAVC     00000005         13    409631         0         1         7
 IPX      00000005         11    844851      4964         0        11
 BRIDGE   00000005          4    156261         0         0         3
 OSI      00000005          8    678413       187         0         8
```

The display is interpreted as follows:

- Event flags are flags used throughout the system to signal events. Normal operation has only the top flag permanently set and on occasion you will notice some flags changing rapidly. If any flag other than the highest on remains set (not 0) it usually indicates the process associated with this flag is no longer responding. Report this to software support.
- Queued is the number of buffers queued to Ethernet. As long as this remains above 0 all is well. If it reaches 0 and remains there there is a serious problem with the system. Report this to software support.
- Lowest is the lowest number of buffers queued at any 1 time. This will normally start at the number queued and progress towards 0 as the system uses resources.

- Error. The number of Ethernet read errors. This should always be a very low number (less than .001% of completions. Note there is a bugette here - the count regularly goes up for unknown reasons and there is no other known impact.
- Cmplt - Completions. The number of Ethernet reads. The Listener is reading every frame on the Ethernet and the display updates 1 per second so the rate this number rises at is the number of frames per second currently on the segment EMU is attached to.
- IOSB error. The number of times the Listener successfully received a frame but the frame was in error. This should always be 0. If this number rises there is either a fault on the local LAN (most likely) or serious EMU system fault (Less likely) or an operating system fault (Least likely).
- Processed. The listener discards most frames read as being not useful to EMU purposes. This includes any frame containing user data. The remaining frames are processed for network management information. This count is the number of frames processed and this number, expressed as a ratio of Completions gives a rough picture of the efficiency of the attached LAN.
- Frame errors. Each Ethernet frame is validated and any errors found are recorded here. They are: Multicast source. A frame was received with a multicast source address. This is not allowed by Ethernet rules and can trigger a broadcast storm. It cannot be determined who sent the frame. Zero Source. A frame was received with the top 3 bytes of the address set to 0. This is not a valid Ethernet address. Zero Destination. Same as in Zero Source. Zero PTY. The protocol type in the frame was 0. This is meaningless as this type is not defined. No station acting in accordance with Ethernet rules will receive this frame.
- The remainder of the display shows the processors that are active. Each processor has it's own set of counters and are interpreted as follows:

    - Name. Is simply the name this processor is known by.
    - Status. A bit pattern showing the current processing status. Only the bottom 3 bits are used:

Table 3–2: PSR Status Bits

Bit	Meaning When Set
0	Processor is able to receive
1	Type Checked [1]
2	All Traffic [2]

Notes:

1. Type checked. This processor receives frames only of the type specified in PSRTBL
2. All traffic. Most processors receive only multicast traffic. If this bit it is set it receives all traffic on that protocol type.

See warnings in PSRTBL before changing any of these bits.

3.3.8.2 Monitor PSRs

PSRs are the Protocol Specific Routines. There is 1 for each protocol EMU processes. The display appears as follows:

```
CCCI Ltd 1993          Ethernet Monitor PSR Display
  Name      Rec         Ret    Fmterr  Comm  Err   NoIPC  Alt  Rlt  Name
 OSI       681229      681229       0      0     0      0     0    0   0
 LAVC      411312      411312       0      0     0      0     0    0   0
 LAT       157144      157144       0      0     1      0     0    3   0
 MOP        60324       60324     106      0     0      0     0 4470   0
 IP       1229358     1229358       0      0     0      0     0   19   0
 IPX       849122      849122   44444      0     0      0     0    0   0
 ETHNET   4737078     4737078       0      0     0      0     0    7   0
 RECORD         0           0       0      0     0      0     0    0   0
 BRIDGE    156907      156907       0      0     0      0     0    0   0
```

The display has been compressed in this example to fit on the page. It is interpreted as follows:

- Name. The Name (Usually the protocol name) the routine is known by.
- Rec. The number of frames the PSR received from the Listener.
- Ret. The Number of frames processed and returned to the Listener. This should always be equal to Rec.
- Fmterr. Format error. A fame with unexpected format. Usually this is cause by a device sending a faulty frame. If the number rises dramatically, it can indicate the processor is faulty. Report this to software support. Not in the above display the high number being reported by IPX. This is being investigated.
- Comm. The number of command buffers received. Internal processes can send commands to PSRs to affect their processing.
