

NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL

_S

Ps

NP

NP

\$G

\$O

NP

PA

_L

NN		NN	MM	MM	LL	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	
NN		NN	MM	MM	LL	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	
NN		NN	MMM	MMM	LL	DD	DD	FF	
NN		NN	MMM	MMM	LL	DD	DD	FF	
NNNN		NN	MM	MM	LL	DD	DD	FF	
NNNN		NN	MM	MM	LL	DD	DD	FF	
NN	NN	NN	MM	MM	LL	DD	DD	EEEEEEEE	FFFFFFFF
NN	NN	NN	MM	MM	LL	DD	DD	EEEEEEEE	FFFFFFFF
NN	NNNN	NN	MM	MM	LL	DD	DD	EE	FF
NN	NNNN	NN	MM	MM	LL	DD	DD	EE	FF
NN	NN	NN	MM	MM	LL	DD	DD	EE	FF
NN	NN	NN	MM	MM	LL	DD	DD	EE	FF
NN	NN	NN	MM	MM	LLLLLLLLLL	DDDDDDDD	EEEEEEEEEE	FF	
NN	NN	NN	MM	MM	LLLLLL.LLLL	DDDDDDDD	EEEEEEEEEE	FF	

MM		MM	DDDDDDDD	LL
MM		MM	DDDDDDDD	LL
MMM	MMM	DD	DD	LL
MMM	MMM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DD	DD	LL
MM	MM	DDDDDDDD	LLLLLLLLLL	
MM	MM	DDDDDDDD	LLLLLLLLLL	

: NMLDEF.MDL - internal definitions for NML

Version 'V04-000'

```
*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****
```

++

: FACILITY: VAX/VMS DECnet Network Management Listener

: ABSTRACT:

: This file contains various MDL definitions for NML.

: ENVIRONMENT:

: n/a

: AUTHOR: Scott Davidson CREATION DATE: 1-Aug-1979

: MODIFICATION HISTORY:

```
V03-011 MKP0016      Kathy Perko    25-Mar-1984
      Add node parameter parsing constants.

V03-010 MKP0015      Kathy Perko    6-Jan-1984
      Add X25 Access Module entity.

V03-009 MKP0014      Kathy Perko    7-July-1983
      Add definitions for ISAM node permanent database with
      four ISAM keys.

V03-008 MKP0013      Kathy Perko    8-May-1983
      Coordinate NML$LOG bits in NML and MOM.
```


V03-007 MKP0012 Kathy Perko 24-April-1983
Delete service functions from NML. Also, add
support for circuit service substate and service
physical address.

V03-006 MKP0011 Kathy Perko 17-Jan-1983
Support for NI Configurator Module.

V03-005 MKP0010 Kathy Perko 18-Nov-1982
Add module as a source for logging events.
Add Phase IV.

V03-004 MKP0009 Kathy Perko 27-Sept-1982
Add Area and Adjacency entities.

V03-003 MKP0008 Kathy Perko 21-Sept-1982
Add dispatch table definitions for ZERO.

V03-002 MKP0007 Kathy Perko 22-June-1982
Set up "active network" for X-25 Protocol networks and
enhance Entity Table (in NMLDAT) to include ACTIVE and
KNOWN search key IDs, lengths, values, and operators.
Add X29-Server entity.

V03-001 MKP0006 Kathy Perko 17-Mar-1982
Add NML entity codes and permanent database keys for
X-25 Access and Server Modules.

V02-006 MKP0005 Kathy Perko 20-Nov-1981
Add parameter grouping flags for X.25 Protocol DTEs,
Groups, and Networks.

V02-005 MKP0004 Kathy Perko 17-Nov-1981
Add circuits to event source block definitions.

V02-004 MKP0003 Kathy Perko 3-Nov-1981
Add field to MSB for second line of message text
for RMS signalled messages.

V02-003 MKP0002 Kathy Perko 14-Oct-1981
Add flags for parsing V2 to V3 SET LINE conversions.