- Err. The number of error messages this processor has written to the error log. If this number rises quickly, a review the error log should indicate the problem. If the problem cannot be understood or rectified, please contact software support.
- NoIPC. No Inter Process Communication Buffer. EMU processes communicate using a common buffer pool. If a process requests a buffer and none are available this event is counted. A small (Less than 5 per hr) number is expected. These counts are used in the auto tuning process and often a restart of EMU will retune the system and alleviate this problem.
- Alt. The number of Alerts this processor has raised.
- Rlt. The number of relater frames this processor has sent. Relater frames carry information that one processor determines is 'of interest' to another. This information is relayed in Relater frames.
- Name. No longer used.

3.3.8.3 Monitor Probes

Probes are the processes that interrogate cooperating nodes on the network for configuration and performance data. Currently EMU supports SNMP for IP addresses, CMIP for OSI addresses, NICE for DECnet, MOP, LAT and Ethernet protocols. This display charts the amount of data EMU is adding to the network and is displayed as follows:

```
    Probe                  Sent      Receive        Error     NoRespose

 SNMP                     10424         9215          472          1250
 CMIP                      1364         2301          742            82
 NICE                        95          539            0            20
 MOP                        139          139            0             0
 LAT                       1713         1713            0             0
```

Note the LAT processor reads only from the local node - it does not send any data across the network. The same is true of the Ethernet probe.

The display is interpreted as follows:

- Name. The name the probe is known by.
- Sent. The number of queries sent by the probe
- Receive. The number of good responses received
- Error. The number of responses received indicating an error in either the request or the response. Each response of this type is logged in the error log.
- NoResponse. The number of frames sent that received no response.

3.3.8.4 View Sections

EMU builds a number of internal databases in memory sections. Selecting this item displays a further menu of sections that can be viewed directly. For most sections there is a standard format displayed. The display is split into 3 parts:

- Top level. Gives information about the section being viewed
  Entries: The number of physical entries in the section
  Recsize: The size of each entry
  Max Entries: The maximum number of entries the section will currently contain. If this number of entries is reached, the controlling process gains exclusive access to the section, writes it out to a file, deletes the section, creates a larger section and restores the data. It then releases access to the system.
- PID: This is the internal identifier EMU uses to target a particular process, and therefore it's section. This PID relates to the Event Flags in the Listener display.
- Middle Level: Not currently used.
- Main Level. Each PSR section displays the standard header and then the PSR specific data. The common part is documented here.
- Common Header:
  - Flags: This is the PID of this process

- Boxid:This is a number generated internally and used to relate the various protocols found on the network to the various devices detected. Throughout this system a BOX is synonymous with a single physical device. Thus a router, PC or VAX are all BOXes with a single BOXID associated
- Ptypbits. A bit pattern indicating the protocols detected on this device. A bit set in this field corresponds to the PID of a protocol this device is running. That is to say, if bit 12 is sent in this field, EMU has determined that protocol 12 (Ethernet) is running on this device.
- Control. A bit pattern indicating the status of this record. Currently 3 bits are used to indicate:
  - Record is deleted. This area may be overwritten at any time.
  - Update. This record has been changed since list time the corresponding probe updated the record. It will be updated in the next cycle.
  - NoPoll. Do not poll this device on this protocol. May be set by the user or the system. The system will set this bit if the device responds with any indication that it does not allow logins.
- Last. The last time EMU detected a frame from this address on this protocol. If the last time recorded exceeds the Ageout for this protocol, the record will be deleted.
- FST. The first time EMU detected a frame from this address on this protocol. Helpful when trying to determine events around a particular incident. This event is always alerted.
- ALT. Time last alert was sent for this device on this protocol.
- Status. Not currently implemented
- Accesses. The number of times this record was looked up. The search is always done sequentially and during section build, the records are sorted such that records accessed most often are nearer the top of the section. This speeds up searching dramatically. Note: Not really true. This was (and remains) the intention but in fact, it is not implemented.
- Len. The length of the main lookup key in this section. The location of the key is constant in all sections and combined with this field, allows a single common search routine to operate.
- HowSet. How this device was found and the process that caused it to be created. Changed as a result of other problems. This field is now always set to the 'parent' protocol regardless of who actually set it. Makes it a bit useless.
- Current Readers. The number of process currently reading this record. Combined with Current Writers, these fields are used to organise access to the record so that any retrieval is valid. That is a reader will not retrieve a half written record and writers do not overwrite each other's data.
- Updt. Time this device was last updated on this protocol.
- Lpol. Last time this device was polled on this protocol.

- Polls. The number of times this device was polled on this protocol. This field along with Responses allows EMU to determine if it should continue polling a device that is unlikely to respond.
- Interval. The number of seconds (Default = 3 days) between updates
- Support. A bit pattern showing the various management functions this device supports on this protocol. The pattern is specific to the protocol.

3.3.8.5 Dump_Database

The database can be formatted and dumped. Selecting this item displays a prompt for file name. If a file name is entered, the dump is sent to both screen and file. If the prompt is not answered, the dump is to screen only. Be careful here - the database is usually quite long - 15,000+ lines will normally be printed.

3.3.8.6 Trace

This is a debugging tool used to see in real time the frames passed between EMU processes. Currently 2 processes support the trace facility:

- EBUFFS. These are the buffers passed from the listener to the PSRs. The trace shows these buffers as they are send from the listener to the PSRs and the PSRs returning them to the listener.
- Relater. These buffers are passed from the PSRs to the relater and are the controlling inputs to that process. Should any node display incorrect relationships (like an address that it does not have or the reverse), this trace can be used to find out why.

3.3.8.7 Scanner

The scanner process scans the database for parameters collected from the network via polling that should appear in the PSR level databases. A small database detailing which parameters to search for and where to send them is controlled by this interface. If you value your life, do not change these fields. Incorrect entries in this database cause immediate and irreparable brain damage to the system.

3.3.8.8 Translate Tables

The network is parameter driven and the value of those parameters is what EMU collects, analyses, stores and displays. Most parameters are not in any form suitable for human consumption and therefore must be translated. Most of the parameters in this area are not documented in any central location nor is there any definitive list. The tables provided are the best known to the developers but cannot be guaranteed to be either complete or correct. In this subsystem you can edit the source tables and compile them into the system for use by the parameter translation mechanism. There are currently 4 tables provided:

- Netware SAP. This is a code showing the service(es) any Netware node broadcasts and the correct translation of this is very useful as it describes in a few words the major use of this device.

- MOP Device. Any node supporting Digital's MOP protocol broadcasts a frame describing itself on a regular basis. One of the parameters is it's device type - again very useful in describing in a word or two what this node is.
- Ethernet type. Any protocol on the Ethernet must send it's frames in Ethernet format. In the Ethernet format is a field describing the protocol the application is using. One of the simple tricks EMU pulls is simply reading that field for each frame and therefore is able to list all the protocols any device runs. This table translates the codes into meaningful words. †
- Ethernet UOI (Universal Organisation Identifier). In simpler times, this field was known as the manufacturer's code. Even more simply: The top 3 bytes of any hardware address is a code showing the manufacturer of the Ethernet device in the box. This table controls that translation.

3.4 Alert

For a number of reasons (mostly performance and laziness) the user interface to the ALERT subsystem is not implemented under the main EMU menu system but is carried separately. This interface allows you to monitor real time alerts and browse through previous alerts conveniently. To start the alert interface:
$ ALERT
This brings up a screen with a 3 item menu.

3.4.1 Main Menu

The main menu in the alert system allows you to:

1. Monitor. Selecting this item clears the screen and waits for alerts from the system. As each alert is received, it is formatted and displayed here. To return to the main menu, press Ctrl Z.
2. Review. Selecting this item allows you to browse through the alerts previously generated and logged. All alerts are logged to the file EMU5_DAT:ALERT.DAT. It is advised to check the size of this file on occasion as there is no mechanism to control it's size and on an active (some say flaky) network many alerts can be generated over a short time. Selecting this item brings up another menu:
   - Since. Allows you to set the time from which previously logged alerts are re-displayed. The format is in VMS standard absolute time (DD-MMM-YYYY HH:MM:SS.CC) and any part missing from the input will default to current time (though you MUST include the separating characters. For example if you input '– 13' you will get all alerts since 13:00 (1 PM) from today. No input will display all alerts from the time the alert logging file was created - usually system start time.

† Ethernet types are actually in 2 formats: Type II (a 2 byte type field) or IEEE (a 2 byte SSAP/DSAP).If DSAP/SSAP is 'AAAA' then it is extended Ethernet(a 3rd type!) and a 5 byte field is used. The 5 byte field is in fact the 2 byte type coded with the UOI prepended. If you find this confusing do not panic - it is not a test of networking skills so much as a test of patience.