V02-002 MKP0001 Kathy Perko 18-Sept-1981
Change network management version number to 3.0.0


```
:: Symbol definitions for the Network Management Listener
::
```

```
$STRUCT NML
```

```
:: Message parsing flag bit definitions (set in NML$GL_PRS_FLGS)
```

```
:: Common parsing flags.
```

```
V<M
  PRS_VMS           : VMS specific function
  PRS_ALL           : Set/clear all parameters
  PRS_QUALIFIER     : NICE command includes a qualifier
  PRS_ENTITY_FOUND  : On multiple entity operations, at least one
                    : entity has been set, shown, etc.
```

```
>
:: Specific entity parsing flags.
```

```
:: Node parsing flags.
```

```
V<M
  8
  PRS_EXEPG        : Executor-only parameter group
  PRS_NODPG        : Executor/node parameter group
  PRS_REMPG        : Remote node parameter group
  PRS_LOCPG        : Loop node parameter group
```

```
>
:: Logging parsing flags.
```

```
V<M
  8
  PRS_EXESNK       : Sink node is executor
  PRS_SKNOD        : Sink node specified
  PRS_KNOSNK       : Known sink nodes
  PRS_EFIPG        : Event filter parameter group
  PRS_ESIPG        : Event sink parameter group
  PRS_EVE          : Event parameter processed
```

```
>
:: Line parsing flags.
```

```
V<M
  8
  PRS_LINE_6       : Line flags (LINS definitions)
  PRS_V2_LINE      : Command parameters are for a line.
  PRS_V2_CIRCUIT   : Command parameters are for a circuit.
  PRS_V2_STA       : Command contains a state change.
```

```
>
:: NICE message parsing constants
```



```
;; Network Management version definitions for message handling
;; (Set in NML$GB_CMD_VER)
```

```
C<
  PHASE2, 1          : Phase II function
  PHASE3_OR_4, 2     : Phase III or IV function
>
```

```
;; NML return codes
```

```
C<,$
  STS_SUC, 1          : Success
  STS_FUN, -1*2       : Unrecognized function or option
  STS_INV, -2*2       : Invalid message format
  STS_PRI, -3*2       : Privilege violation
  STS_SIZ, -4*2       : Message too long
  STS_MPR, -5*2       : Network management program error
  STS_PTY, -6*2       : Unrecognized parameter type
  STS_MVE, -7*2       : Incompatible management version
  STS_CMP, -8*2       : Unrecognized component
  STS_IDE, -9*2       : Invalid identification format
  STS_LCO, -10*2      : Line communication error
  STS_STA, -11*2      : Component in wrong state
  STS_FOP, -13*2      : File open error
  STS_FCO, -14*2      : Invalid file contents
  STS_RES, -15*2      : Resource error
  STS_PVA, -16*2      : Invalid parameter value
  STS_LPR, -17*2      : Line protocol error
  STS_FIO, -18*2      : File i/o error
  STS_MLD, -19*2      : Mirror link disconnected
  STS_ROO, -20*2      : No room for new entry
  STS_MCF, -21*2      : Mirror connect failed
  STS_PNA, -22*2      : Parameter not applicable
  STS_PLO, -23*2      : Parameter value too long
  STS_HAR, -24*2      : Hardware failure
  STS_OPE, -25*2      : Operation failure
  STS_SYS, -26*2      : System-specific network management function not supported
  STS_PGP, -27*2      : Invalid parameter grouping
  STS_BLR, -28*2      : Bad loopback response
  STS_PMS, -29*2      : Parameter missing
>
```

```
;; Network Management parameters
```

```
K
  <
    VERSION, 4        : Network Management version
    DEC_ECO, 0        : DIGITAL ECO number
    USER_ECO, 0       : User ECO number
    FAC_CODE, 505     : Facility code
    SIG_CODE, 505*65536 : Signal code (505*16)
  >
```

```
;; Data definitions
```

```
K<
```



```

: *
: BYTE          : Byte
: WORD          : Word
: LONG          : Longword
: QUAD          : Quadword
: STRING        : String
>
: Special permanent data base record search key parameter codes.
C<
: KEY_LINE, -1  : Line
: KEY_SINK, -2  : Logging sink
:              : Node uses name or address
: KEY_EXE, -3   : Executor node
:              : Object uses name
: KEY_CIR, -4   : Circuit
: KEY_NET, -5   : Protocol Module Network
: KEY_X25_SERV, -6 : X25-Server Module
: KEY_X29_SERV, -7 : X29-Server Module
: KEY_TRACE, -8 : X25-Trace Module
: KEY_NI_CONFIG, -9 : NI Configurator Module
: KEY_X25_ACCESS, -10 : X25 Access Module Network
>

```

```

: Internal entity id codes. Do not reorder. Entries in
: NML$AB_ENTITYDATA table depend on this order.