- NodeName. Displays alerts only for the node specified. Wildcards will be allowed once this section is finished - that is it is not yet implemented.
- Class. Alerts are divided into classes: FAULT, CONFIGURATION, ACCOUNTING, PERFORMANCE, SECURITY. This is the OSI standard (FCAPS) arrangement and seemed such a good idea, we stole it. In addition, EMU defines a SYSTEM category to allow the system itself to report problems it believes need operator assistance. In the current version, only CONFIGURATION alerts are generated. Note this function is not yet implemented.
- Execute. Once the selection criteria has been set up (Currently once time has been established) selecting this item causes the selected alerts to be displayed 1 at a time in the screen. The display screen allows 'next' and 'previous' alerts to be displayed.

3.4.2 Alert Format

All alerts are displayed in a standard format:

- Line 1 Shows the class of the alert and the device the alert is for. If the system knows the name of the device it is used, otherwise the address of the device is displayed.
- Line 2 shows the date and time the alert was received and the subsystem that sent it. This will usually be a PSR(Protocol Specific Routine)
- Line 3 will usually be the subclass. In the case of configuration alerts there are 3 subclasses: ADD, DELETE or MODIFY.
- If there is additional information this follows and will normally be a list of parameters that were added, deleted or modified.

3.5 Utilities

Utilities are those routines that are little used, not to be used by the uninitiated or otherwise inappropriate to be included in the main user interface. It is firmly advised that these routines be well understood before use. The effect of misuse can be anything from nothing to death by brain removal. In the later case, the effect may be quite subtle with death taking a long time and inflicting pain and frustration on all affected.

3.5.1 Error Log

The error log file EMU5_LOG:EMU_LOG.ERR is a typeable file (that is you can type/print/edit it) that the system writes errors and other significant information to. An extract of the file and interpretation follow:

```
%RELATR-W-VERADDR, 22-JUN-1998 11:50:00.08 RELATER Invalid Target
addr 002A54F4 For PID 0000000C From 00000006
  ------------------------------------------------------------
%CFGMON-E-SNDALT, 22-JUN-1998 11:51:40.21 NETMON SEND_ALERT Error
00000601
  ------------------------------------------------------------
%NODSCN-I-ADDIP, 23-JUN-1998 12:23:05.07 CFGMON Added node
171.144.141.128
  ------------------------------------------------------------
```

Notes:

1. The first section is the facility(RELATR), message level (I = Information, W = warning, E = Error) and short form of the error message (VERADDR).
2. The second part is the time and date the message was logged.
3. The remainder is variable and is intended to provide a complete picture of the problem (or not). The messages above mean:
    1. The relater received a frame containing an invalid address. The MOP module (00000006) sent address 002a54f4 specifying it as an Ethernet address. This is not a valid Ethernet address.
    2. The Network Monitor process sent an alert message that was too large for the alert buffer. In EMU error messages, any time a message contains the keyword 'Error', it is always followed by a VMS standard error code. To translate this code type at DCL:
    $ WRITE SYS$OUTPUT F$MESSAGE("%X601")
    In this case, code 601 is buffer overflow.
    3. The NodScn process found another IP address and added it to the database.

3.5.2 PARAMTBL

Each parameter in the EMU database has an associated record describing how to format it, the privileges needed to view it and how the alert mechanism reacts if it is added, deleted or modified.
Each parameter is associated with a protocol and a table. In some cases where there are multiple tables within a protocol, the table names are also in this file and can be modified similarly to parameter names.

3.5.3 Operations

The start the program:
$ PARAMTBL
A menu appears with 2 items:

- Parameters. Selecting this item allows you to view, change or delete the characteristics of parameters.

- Tables. Selecting this item allows you to view, change or delete the characteristics of tables. In the case of tables there is only 1 characteristic - it's name. Note that changing the name will prevent the help entry on this table name from working unless the corresponding entry in the help library is also changed.

3.5.3.1 Parameters

You may list, add modify or delete parameters. In all cases selecting an item takes you through a menu structure of the protocols and tables to find a particular parameter.

- List. This will list all that parameters in a particular table. There are 5 columns displayed:
  1. PRM. The parameter number.
  2. Keyword. This is the name that will be used when displaying this parameter.
  3. Ctl/Rtn. This is the formatting indicator used to format the parameter value for display. It is either an FAO directive or a routine number.