```

```

C<
: LINE          : Line
: LOGGING       : Logging
: SINK          : Logging sink
: NODE          : Node (by address)
: NODEBYNAME    : Node by name
: LOOPNODE      : Loop node (by name only)
: ADJACENT_NODE : Nodes one hop away, volatile
:              : database only
: EXECUTOR      : Executor node (by address=0)
: OBJECT        : Object
: CIRCUIT       : Circuit
: CIRCUIT_ADJACENT : By circuit, Nodes one hop away,
:              : volatile database only.
: CIRCUIT_ADJ_SRV : By circuit, service adjacencies
: AREA          : Area, volatile database only.
: X25_ACCESS    : X25-Access Module
: PROT_NET      : X25-Protocol Module Networks
: PROT_DTE      : X25-Protocol Module DTEs
: PROT_GRP      : X25-Protocol Module Groups
: X25_SERV      : X25-Server Module
: X25_SERV_DEST : X25-Server Module Destination
: TRACE         : X25-Trace Module
: TRACEPNT      : X25-Trace Module Trecepoint
: X29_SERV      : X29-Server Module
: X29_SERV_DEST : X29-Server Module Destination
: NI_CONFIG     : NI Configurator Module

```

```
LINKS          : Logical links, volatile database only.
PROTOCOL       : Protocol Module
MAXENTITY      : Maximum entity number
>
: Internal information table index codes.
:
(<
SUMMARY        : Summary          (NMASC_INF_SUM)
STATUS         : Status           (NMASC_INF_STA)
CHARACTERISTICS : Characteristics (NMASC_INF_CHA)
COUNTERS       : Counters         (NMASC_INF_COU)
EVENTS         : Events           (NMASC_INF_EVE)
ZERO           : Zero counters
SERVICE       : Service parameters
MAXINFO        : Maximum information type
>
```

E


```

: Network Management Node database definitions. Used for manipulating
: the node permanent database. This database (unlike the other entity
: permanent databases) uses 4 ISAM keys.

```

```

$STRUCT NMN
F KEY_LIS,L      : 3rd alternate key = list node (consists of node
                  : address key concatenated with node
                  : type key)
S KEY_ADD,0,W    : Primary key in record = node address
S KEY_TYP,2,W    : 1st alternate key in record = node type (executor,
                  : remote, or loopnode).
F KEY_NAM,T,6    : 2nd alternate key in record = node name
L NODE_KEYS_LEN : Length of all three keys.
F NOD_PARAMS,A   : Beginning of node's NICE parameters
                  : in the record.

```

```

C <
: Keys for accessing node permanent database.
ADD_KEY_REF, 0   : The primary key = node address
TYP_KEY_REF, 1   : The 1st alternate key = node type
                  : (nml$c typ_exec, nml$c typ_remote,
                  : nml$c typ_loopnode). Overlaps with
                  : node address key.
NAM_KEY_REF, 2   : The 2nd alternate key = node name
LIS_KEY_REF, 3   : The 3rd alternate key = node address
                  : concatenated with node type. Used
                  : to LIST nodes in order by address,
                  : but with exec first, then remotes, and
                  : last loopnodes.

```

```

: Lengths of node permanent database keys

```

```

ADD_KEY_LEN, 2   : The primary key = node address
TYP_KEY_LEN, 2   : The first alternate key overlaps with the
                  : node address key.
NAM_KEY_LEN, 6   : The second alternate key = node name
LIS_KEY_LEN, 4   : The third alternate key = node address
                  : concatenated with node type.

```

```

: Key values for node key = type. The LIST key concatenates the node
: address key with the node type key. This allows the the LIST
: command to get nodes by type and, within type, sequentially by node
: address. The key value is constructed with a zero for the node
: address; hence when you do a $GET of (type OR 0) with a match type
: of GTR, it will get the first node of that type in the file.
: Subsequent sequential reads will return the nodes of that type in
: ascending order by address.

```

```

TYP_EXEC, 0      : type = executor node
TYP_REMOTE, 1    : type = a remote node
TYP_LOOPNODE, 2  : type = a loopnode

```

```

:

```

: Input values to node database routines. Used for determining
: what RMS operations to perform.