  4. Flg. This is a bit pattern that controls whether the parameter should be polled for and included in the database and if so, what the alert mechanism should do when it is created, modified or deleted.
  5. LvL. The level this parameter is displayed at. In the user interface the user may select displays from Brief to Verbose displays. This number corresponds to the level.
- Add. Adds a parameter to the file. **This function should only be used by developers. It is included here only for completeness.** To add a parameter you need to know:
  - The Table the parameter is in. Tables are either assigned internally by the system or calculated from data received from the network. Tables are found via menus that appear on the left of the screen.
  - The parameter number. Usually this is assigned by the protocol but in cases where the protocol does not assign this, the system does. All parameters have an assigned number and belong to exactly 1 table.
  - Formatting. How this parameter is formatted. It is either by FAO directive or routine. If the parameter cannot be formatted by a simple FAO string (usual case) a routine is supplied. A menu of available routines is presented on the left of the screen.
- Modify. You may change any aspect of a parameter entry. **Be careful and sure of what you are doing here - changes here can have very subtle and undesired effects.** If Modify is selected a menu of protocols appears. Using this menu will take you down through the table hierarchy to arrive at a particular table. Once a table is arrived at you will be asked for a parameter number. This parameter must already exist. To find parameter numbers use the list function. A brief explanation of each field you can change follows:
  - Security. A menu listing all VMS privileges is displayed. Selecting any item causes the selected parameter to require the user to have that privilege in order to view it's presence. That is to say, if you select CMKRNL, the next time you view the

parameter list and you do not have the CMKRNL privilege, the parameter will not be seen. Selecting an enabled privilege, disables it.

- Keyword. This is the word or phrase this parameter is known by. Any printable string up to 64 characters is allowed. If you change the name of a parameter, the help entry will cease to work until you also change the help entry in the help library. Instructions on how to do this are provided in the Hints section.
- Formatting. Do not change this. If you do, software support will yell at you when you tell them the system is broken. If you change this, you will break the system.
- Report Level. All parameters belong to a level in range 1-5. The level is used in displaying lists. In general the most interesting parameters are 1 and the least interesting parameters are 5. The display is controlled by the hidden menu on the query display and correspond as:

  1. Summary. The lowest level. Only those parameters defined as 'most interesting' are displayed. Generally this would include only names and addresses.
  2. Brief. Names, addresses and those considered major parameters are displayed.
  3. Normal. All but the most obscure parameters are displayed.
  4. Full. All parameter that can be formatted are displayed.
  5. Verbose. All parameters are displayed. If there is no formatting available for a parameter, a default translation is applied.

- Include in database. Whether or not to include this parameter, when received in the database. If not included, it will not be monitored or processed in any manner. Normally, most parameters are included except for those which are very dynamic and are not useful at this level. Examples of these are:

  - TCP connections in MIB-II. The list of addresses and ports currently connected to this host.
  - Bridge FDB tables. The list of MAC addresses and destinations in a bridge.

  Both of these examples, are (in most cases) very long lists that change rapidly. The result of putting them in the database is to use a LOT of space up with data that is obsolete in minutes and depending on how other parameters are set, will cause excessive alerts. **Be sure you understand the impact and use of any parameter before changing this field.**Default is yes.
- Include in monitors. Essentially the opposite of include in database. Monitors are intended to retrieve and display the rapidly changing parameters. Monitors are not yet implemented. Default is no.
- Alert on Create. Exactly that. When this parameter is created, send an alert to that effect. Default is no.
- Alert on Modify. Exactly that. When this parameter is modified, send an alert to that effect. Default is no.

- Alert on Delete. Exactly that. When this parameter is deleted, send an alert to that effect. Default is no.

3.5.3.2 PARAMTBL Summary

This utility is very useful for customising your system at low levels. With a few keystrokes you can change the way data is displayed across the system and using the security facility, limit access to a very fine degree. With power comes responsibility; Be sure of what you are doing before you do it. As a safety, backup the file EMU5_DAT:MAPPER.DAT before you make changes. If it all goes horribly wrong, restore your backup.

3.5.4 MIB compiling and Registration

MIBs (Management Information Base) are readable files produced by the manufacturer of a device that supports SNMP (Simple Network Management Protocol). At this writing SNMP is supported only in the IP world. The MIB defines the parameters that the device supports, how to access them, the format of the parameters and (often) a short explanation of what it means.
MIBS must be compiled before they are usable by the system. EMU provides a compiler in 2 parts for a number of reasons:

- It is never advisable to edit a MIB although it is not unknown for errors to be present in the MIB itself. Using the EMU MIB compiler, you can edit the output of the 1st pass to correct any errors of this kind.
- Simple is a relative term. At the protocol level SNMP is not simple to implement although it is much simpler than many. At parameter (MIB) level it is often too simple. A prime example of this is how a MAC (Ethernet) address is specified; it is always shown as an Octet String - that is a string of bytes. Octet strings are generally printable characters but in the case of a MAC address this formatting does not work. EMU therefore redefines these parameters using private definitions and therefore maintaining correct output. Again, output from the 1st part of the compile can be edited to correct these problems.

The private definitions EMU defines are as follows. Note that they are defined using the ASN.1 tags reserved for private data types and therefore will not interfere with any existing or future definitions contained in a valid implementation of SNMP.

Table 3–3: EMU ASN.1 Private DataTypes

EMU Symbol	Tag Value	DataType
SNMP_ASN1E_MACADDR	193	Mac address
SNMP_ASN1E_DISPSTR	194	Printable string
SNMP_ASN1E_TBLTOP	195	Table top [1]
SNMP_ASN1E_TBLENT	196	Table entry[1]
SNMP_ASN1E_BRIDID	197	Bridge ID

Note 1: These 2 types are used internally only. They should never be edited into a MIB at any stage.

There are a number of enhancements included in this compiler that are not present in industry standard ones:

- The 1st pass extracts all the description fields and builds a VMS standard help library. After registration this help library is available to the user interface providing the user with access to all parameter definitions in readable format. Because MIB parameters are many and often have cryptic or obscure names, this is seen as an essential service.
- The private definitions allow more precise and clearer display of values than those that simply following the protocol.

3.5.4.1 Usage Instructions

To compile a MIB and make it available for use follow these instructions:

1. Ensure the MIB you want to compile is in EMU5_MIB: directory. MIBs can normally be obtained by contacting the manufacturer of the device via either the Web or FTP. EMU supplies many of the more common MIBs precompiled and registered. To find which MIBs are already available run PARAMTBL, Select Internet and the resulting menu will display all registered MIBs. Other MIBs are simply supplied and are located in EMU5_ MIB:.
2. Run EMU5_EXE:MIB_COMPILE1.The program will ask for the MIB name. Enter it. This compile phase:
   - Opens the input file and creates the Help and output files with the same name as the MIB and with extensions .HLP and MC1 respectively.
   - IMPORTS all files specified in the IMPORT statements in the MIB. These must exist and have been previously complied by this routine. If any do not exist you are given the choice to continue or exit. It is often a good idea to continue as much of the IMPORTED information is not used. If at a later stage the program is unable to define a parameter, exit the program, compile the offending MIBs and start again.
   - Extracts all definition statements and writes then out the the HLP file.
   - Builds the schema internally. See the hints section for a brief description of the schema.
   - Once the MIB is entirely read in, and if there were no errors detected, the schema is written out to the .MC1 file, The Help library is created and the help file just created is inserted into the help library.
3. Run EMU5_EXE:MIB_COMPILE2.The program will ask for the MIB name. Enter it. This compile phase:
   - Takes the MC1 file previously created and converts it to a form usable by the system. It creates a file with the same name as the MIB and extension .MC2. This is the file that the rest of the system searches for when listing out available MIBs.

- The program displays it's progress by showing the number of lines read/ written on the screen. If the program exits without error, creation has been successful.

4. Run EMU5_EXE:MIB_REGISTER.The program will ask for the MIB name. Enter it. The program will determine if the MIB is already registered and if so, you may replace it with the new definitions. Otherwise it advises the number assigned to this MIB and includes all definitions in EMU5_DAT:MAPPER.DAT. The MIB is now available for the system to use on any device that supports it.
5. After making any required adjustments to the system' usage of the parameters just registered (see below for a hint), you then tell the system which nodes to use this MIB on by:
   - In the main EMU user interface select a node supporting this MIB.
   - Select IP protocol. If the device does not support IP, it does not support any MIBs.
   - The menu of available MIBs is displayed along with Set MIB Params. Select 'Set MIB Params'
   - A further menu appears. Select 'Set MIB'
   - A menu of available MIBs appears. Each item is a toggle; Selecting an already selected MIB deselects it and selecting an unset MIB sets it. The Selected MIBs are shown in a bit pattern at the top of the screen - each MIB has a corresponding bit in the pattern that when set (to 1) indicates to the system to use this MIB when updating this address.

You can test this MIB with another utility that is also useful in browsing the network: MIB-WALKER.

Note that by default all IP addresses are assumed to support MIB-II - the basic MIB that most SNMP devices support - at least in part. Internally to the system this is MIB number 1. DO NOT CHANGE THIS DEFINITION. If this is the only MIB a device supports or the only MIB you want to act upon a device then no action is necessary.

3.5.5 MIBWALKER

A MIBWALKER is a generic program that allows you to retrieve and display the results of data from a node supporting SNMP. A number of conditions must be met before this will work:

- The device you are targeting must be running, attached to the network and be prepared to respond to SNMP requests.
- You must use the correct MIB to talk to it.
- If the Device does not allow public access you must know the community string it supports.

To run the MIB walker type 'MIBWALKER' at the DCL prompt. This system wide symbol is set up at EMU start time. Note that EMU is not required to be running in order to execute this routine.

Running the program displays a menu on the screen:

- Set IP address. Select this item and enter the IP address you wish to talk to. Wildcards or names are not allowed. In future, this routine may be integrated with EMU to allow more friendly searches.
- Set MIB. Set the MIB you want to use on this device. A menu of available MIBs is presented. Select one.
- Set Community String. By default the transactions will use the public string. If the device does not accept this, select this item and enter the community string the device uses. Selecting this item and providing no input causes the public string to be used. Note the string is case sensitive.
- Walk the MIB. This MIB walker is a bit unusual in that it does not actually 'walk' the MIB. Selecting this item (after setting at least the IP address and MIB) presents a menu of tables and items this MIB supports. Selecting any item sends the appropriate request to the device and if an answer is received, displays the results. Pressing the HELP key anywhere in this menu invokes the help system and displays and explanations of the associated parameters available. Exit this menu with CTRL Z.
- Toggle Log file. Selecting this item turns logging on or off. If logging is off it turns it on, if off, it turns it on. When on all displays to the screen are also written to the file MIBWALKER2.LOG in the default directory.

All the functions are independent. That is you can set the MIB and switch between addresses or vice versa. It is often a good idea to run this program after compiling and registering a MIB and noting which parameters should be adjusted using PARAMTBL. For many MIBS you may not want all the parameters (by default all will be collected) or you may want to alert when additions, deletions or modifications are detected on some parameters. By default no alerts are generated.

3.6 Errors and Omissions

This is the beta release of software. Some functions described in the user guide are not implemented and others may not perform exactly as expected. It is the purpose of this release to find these errors, note the usefulness of the system as it stands and to collect feedback for it's further development. This section describes the known errors and some possible workarounds. These are given in no particular order:

- Monitor Listen Display. The error count increments regularly. No other symptoms found (that is there are no known errors). Probable coding issue. No effect on system.
- On rare occasions the system does not shutdown fully. Before starting the system it is a good idea to start the user interface by typing EMU and the DCL prompt. If the error 'fatal controller error' is returned, the system is fully shutdown. If not, wait a few minutes and try again. If still not shutdown, locate the EMU processes (all EMU processes have 'EMU_' as the 1st 4 characters of their names) and 'STOP/ID' them. Note if this action is necessary it is imperative that on next startup the databases are cleared.
- DECnet IV PSR does not display on the MONITOR PSR display. Reason unknown. No other known effect.
- User interface often hangs upon exit from trace facility. Control Y the process and restart the interface. Will be fixed in full release.

- Bridge Ids in the FDDI display are not translated correctly. The selector wrongly assumes a specific table this is in. Will be repaired in full release.
- System is prone to slowdown after running for about 10 days. Cause is fragmentation of the database file which is in turn is caused by excessive read/writes to it. A fix has been engineered and can be expected in the full release. In the meantime, the file can be reorganised using CONVERT on occasion. Note that the system must be shutdown to do this.
- Not all parameters are translated. The two main reasons for this are:
  1. The parameter is not documented and requires reverse engineering to determine what it is and how to translate it.
  2. The parameter is complex and requires special routines to format it. This is an ongoing exercise
- The hidden menu function that filters parameter display based on 'importance' is not effective largely because the parameters are all set to the default of 0 (with the exception of MIB generated parameters which are set to 3). The mechanism also needs some adjustment to filter out untranslatable parameters. The task of revaluing the individual parameters is left to the user at this point although the system may provide more convenient templates in future.
- MOP sends too many relater frames. The MOP module acquires both the MAC in use and the burned in MAC address on the card. There is no way for MOP to determine if the hardware address has been passed to other databases therefore always sends this information on each cycle. There is no effect other than uselessly using a small resource. A fix is in progress.

A file EMU5_DAT:EMUBUG.DAT contains all known errors and fixes at release time.
Please send any additions to this list to:
system@ccci4.demon.co.uk

3.7 Hints

This section is included to impart some uses of the system that may not be immediately obvious.

3.7.1 Further processing of Reports

Reports are always generated as flat files and can be easily formatted and manipulated using simple DCL procedures. Some useful things to note:

- All data is presented as list with each parameter separated by a vertical slash (|). The number of slashes present is always the number of parameters less 1 - if a parameter does not exist there is nothing between the slashes where it would be if it did. This allows for very easy detection and extraction using the F$ELEMENT lexical function in VMS and easy importing to Excel.

- The first parameter is always a number. The number itself has little meaning but note that all parameters associated with a single box always have the same number. Put more simply: When the number changes, it is a different device. A number of example report formatting procedures are in EMU5_RPT:.

EMU does not currently provide an interface capable of extracting data conditionally. That is there is no way to say "Get all nodes with names beginning with "TST" and having at least 1 IP address that does not begin with "171". This is however easily accomplished by generating a report with all occurrences of the parameters and writing a simple procedure (about 10 lines) to apply the logic.

3.7.2 Seeding the IP database

There is no direct way to add an address to EMU but in the IP section this can be done indirectly by 'pinging' the address. If the node answers, it will be added to the database as the IP module listens for and receives all ICMP frames (A 'ping' uses an ICMP echo request and response sequence). It may be useful to keep a procedure containing ping commands to important IP addresses and run it at system startup as this will speed up finding further IP addresses.

3.7.3 Finding a MOP console user

If, when using TSM to access a server and the connection fails, it is often because another system is using the console port. The message returned does not help to make this obvious. Within EMU, find the server you are trying to connect to and force an update using the hidden menu. Wait about 30 seconds and redisplay. Assuming the update has been successful, (note the last update time at screen top), the MAC address of the system connected to it will be displayed as 'Console User'. Find this MAC address to identify the system that 'owns' the console.

Index

A

Adding a node to IP database, 3–23
Alert - Format, 3–14
Alert - Main Menu, 3–13
Alert Subsystem, 3–13

B

BOXID, 3–10
BUILD.COM, 2–1

D

Device display format, 3–4
Device menu, 3–3
Device Query menu, 3–4
Dump_Database, 3–12

E

EMU_LOGICALS.COM, 2–1
Error Log, 3–14
Ethernet type, 3–13

F

Forcing Update, 3–5

G

Getting help, 3–3

H

Hidden Menu, 3–5
History, 1–2

I

installation, 2–1

M

Main Features, 1–1
Main menu, 3–3
Menu Security, 3–3
MIB, 3–18
MIB Compilers, 3–19
MIB compiling and Registration, 3–18
MIB Register, 3–20
MIBWALKER, 3–20
Monitor Listener, 3–7
Monitor Probes, 3–10
Monitor PSRs, 3–9
MOP console, 3–23
MOP Device, 3–12

N

Netware SAP, 3–12
Network menu, 3–7

O

Obtaining MIBs, 3–19

P

Parameter Display, 3–5
Parameters not translated, 3–22
PARAMTBL, 3–5, 3–15
Private DataTypes, 3–18
PSR, 3–14

R

reporting menu, 3–6
Report Processing, 3–22

S

Scanner, 3–12
Seeding the IP database, 3–23
SNMP, 3–18
START_EMU.COM, 2–2
Supplied MIBs, 3–19
System Account quotas, 2–2
System menu, 3–7
System Overview, 1–2
System Start, 3–1
System Stop, 3–1

T

TPU, 3–6
Trace, 3–12
Translate Tables, 3–12

U

UOI, 3–13
Update Cycle, 3–5
User Interface, 3–3
Utilities, 3–14

V

View Sections, 3–10
VMS Error Translation, 3–14