PUT_REC, 1 : Do a \$PUT (write a new record)
UPDATE_REC, 2 : Do a \$UPDATE (update an existing record)
DELETE_REC, 3 : Do a \$DELETE (delete a record)
GET_REC, 4 : Do a \$GET (read a record)
>

E

Message segment block (MSB) definitions

STRUCT MSB

F	FLAGS,L	; Flags
V<M		
	CODE_FLD	; Status code present (not used)
	DET_FLD	; Error detail field present (DETAIL)
	MSG_FLD	; Message text field present (TEXT)
	MSG2_FLD	; Second line of message text present (TEXT2)
	ENTD_FLD	; Entity descriptor field present (ENTITY)
	DATA_FLD	; Data descriptor field present (DATA)
	SYSM_FLD	; System message field present (TEXT)
>		
F	,L	
S	CODE,,B	; Status code
F	,L	
S	DETAIL,,W	; Detail
F	TEXT,L	; Status code for text message.
F	TEXT2,L	; Status code for second line of text msg.
F	ENTITY,A	; Entity descriptor address
F	DATA,A	; Data descriptor address
L	LENGTH	; Maximum MSB size

E

: NML internal logging (debugging) flags
: These flags are used to enable logging of specified data to the NML log
: file. The flags are defined by translating the logical name NML\$LOG.

\$STRUCT DBG

```
C<
  NETIO      : Network send/receive logging
  FILEIO     : File read/write logging
  NPARSE     : NPARSE state transition logging
  LOOPIO     : Loopback transmit/receive logging
  ACPQIO     : NETACP QIO logging
  MOPIO      : MOP send/receive logging
  SRVTRC     : Trace service operations
  EVENTS     : Network event (EVL) logging
>

C<.,16,1
  DMPNOD     : Dump node permanent data base file
  DMPLIN     : Dump line permanent data base file
  DMPLOG     : Dump logging permanent data base file
  DMPOBJ     : Dump object permanent data base file
  DMPCIR     : Dump circuit permanent data base file
  DMPX25     : Dump X25 module permanent data base file
  DMPX29     : Dump X29 module permanent data base file
  DMPCNF     : Dump Configurator Module permanent data base file
>
```

E

;
; Parameter semantic table (PST) definitions
;

\$STRUCT PST

F	DATAID,W	; DNA parameter code
F	FORMAT,B	; Parameter format (byte, word, longword, etc.)
F	DATATYPE,B	; Data type code (coded, coded multiple, etc.)
F	MINVALUE,L	; Minimum value or string length
F	MAXVALUE,L	; Maximum value or string length
F	NFBID,L	; ACP parameter identifier
L	ENTRYLEN	; Parameter semantic table entry length

E

Entity information table definitions.

\$STRUCT EIT

F FILEID,B : Permanent data base file id code
F DETAIL,W : NICE error detail entity code
F KEY,W : Permanent data base search key
F DATABASE,B : Volatile data base ID
F SRCH_ID1,L : Volatile data base search key one ID for one entity
F SRCH_ID2,L : Volatile data base search key two ID for one entity

SHOW KNOWN search key ID, length, value, and operator.

F KNO_SRCH_ID1,L : Search key one ID
F KNO_SRCH_LEN1,L : Search key one length
F KNO_SRCH_VAL1,L : Search key one value
F KNO_OPERT,B : Sense search one operator (EQL, NEQ, etc.)

SHOW ACTIVE search key ID, length, value, and operator.

F ACT_SRCH_ID1,L : Search key one ID
F ACT_SRCH_LEN1,L : Search key one length
F ACT_SRCH_VAL1,L : Search key one value
F ACT_OPERT,B : Sense of search one operator (EQL, NEQ, etc.)

F ALLTAB,A : Parameter table for SET ALL
L ENTRYLEN : Entry length

E

Change parameter table definitions.

\$STRUCT CPT

F PSTINDEX,W : Parameter semantic table index
F DEFINE RTN,A : Define routine address
F PURGE RTN,A : Purge routine address
F SET RTN,A : Set routine address
F CLEAR RTN,A : Clear routine address
L ENTRYLEN : Length of table entry

E

: Change dispatch table definitions. This table is used by NMLCHANGE or
: NMLREAD to dispatch to the correct change routine for the entity.

\$STRUCT DT

F DISPATCH,L₁ ; Dispatch routine for this entity
M

: Dispatch table definitions for NICE change operations.

F SEDE_PARSE,L ; Set/Define NPARSE table for parsing the
; NICE command parameters
F CLPU_PARSE,L ; Clear/Purge NPARSE table for parsing the
; NICE command parameters.
F SET_ROUTINES,A,4 ; SET routines for entity.
F CLEAR_ROUTINES,A,4 ; CLEAR routines for entity.
F DEFINE_ROUTINES,A,4 ; DEFINE routines for entity.
F PURGE_ROUTINES,A,4 ; PURGE routines for entity.

P 1

: Dispatch table definitions for NICE read operations.

F SHOW_ROUTINES,A,5 ; SHOW routines for entity.
F LIST_ROUTINES,A,5 ; LIST routines for entity

E

\$STRUCT CHG

: Each of the DT\$ change ROUTINES fields breaks down into the following
: routine offsets. They are offsets in order to make NMLSHR PIC.

F ENTITY,L ; Offset to change routine for single entity
F ENTITY_W_QUAL,L ; Offset to change routine for single entity
; with a qualifier
F KNOWN,L ; Offset to change routine for KNOWN entities.
F KNOWN_W_QUAL,L ; Offset to change routine for KNOWN entities
; with qualifier
L CHG_TABLEN ; Length of dispatch table

E

\$STRUCT RD

: Each of the DT\$ read ROUTINES fields breaks down into the following
: routine offsets. They are offsets in order to make NMLSHR PIC.

F ENTITY,L ; Offset to read routine for single entity
F ENTITY_W_QUAL,L ; Offset to read routine for single entity
; with a qualifier
F KNOWN,L ; Offset to read routine for KNOWN entities.
F KNOWN_W_QUAL,L ; Offset to read routine for KNOWN entities

```
F ACTIVE,L      ; with qualifier
L RD_TABLEN     ; Offset to read routine for ACTIVE entities.
                 ; Length of dispatch table
```

E

\$STRUCT ZER

```
: Each of the ZER$ zero ROUTINES fields breaks down into the following
: routine offsets. They are offsets in order to make NMLSHR PIC.
F DISPATCH,L    ; Dispatch routine for this entity
F ENTITY,L      ; Offset to zero routine for single entities.
F KNOWN,L       ; Offset to zero routine for KNOWN entities.
L ZER_TABLEN    ; Length of dispatch table.
```

E


```

:
: This file defines the event data base ures.
:

```

```

: Event source block definitions.
:

```

```

$STRUCT SRC

```

```

F LENGTH,W           ; Byte count of entire source block
F SINKTYPE,B         ; Sink type code: NMASC_SNK_CON
                        NMASC_SNK_FIL
                        NMASC_SNK_MON
F SRCTYPE,B          ; Source type code: NMASC_ENT_KNO
                        NMASC_ENT_NOD
                        NMASC_ENT_CIR
                        NMASC_ENT_LIN
F SOURCE,T,18         ; Source id string
S NODADR,O,W          ; Source = node (NMASC_ENT_NOD)
                        Node address
                        Source = line or circuit (NMASC_ENT_LIN or
                        NMASC_ENT_CIR)
S IDLENGTH,O,B        ; Source id string length
S ID,,T,17            ; Source id string
F MSKCOUNT,W         ; Count of event blocks (mask sets)
L LENGTH              ; Length of fixed part of source block

```

```

E
:
: Event block definitions.
:

```

```

$STRUCT EVT

```

```

F CLASS,W            ; Event class
F ,W                 ; Spare
F LOGMSK,Q           ; Event log mask
F FILTERMSK,Q        ; Event filter mask
L LENGTH             ; Length of event block

```

```

E
:
: Event table definitions.
:

```

```

$STRUCT ETB

```

```

F CLASS,W            ; Event class
F GLOBAL,A           ; Global filter mask
F NODE,A             ; Node filter mask
F CIRCUIT,A          ; Circuit filter mask
F LINE,A             ; Line filter mask
F MODULE,A           ; Module filter mask

```

L ENTRYLEN

; Length of event table entry

E
:
:
:
:

End of NMLDEF.MDL

0280

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY