```
LLL                 000000000          GGGGGGGGGGGG     IIIIIIIII     NNN           NNN
LLL                 000000000          GGGGGGGGGGGG     IIIIIIIII     NNN           NNN
LLL                 000000000          GGGGGGGGGGGG     IIIIIIIII     NNN           NNN
LLL             000           000   GGG                    III      NNN           NNN
LLL             000           000   GGG                    III      NNN           NNN
LLL             000           000   GGG                    III      NNN           NNN
LLL             000           000   GGG                    III      NNNNNN        NNN
LLL             000           000   GGG                    III      NNNNNN        NNN
LLL             000           000   GGG                    III      NNNNNN        NNN
LLL             000           000   GGG                    III      NNN    NNN    NNN
LLL             000           000   GGG                    III      NNN    NNN    NNN
LLL             000           000   GGG                    III      NNN    NNN    NNN
LLL             000           000   GGG    GGGGGGGGG       III      NNN       NNNNNN
LLL             000           000   GGG    GGGGGGGGG       III      NNN       NNNNNN
LLL             000           000   GGG    GGGGGGGGG       III      NNN       NNNNNN
LLL             000           000   GGG          GGG       III      NNN           NNN
LLL             000           000   GGG          GGG       III      NNN           NNN
LLL             000           000   GGG          GGG       III      NNN           NNN
LLLLLLLLLLLLLLL     000000000          GGGGGGGGG        IIIIIIIII     NNN           NNN
LLLLLLLLLLLLLLL     000000000          GGGGGGGGG        IIIIIIIII     NNN           NNN
LLLLLLLLLLLLLLL     000000000          GGGGGGGGG        IIIIIIIII     NNN           NNN
```

```
   AAAAAA    UU      UU  DDDDDDDD      IIIIII    TTTTTTTTTT
   AAAAAA    UU      UU  DDDDDDDD      IIIIII    TTTTTTTTTT
  AA    AA   UU      UU  DD      DD      II        TT
  AA    AA   UU      UU  DD      DD      II        TT
  AA    AA   UU      UU  DD      DD      II        TT
  AA    AA   UU      UU  DD      DD      II        TT
  AA    AA   UU      UU  DD      DD      II        TT
  AA    AA   UU      UU  DD      DD      II        TT
  AAAAAAAAAA UU      UU  DD      DD      II        TT
  AAAAAAAAAA UU      UU  DD      DD      II        TT
  AA    AA   UU      UU  DD      DD      II        TT        ....
  AA    AA   UU      UU  DD      DD      II        TT        ....
  AA    AA   UUUUUUUUUU  DDDDDDDD      IIIIII    TT        ....
  AA    AA   UUUUUUUUUU  DDDDDDDD      IIIIII    TT        ....

  LL           IIIIII      SSSSSSSS
  LL           IIIIII      SSSSSSSS
  LL             II      SS
  LL             II      SS
  LL             II      SS
  LL             II      SS
  LL             II        SSSSSS
  LL             II        SSSSSS
  LL             II              SS
  LL             II              SS
  LL             II              SS
  LL             II              SS
  LLLLLLLLLL   IIIIII    SSSSSSSS
  LLLLLLLLLL   IIIIII    SSSSSSSS
```

```
;     1        0001  0 MODULE audit (IDENT = 'V04-000',
;     2        0002  0                ADDRESSING_MODE(EXTERNAL = GENERAL)) =
;     3        0003  1 BEGIN
;     4        0004  1
;     5        0005  1 !
;     6        0006  1 !*****************************************************************************
;     7        0007  1 !*                                                                           *
;     8        0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     9        0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    10        0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11        0011  1 !*                                                                           *
;    12        0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13        0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14        0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15        0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16        0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17        0017  1 !*  TRANSFERRED.                                                             *
;    18        0018  1 !*                                                                           *
;    19        0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20        0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21        0021  1 !*  CORPORATION.                                                             *
;    22        0022  1 !*                                                                           *
;    23        0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24        0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25        0025  1 !*                                                                           *
;    26        0026  1 !*                                                                           *
;    27        0027  1 !*****************************************************************************
;    28        0028  1
;    29        0029  1 !++
;    30        0030  1 ! FACILITY:
;    31        0031  1 !
;    32        0032  1 !       LOGIN
;    33        0033  1 !
;    34        0034  1 ! ABSTRACT:
;    35        0035  1 !
;    36        0036  1 !       Performs security auditing functions for LOGINOUT.
;    37        0037  1 !
;    38        0038  1 ! ENVIRONMENT:
;    39        0039  1 !
;    40        0040  1 !       VAX/VMS operating system
;    41        0041  1 !
;    42        0042  1 ! AUTHOR:
;    43        0043  1 !
;    44        0044  1 !       Mark Bramhall, 23-Mar-1984
;    45        0045  1 !
;    46        0046  1 ! MODIFIED BY:
;    47        0047  1 !
;    48        0048  1 !       V03-002 MHB0146         Mark Bramhall           27-Apr-1984
;    49        0049  1 !               Make physical terminal packet optional.
;    50        0050  1 !
;    51        0051  1 !       V03-001 MHB0123         Mark Bramhall            5-Apr-1984
;    52        0052  1 !               Use mandatory audit flag NSA$M_ARG_FLAG_MANDY.
;    53        0053  1 !--
```

```
;    55          0054  1 !
;    56          0055  1 ! Include files
;    57          0056  1 !
;    58          0057  1
;    59          0058  1 LIBRARY 'SYS$LIBRARY:LIB';              ! VAX/VMS system definitions
;    60          0059  1
;    61          0060  1 !
;    62          0061  1 ! Table of contents
;    63          0062  1 !
;    64          0063  1
;    65          0064  1 FORWARD ROUTINE
;    66          0065  1     security_audit:     NOVALUE;        ! Perform a security audit
;    67          0066  1
;    68          0067  1 !
;    69          0068  1 ! External routines
;    70          0069  1 !
;    71          0070  1
;    72          0071  1 EXTERNAL ROUTINE
;    73          0072  1     nsa$event_audit;                  ! Kernel mode auditing routine
;    74          0073  1
;    75          0074  1 !
;    76          0075  1 ! External storage (flags)
;    77          0076  1 !
;    78          0077  1
;    79          0078  1 EXTERNAL
;    80          0079  1     job_type,                         ! Job type (JIB$C_xxx values)
;    81          0080  1     subprocess:         BYTE,         ! True if subprocess
;    82          0081  1     pcb_sts:            BITVECTOR,    ! PCB status (copy of PCB$L_STS)
;    83          0082  1     nsa$gr_alarmvec:    VECTOR [,BYTE], ! Security audit alarm vector
;    84          0083  1     nsa$gr_journvec:    VECTOR [,BYTE]; ! Security audit journal vector
;    85          0084  1
;    86          0085  1 !
;    87          0086  1 ! External storage (auditing data)
;    88          0087  1 !
;    89          0088  1
;    90          0089  1 EXTERNAL
;    91          0090  1     parent_pid,                       ! Parent process PID
;    92          0091  1     ctl$t_nodeaddr:     VECTOR [,BYTE], ! Node address (ASCIC)
;    93          0092  1     fail_password:      VECTOR,       ! Failing password desc
;    94          0093  1     term_name:          VECTOR,       ! Terminal name desc
;    95          0094  1     phy_term_name:      VECTOR,       ! Physical terminal name desc
;    96          0095  1     ctl$t_nodename:     VECTOR [,BYTE], ! Node name (ASCIC)
;    97          0096  1     ctl$t_remoteid:     VECTOR [,BYTE], ! Remote ID (ASCIC)
;    98          0097  1     creator_username:    VECTOR;      ! Creator process username desc
```

```
;  100          0098  1 !
;  101          0099  1 ! Validate NSA$K_RECTYP_LOGx to NSA$B_EVT_LOGx ordering so that we
;  102          0100  1 ! can use a record's type (NSA$K_RECTYP_LOGx) to index into both
;  103          0101  1 ! the security audit alarm vector (NSA$GR_ALARMVEC) and security
;  104          0102  1 ! audit journal vector (NSA$GR_JOURNVEC).  In addition, define the
;  105          0103  1 ! offset (TO_EVT_BYTE_BIAS) to be used to do the indexing.
;  106          0104  1 !
;  107          0105  1
;  108          0106  1 LITERAL
;  109          0107  1     to_evt_byte_bias = $BYTEOFFSET(nsa$b_evt_logb) - nsa$k_rectyp_logb;
;  110          0108  1
;  111          0109  1 $ASSUME($BYTEOFFSET(nsa$b_evt_logb),EQL,nsa$k_rectyp_logb + to_evt_byte_bias);
;  112          0110  1 $ASSUME($BYTEOFFSET(nsa$b_evt_logi),EQL,nsa$k_rectyp_logi + to_evt_byte_bias);
;  113          0111  1 $ASSUME($BYTEOFFSET(nsa$b_evt_logf),EQL,nsa$k_rectyp_logf + to_evt_byte_bias);
;  114          0112  1 $ASSUME($BYTEOFFSET(nsa$b_evt_logo),EQL,nsa$k_rectyp_logo + to_evt_byte_bias);
;  115          0113  1
;  116          0114  1 !
;  117          0115  1 ! Validate NSA$K_RECTYP_LOGx ordering so that we know our "indexed by
;  118          0116  1 ! record type" arrays are in the correct order.  The order should be
;  119          0117  1 ! LOGB, LOGI, LOGF, then LOGO incrementing by 1 each time.  In addition,
;  120          0118  1 ! define the offset (TYPE_INDEX_BIAS) to be used to do the indexing
;  121          0119  1 ! and define the size (TYPE_INDEX_SIZE) of the arrays.
;  122          0120  1 !
;  123          0121  1
;  124          0122  1 LITERAL
;  125          0123  1     type_index_bias = 0 - nsa$k_rectyp_logb,
;  126          0124  1     type_index_size = 4;
;  127          0125  1
;  128          0126  1 $ASSUME(0,EQL,nsa$k_rectyp_logb + type_index_bias);
;  129          0127  1 $ASSUME(1,EQL,nsa$k_rectyp_logi + type_index_bias);
;  130          0128  1 $ASSUME(2,EQL,nsa$k_rectyp_logf + type_index_bias);
;  131          0129  1 $ASSUME(3,EQL,nsa$k_rectyp_logo + type_index_bias);
;  132          0130  1 $ASSUME(4,EQL,type_index_size);
;  133          0131  1
;  134          0132  1 !
;  135          0133  1 ! Validate job type (JOB_TYPE = JIB$C_xxx) ordering so that we know our
;  136          0134  1 ! "indexed by job type" arrays are in the correct order.  The order
;  137          0135  1 ! should be DETACHED, NETWORK, BATCH, LOCAL, DIALUP, then REMOTE
;  138          0136  1 ! incrementing by 1 each time.  An additional catagory (SUBPROCESS_INDEX)
;  139          0137  1 ! for subprocesses (SUBPROCESS = true) is added to correspond to all
;  140          0138  1 ! possible record subtypes.  We then have array entries for subtypes
;  141          0139  1 ! DET, NET, BAT, LOC, DIA, REM, and SUB.  In addition, define the size
;  142          0140  1 ! (JOB_TYPE_INDEX_SIZE) of the arrays.
;  143          0141  1 !
;  144          0142  1
;  145          0143  1 LITERAL
;  146          0144  1     subprocess_index = 6,
;  147          0145  1     job_type_index_size = 7;
;  148          0146  1
;  149          0147  1 $ASSUME(0,EQL,jib$c_detached);          ! For NSA$K_RECTYP_LOGx_DET
;  150          0148  1 $ASSUME(1,EQL,jib$c_network);           ! For NSA$K_RECTYP_LOGx_NET
;  151          0149  1 $ASSUME(2,EQL,jib$c_batch);             ! For NSA$K_RECTYP_LOGx_BAT
;  152          0150  1 $ASSUME(3,EQL,jib$c_local);             ! For NSA$K_RECTYP_LOGx_LOC
;  153          0151  1 $ASSUME(4,EQL,jib$c_dialup);            ! For NSA$K_RECTYP_LOGx_DIA
;  154          0152  1 $ASSUME(5,EQL,jib$c_remote);            ! For NSA$K_RECTYP_LOGx_REM
;  155          0153  1 $ASSUME(6,EQL,subprocess_index);        ! For NSA$K_RECTYP_LOGx_SUB
;  156          0154  1 $ASSUME(7,EQL,job_type_index_size);
```

```
;  158          0155  1 !
;  159          0156  1 ! Pure storage of event masks and record subtypes (via BINDs)
;  160          0157  1 !
;  161          0158  1
;  162          0159  1 BIND
;  163          0160  1     audit_vector_masks =                ! Audit vector event mask array
;  164          0161  1         UPLIT BYTE(nsa$m_evt_log_det,
;  165          0162  1                    nsa$m_evt_log_net,
;  166          0163  1                    nsa$m_evt_log_bat,
;  167          0164  1                    nsa$m_evt_log_loc,
;  168          0165  1                    nsa$m_evt_log_dia,
;  169          0166  1                    nsa$m_evt_log_rem,
;  170          0167  1                    nsa$m_evt_log_sub)
;  171          0168  1             : VECTOR [job_type_index_size,BYTE];
;  172          0169  1
;  173          0170  1 BIND
;  174          0171  1     audit_logb_subtypes =               ! Audit LOGB record subtype array
;  175          0172  1         UPLIT WORD(nsa$k_rectyp_logb_det,
;  176          0173  1                    nsa$k_rectyp_logb_net,
;  177          0174  1                    0,                  !  LOGB w/ BAT doesn't exist
;  178          0175  1                    nsa$k_rectyp_logb_loc,
;  179          0176  1                    nsa$k_rectyp_logb_dia,
;  180          0177  1                    nsa$k_rectyp_logb_rem,
;  181          0178  1                    0)                  !  LOGB w/ SUB doesn't exist
;  182          0179  1             : VECTOR [job_type_index_size,WORD];
;  183          0180  1
;  184          0181  1 BIND
;  185          0182  1     audit_logi_subtypes =               ! Audit LOGI record subtype array
;  186          0183  1         UPLIT WORD(nsa$k_rectyp_logi_det,
;  187          0184  1                    nsa$k_rectyp_logi_net,
;  188          0185  1                    nsa$k_rectyp_logi_bat,
;  189          0186  1                    nsa$k_rectyp_logi_loc,
;  190          0187  1                    nsa$k_rectyp_logi_dia,
;  191          0188  1                    nsa$k_rectyp_logi_rem,
;  192          0189  1                    nsa$k_rectyp_logi_sub)
;  193          0190  1             : VECTOR [job_type_index_size,WORD];
;  194          0191  1
;  195          0192  1 BIND
;  196          0193  1     audit_logf_subtypes =               ! Audit LOGF record subtype array
;  197          0194  1         UPLIT WORD(nsa$k_rectyp_logf_det,
;  198          0195  1                    nsa$k_rectyp_logf_net,
;  199          0196  1                    nsa$k_rectyp_logf_bat,
;  200          0197  1                    nsa$k_rectyp_logf_loc,
;  201          0198  1                    nsa$k_rectyp_logf_dia,
;  202          0199  1                    nsa$k_rectyp_logf_rem,
;  203          0200  1                    nsa$k_rectyp_logf_sub)
;  204          0201  1             : VECTOR [job_type_index_size,WORD];
;  205          0202  1
;  206          0203  1 BIND
;  207          0204  1     audit_logo_subtypes =               ! Audit LOGO record subtype array
;  208          0205  1         UPLIT WORD(nsa$k_rectyp_logo_det,
;  209          0206  1                    nsa$k_rectyp_logo_net,
;  210          0207  1                    nsa$k_rectyp_logo_bat,
;  211          0208  1                    nsa$k_rectyp_logo_loc,
;  212          0209  1                    nsa$k_rectyp_logo_dia,
;  213          0210  1                    nsa$k_rectyp_logo_rem,
;  214          0211  1                    nsa$k_rectyp_logo_sub)
```

```
;  215          0212  1          : VECTOR [job_type_index_size,WORD];
;  216          0213  1
;  217          0214  1 BIND
;  218          0215  1     audit_subtypes =                    ! Audit type to subtype array array
;  219          0216  1         UPLIT LONG(audit_logb_subtypes,
;  220          0217  1                    audit_logi_subtypes,
;  221          0218  1                    audit_logf_subtypes,
;  222          0219  1                    audit_logo_subtypes)
;  223          0220  1         : VECTOR [type_index_size,LONG];
```

```
;  225          0221  1 !
;  226          0222  1 ! Packet building bits and prototype masks
;  227          0223  1 !
;  228          0224  1
;  229          0225  1 LITERAL
;  230          0226  1     packet_parent_pid       = 0,        ! Packet w/ Parent process's PID
;  231          0227  1     packet_node_address     = 1,        ! Packet w/ Node address
;  232          0228  1     packet_logf_status      = 2,        ! Packet w/ Login failure status
;  233          0229  1     packet_fail_password    = 3,        ! Packet w/ Failing password
;  234          0230  1     packet_term_name        = 4,        ! Packet w/ Terminal name
;  235          0231  1     packet_phy_term_name    = 5,        ! Packet w/ Physical terminal name
;  236          0232  1     packet_node_name        = 6,        ! Packet w/ Node name
;  237          0233  1     packet_remote_id        = 7,        ! Packet w/ Remote ID
;  238          0234  1     packet_creator_username = 8,        ! Packet w/ Creator process's username
;  239          0235  1     max_pos_packets         = 9,        ! Max number of possible packets
;  240          0236  1     max_packet_size         = 2 + 2 + 8; ! Max packet size (type + mech + quad)
;  241          0237  1
;  242          0238  1 LITERAL
;  243          0239  1     det_packets = 0,                                ! DET: <nothing>
;  244          0240  2     net_packets = (1 ^ packet_node_address)         ! NET: Node address
;  245          0241  2                 OR (1 ^ packet_node_name)           !      Node name
;  246          0242  1                 OR (1 ^ packet_remote_id),          !      Remote ID
;  247          0243  1     bat_packets = 0,                                ! BAT: <nothing>
;  248          0244  2     loc_packets = (1 ^ packet_term_name)            ! LOC: Terminal name
;  249          0245  1                 OR (1 ^ packet_phy_term_name),      !      Physical terminal name
;  250          0246  2     dia_packets = (1 ^ packet_term_name)            ! DIA: Terminal name
;  251          0247  1                 OR (1 ^ packet_phy_term_name),      !      Physical terminal name
;  252          0248  2     rem_packets = (1 ^ packet_node_address)         ! REM: Node address
;  253          0249  2                 OR (1 ^ packet_term_name)           !      Terminal name
;  254          0250  2                 OR (1 ^ packet_node_name)           !      Node name
;  255          0251  1                 OR (1 ^ packet_remote_id),          !      Remote ID
;  256          0252  1     sub_packets = (1 ^ packet_parent_pid);          ! SUB: Parent process's PID
```

```
;  258          0253  1 !
;  259          0254  1 ! Pure storage of packet masks (via BINDs)
;  260          0255  1 !
;  261          0256  1
;  262          0257  1 BIND
;  263          0258  1     audit_logb_packets =                    ! Audit LOGB packet mask array
;  264          0259  2         UPLIT WORD(det_packets OR (1 ^ packet_fail_password)
;  265          0260  1                                OR (1 ^ packet_creator_username),
;  266          0261  1                    net_packets OR (1 ^ packet_fail_password),
;  267          0262  1                    0,                      !  LOGB w/ BAT doesn't exist
;  268          0263  1                    loc_packets OR (1 ^ packet_fail_password),
;  269          0264  1                    dia_packets OR (1 ^ packet_fail_password),
;  270          0265  1                    rem_packets OR (1 ^ packet_fail_password),
;  271          0266  1                    0)                      !  LOGB w/ SUB doesn't exist
;  272          0267  1             : VECTOR [job_type_index_size,WORD];
;  273          0268  1
;  274          0269  1 BIND
;  275          0270  1     audit_logi_packets =                    ! Audit LOGI packet mask array
;  276          0271  1         UPLIT WORD(det_packets OR (1 ^ packet_creator_username),
;  277          0272  1                    net_packets,
;  278          0273  1                    bat_packets,
;  279          0274  1                    loc_packets,
;  280          0275  1                    dia_packets,
;  281          0276  1                    rem_packets,
;  282          0277  1                    sub_packets)
;  283          0278  1             : VECTOR [job_type_index_size,WORD];
;  284          0279  1
;  285          0280  1 BIND
;  286          0281  1     audit_logf_packets =                    ! Audit LOGF packet mask array
;  287          0282  2         UPLIT WORD(det_packets OR (1 ^ packet_logf_status)
;  288          0283  1                                OR (1 ^ packet_creator_username),
;  289          0284  1                    net_packets OR (1 ^ packet_logf_status),
;  290          0285  1                    bat_packets OR (1 ^ packet_logf_status),
;  291          0286  1                    loc_packets OR (1 ^ packet_logf_status),
;  292          0287  1                    dia_packets OR (1 ^ packet_logf_status),
;  293          0288  1                    rem_packets OR (1 ^ packet_logf_status),
;  294          0289  1                    sub_packets OR (1 ^ packet_logf_status))
;  295          0290  1             : VECTOR [job_type_index_size,WORD];
;  296          0291  1
;  297          0292  1 BIND
;  298          0293  1     audit_logo_packets =                    ! Audit LOGO packet mask array
;  299          0294  1         UPLIT WORD(det_packets,
;  300          0295  1                    net_packets,
;  301          0296  1                    bat_packets,
;  302          0297  1                    loc_packets,
;  303          0298  1                    dia_packets,
;  304          0299  1                    rem_packets,
;  305          0300  1                    sub_packets)
;  306          0301  1             : VECTOR [job_type_index_size,WORD];
;  307          0302  1
;  308          0303  1 BIND
;  309          0304  1     audit_packets =                         ! Audit type to packet array array
;  310          0305  1         UPLIT LONG(audit_logb_packets,
;  311          0306  1                    audit_logi_packets,
;  312          0307  1                    audit_logf_packets,
;  313          0308  1                    audit_logo_packets)
;  314          0309  1             : VECTOR [type_index_size,LONG];
```

```
;  316     0310  1 GLOBAL ROUTINE security_audit (record_type, logf_status): NOVALUE =
;  317     0311  1
;  318     0312  1 !+++
;  319     0313  1 !
;  320     0314  1 !       Optionally perform a security audit.
;  321     0315  1 !
;  322     0316  1 ! Inputs:
;  323     0317  1 !
;  324     0318  1 !       record_type = Audit record type (NSA$K_RECTYP_LOGx)
;  325     0319  1 !       logf_status = Login failure status for LOGF records
;  326     0320  1 !
;  327     0321  1 ! Outputs:
;  328     0322  1 !
;  329     0323  1 !       None.
;  330     0324  1 !
;  331     0325  1 !---
;  332     0326  1
;  333     0327  2 BEGIN
;  334     0328  2
;  335     0329  2 LOCAL
;  336     0330  2     arglist:                                ! Argument list for NSA$EVENT_AUDIT
;  337     0331  2         BLOCK [nsa$k_arghdr_length + (max_packet_size * max_pos_packets),BYTE],
;  338     0332  2     type_index,                             ! Index into type arrays
;  339     0333  2     job_type_index,                         ! Index into job type arrays
;  340     0334  2     audit_flag,                             ! Audit flag
;  341     0335  2     packets:        BITVECTOR [32],         ! Packets to insert flags
;  342     0336  2     arglist_ptr;                            ! Packet fill in pointer
;  343     0337  2
;  344     0338  2 CH$FILL(0, %ALLOCATION(arglist), arglist); ! Clear out the argument list
;  345     0339  2
;  346     0340  2 type_index = .record_type;                  ! Fetch the type index
;  347     0341  2
;  348     0342  2 job_type_index = .job_type;                 ! Assume job type will be the index
;  349     0343  2 IF .subprocess                              ! But, if a subprocess,
;  350     0344  2 THEN job_type_index = subprocess_index;     !  use the subprocess index
;  351     0345  2
;  352     0346  2 audit_flag = 0;                             ! Assume no audit initially
;  353     0347  2
;  354     0348  2 IF .pcb_sts [$BITPOSITION(pcb$v_secaudit)] ! If mandatory auditing,
;  355     0349  2 THEN audit_flag = nsa$m_arg_flag_mandy;     !  perform mandatory audit
;  356     0350  2
;  357     0351  3 IF (                                       ! If alarm auditing,
;  358     0352  3     .nsa$gr_alarmvec [.type_index + to_evt_byte_bias]
;  359     0353  3     AND
;  360     0354  3     .audit_vector_masks [.job_type_index]
;  361     0355  2    ) NEQ 0
;  362     0356  2 THEN audit_flag =                           !  perform alarm audit
;  363     0357  2         .audit_flag OR nsa$m_arg_flag_alarm;
;  364     0358  2
;  365     0359  3 IF (                                       ! If journal auditing,
;  366     0360  3     .nsa$gr_journvec [.type_index + to_evt_byte_bias]
;  367     0361  3     AND
;  368     0362  3     .audit_vector_masks [.job_type_index]
;  369     0363  2    ) NEQ 0
;  370     0364  2 THEN audit_flag =                           !  perform journal audit
;  371     0365  2         .audit_flag OR nsa$m_arg_flag_journ;
;  372     0366  2
```

```
;  373          0367  2 IF .audit_flag EQL 0                    ! If no audit requested,
;  374          0368  2 THEN RETURN;                             !  simply exit
;  375          0369  2
;  376          0370  2 arglist [nsa$w_arg_type] = .type_index;          ! Set audit record type
;  377          0371  2 arglist [nsa$w_arg_subtype] =                    ! Set audit record subtype
;  378          0372  2     .VECTOR [ .audit_subtypes [.type_index + type_index_bias],
;  379          0373  2               .job_type_index; ,WORD];
;  380          0374  2 arglist [nsa$b_arg_flag] = .audit_flag;          ! Set audit flags
;  381          0375  2 arglist [nsa$b_arg_pktnum] = 0;                  ! Set no packets initially
;  382          0376  2
;  383          0377  2 packets =                                       ! Get packets to do BITVECTOR
;  384          0378  2     .VECTOR [ .audit_packets [.type_index + type_index_bias],
;  385          0379  2               .job_type_index; ,WORD];
;  386          0380  2 arglist_ptr = arglist [nsa$t_arg_list];          ! Address packet(s) in arg list
;  387          0381  2
;  388          0382  2 IF .packets [packet_parent_pid]                 ! Do parent process's PID?
;  389          0383  2 THEN
;  390          0384  3     BEGIN
;  391          0385  3     (.arglist_ptr) <0,16> = nsa$k_pkttyp_epid;
;  392          0386  3     arglist_ptr = .arglist_ptr + 2;
;  393          0387  3     (.arglist_ptr) <0,16> = nsa$k_arg_mech_long;
;  394          0388  3     arglist_ptr = .arglist_ptr + 2;
;  395          0389  3     (.arglist_ptr) <0,32> = .parent_pid; ! Set parent process's PID
;  396          0390  3     arglist_ptr = .arglist_ptr + 4;
;  397          0391  3     arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  398          0392  2     END;
;  399          0393  2
;  400          0394  2 IF .packets [packet_node_address]               ! Do node address?
;  401          0395  2 THEN
;  402          0396  3     BEGIN
;  403          0397  3     (.arglist_ptr) <0,16> = nsa$k_pkttyp_nodeid;
;  404          0398  3     arglist_ptr = .arglist_ptr + 2;
;  405          0399  3     (.arglist_ptr) <0,16> = nsa$k_arg_mech_quad;
;  406          0400  3     arglist_ptr = .arglist_ptr + 2;
;  407          0401  3     CH$COPY(.ctl$t_nodeaddr [0],                ! Set node address
;  408          0402  3             ctl$t_nodeaddr [1],                 !  from control region
;  409          0403  3             0,8,.arglist_ptr);                  !   as a quadword
;  410          0404  3     arglist_ptr = .arglist_ptr + 8;
;  411          0405  3     arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  412          0406  2     END;
;  413          0407  2
;  414          0408  2 IF .packets [packet_logf_status]               ! Do login failure status?
;  415          0409  2 THEN
;  416          0410  3     BEGIN
;  417          0411  3     (.arglist_ptr) <0,16> = nsa$k_pkttyp_status;
;  418          0412  3     arglist_ptr = .arglist_ptr + 2;
;  419          0413  3     (.arglist_ptr) <0,16> = nsa$k_arg_mech_long;
;  420          0414  3     arglist_ptr = .arglist_ptr + 2;
;  421          0415  3     (.arglist_ptr) <0,32> = .logf_status; ! Set login failure status
;  422          0416  3     arglist_ptr = .arglist_ptr + 4;
;  423          0417  3     arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  424          0418  2     END;
;  425          0419  2
;  426          0420  2 IF .packets [packet_fail_password]             ! Do failing password?
;  427          0421  2 THEN
;  428          0422  3     BEGIN
;  429          0423  3     (.arglist_ptr) <0,16> = nsa$k_pkttyp_password;
```

```
;  430          0424  3      arglist_ptr = .arglist_ptr + 2;
;  431          0425  3      (.arglist_ptr) <0,16> = nsa$k_arg_mech_adescr;
;  432          0426  3      arglist_ptr = .arglist_ptr + 2;
;  433          0427  3      (.arglist_ptr) <0,32> = fail_password; ! Set failing password desc addr
;  434          0428  3      arglist_ptr = .arglist_ptr + 4;
;  435          0429  3      arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  436          0430  2      END;
;  437          0431  2
;  438          0432  2  IF .packets [packet_term_name]          ! Do terminal name?
;  439          0433  2  THEN
;  440          0434  3      BEGIN
;  441          0435  3      (.arglist_ptr) <0,16> = nsa$k_pkttyp_devnam;
;  442          0436  3      arglist_ptr = .arglist_ptr + 2;
;  443          0437  3      (.arglist_ptr) <0,16> = nsa$k_arg_mech_adescr;
;  444          0438  3      arglist_ptr = .arglist_ptr + 2;
;  445          0439  3      (.arglist_ptr) <0,32> = term_name;  ! Set terminal name desc addr
;  446          0440  3      arglist_ptr = .arglist_ptr + 4;
;  447          0441  3      arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  448          0442  2      END;
;  449          0443  2
;  450          0444  2  IF .packets [packet_phy_term_name]      ! Do physical terminal name?
;  451          0445  2  AND CH$NEQ(.phy_term_name [0], .phy_term_name [1],
;  452          0446  2              .term_name [0], .term_name [1], 0)
;  453          0447  2  THEN
;  454          0448  3      BEGIN
;  455          0449  3      (.arglist_ptr) <0,16> = nsa$k_pkttyp_devnam;
;  456          0450  3      arglist_ptr = .arglist_ptr + 2;
;  457          0451  3      (.arglist_ptr) <0,16> = nsa$k_arg_mech_adescr;
;  458          0452  3      arglist_ptr = .arglist_ptr + 2;
;  459          0453  3      (.arglist_ptr) <0,32> = phy_term_name; ! Set physical term name desc addr
;  460          0454  3      arglist_ptr = .arglist_ptr + 4;
;  461          0455  3      arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  462          0456  2      END;
;  463          0457  2
;  464          0458  2  IF .packets [packet_node_name]          ! Do node name?
;  465          0459  2  THEN
;  466          0460  3      BEGIN
;  467          0461  3      (.arglist_ptr) <0,16> = nsa$k_pkttyp_nodenam;
;  468          0462  3      arglist_ptr = .arglist_ptr + 2;
;  469          0463  3      (.arglist_ptr) <0,16> = nsa$k_arg_mech_descr;
;  470          0464  3      arglist_ptr = .arglist_ptr + 2;
;  471          0465  3      (.arglist_ptr) <0,32> = .ctl$t_nodename [0]; ! Set node name length
;  472          0466  3      arglist_ptr = .arglist_ptr + 4;
;  473          0467  3      (.arglist_ptr) <0,32> = ctl$t_nodename [1]; !  and address
;  474          0468  3      arglist_ptr = .arglist_ptr + 4;
;  475          0469  3      arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  476          0470  2      END;
;  477          0471  2
;  478          0472  2  IF .packets [packet_remote_id]          ! Do remote ID?
;  479          0473  2  THEN
;  480          0474  3      BEGIN
;  481          0475  3      (.arglist_ptr) <0,16> = nsa$k_pkttyp_usernam;
;  482          0476  3      arglist_ptr = .arglist_ptr + 2;
;  483          0477  3      (.arglist_ptr) <0,16> = nsa$k_arg_mech_descr;
;  484          0478  3      arglist_ptr = .arglist_ptr + 2;
;  485          0479  3      (.arglist_ptr) <0,32> = .ctl$t_remoteid [0]; ! Set remote ID length
;  486          0480  3      arglist_ptr = .arglist_ptr + 4;
```

```
;  487          0481  3     (.arglist_ptr) <0,32> = ctl$t_remoteid [1]; !  and address
;  488          0482  3     arglist_ptr = .arglist_ptr + 4;
;  489          0483  3     arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  490          0484  2     END;
;  491          0485  2
;  492          0486  2 IF .packets [packet_creator_username]   ! Do creator process's username?
;  493          0487  2 THEN
;  494          0488  3     BEGIN
;  495          0489  3     (.arglist_ptr) <0,16> = nsa$k_pkttyp_usernam;
;  496          0490  3     arglist_ptr = .arglist_ptr + 2;
;  497          0491  3     (.arglist_ptr) <0,16> = nsa$k_arg_mech_adescr;
;  498          0492  3     arglist_ptr = .arglist_ptr + 2;
;  499          0493  3     (.arglist_ptr) <0,32> = creator_username; ! Set creator username desc addr
;  500          0494  3     arglist_ptr = .arglist_ptr + 4;
;  501          0495  3     arglist [nsa$b_arg_pktnum] = .arglist [nsa$b_arg_pktnum] + 1;
;  502          0496  2     END;
;  503          0497  2
;  504          0498  2 arglist [nsa$l_arg_count] = (.arglist_ptr - (arglist + 4)) / 4; ! Set # args
;  505          0499  2
;  506        P 0500  2 $CMKRNL(ROUTIN = nsa$event_audit,       : Go do the actual audit
;  507          0501  2         ARGLST = arglist);
;  508          0502  2
;  509          0503  1 END;


                                                        .TITLE  AUDIT
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                    20  08  02  04  01  10  40  00000 P.AAA:  .BYTE   64, 16, 1, 4, 2, 8, 32                          ;
                                                00007       .BLKB   1
0000  0003  0001  0002  0000  0004  0005  00008 P.AAB:  .WORD   5, 4, 0, 2, 1, 3, 0                                ;
0006  0004  0002  0003  0001  0005  0007  00016 P.AAC:  .WORD   7, 5, 1, 3, 2, 4, 6                                ;
0006  0004  0002  0003  0001  0005  0007  00024 P.AAD:  .WORD   7, 5, 1, 3, 2, 4, 6                                ;
0006  0004  0002  0003  0001  0005  0007  00032 P.AAE:  .WORD   7, 5, 1, 3, 2, 4, 6                                ;
  00000000' 00000000' 00000000' 00000000' 00040 P.AAF:  .ADDRESS AUDIT_LOGB_SUBTYPES, -                           ;
                                                        AUDIT_LOGI_SUBTYPES, AUDIT_LOGF_SUBTYPES, -               ;
                                                        AUDIT_LOGO_SUBTYPES                                       ;
0000  00DA  0038  0038  0000  00CA  0108  00050 P.AAG:  .WORD   264, 202, 0, 56, 56, 218, 0                        ;
0001  00D2  0030  0030  0000  00C2  0100  0005E P.AAH:  .WORD   256, 194, 0, 48, 48, 210, 1                        ;
0005  00D6  0034  0034  0004  00C6  0104  0006C P.AAI:  .WORD   260, 198, 4, 52, 52, 214, 5                        ;
0001  00D2  0030  0030  0000  00C2  0000  0007A P.AAJ:  .WORD   0, 194, 0, 48, 48, 210, 1                          ;
  00000000' 00000000' 00000000' 00000000' 00088 P.AAK:  .ADDRESS AUDIT_LOGB_PACKETS, AUDIT_LOGI_PACKETS, -       ;
                                                        AUDIT_LOGF_PACKETS, AUDIT_LOGO_PACKETS                    ;

                                                AUDIT_VECTOR_MASKS= P.AAA                                         ;
                                                AUDIT_LOGB_SUBTYPES=P.AAB
                                                AUDIT_LOGI_SUBTYPES=P.AAC
                                                AUDIT_LOGF_SUBTYPES=P.AAD
                                                AUDIT_LOGO_SUBTYPES=P.AAE
                                                AUDIT_SUBTYPES=     P.AAF
                                                AUDIT_LOGB_PACKETS= P.AAG
                                                AUDIT_LOGI_PACKETS= P.AAH
                                                AUDIT_LOGF_PACKETS= P.AAI
                                                AUDIT_LOGO_PACKETS= P.AAJ
                                                AUDIT_PACKETS=      P.AAK
```

```
                                                        .EXTRN  NSA$EVENT_AUDIT
                                                        .EXTRN  JOB_TYPE, SUBPROCESS
                                                        .EXTRN  PCB_STS, NSA$GR_ALARMVEC
                                                        .EXTRN  NSA$GR_JOURNVEC
                                                        .EXTRN  PARENT_PID, CTL$T_NODEADDR
                                                        .EXTRN  FAIL_PASSWORD, TERM_NAME
                                                        .EXTRN  PHY_TERM_NAME, CTL$T_NODENAME
                                                        .EXTRN  CTL$T_REMOTEID, CREATOR_USERNAME
                                                        .EXTRN  SYS$CMKRNL

                                                        .PSECT  $CODE$,NOWRT,2

                                        07FC 00000      .ENTRY  SECURITY_AUDIT. Save R2,R3,R4,R5,R6,R7,R8,- ; 0310
                                                                R9,R10
                                5A      0000'   CF  9E 00002    MOVAB   AUDIT_VECTOR_MASKS, R10                 ;
                                59 00000000G    00  9E 00007    MOVAB   PHY_TERM_NAME, R9                       ;
                                58 00000000G    00  9E 0000E    MOVAB   TERM_NAME, R8                           ;
                                5E        88    AE  9E 00015    MOVAB   -120(SP), SP                            ;
    0078    8F          00      6E              00  2C 00019    MOVC5   #0, (SP), #0, #120, ARGLIST             ; 0338
                                6E                  00020       ;
                                51        04    AC  D0 00021    MOVL    RECORD_TYPE, TYPE_INDEX                 ; 0340
                                50 00000000G    00  D0 00025    MOVL    JOB_TYPE, JOB_TYPE_INDEX                ; 0342
                                03 00000000G    00  E9 0002C    BLBC    SUBPROCESS, 1$                          ; 0343
                                50                  06  D0 00033    MOVL    #6, JOB_TYPE_INDEX                  ; 0344
                                                    52  D4 00036 1$:  CLRL    AUDIT_FLAG                      ; 0346
                        03 00000000G    00          03  E1 00038    BBC     #3, PCB_STS+3, 2$               ; 0348
                                52                  04  D0 00040    MOVL    #4, AUDIT_FLAG                  ; 0349
                            6A40 00000000G0041      93  00043 2$:  BITB    NSA$GR_ALARMVEC[TYPE_INDEX], -  ; 0355
                                                                    AUDIT_VECTOR_MASKS[JOB_TYPE_INDEX]
                                                    03  13  0004C    BEQL    3$                              ;
                                52                  01  88  0004E    BISB2   #1, AUDIT_FLAG                  ; 0357
                            6A40 00000000G0041      93  00051 3$:  BITB    NSA$GR_JOURNVEC[TYPE_INDEX], -  ; 0363
                                                                    AUDIT_VECTOR_MASKS[JOB_TYPE_INDEX]
                                                    03  13  0005A    BEQL    4$                              ;
                                52                  02  88  0005C    BISB2   #2, AUDIT_FLAG                  ; 0365
                                                    52  D5  0005F 4$:  TSTL    AUDIT_FLAG                      ; 0367
                                                    01  12  00061    BNEQ    5$                              ;
                                                        04  00063    RET                                     ;
                        04      AE                  51  B0  00064 5$:  MOVW    TYPE_INDEX, ARGLIST+4           ; 0370
                                53          30 AA41     D0  00068    MOVL    AUDIT_SUBTYPES-16[TYPE_INDEX], R3 ; 0372
                        06      AE              6340    B0  0006D    MOVW    (R3)[JOB_TYPE_INDEX], ARGLIST+6 ;
                        08      AE                  52  9B  00072    MOVZBW  AUDIT_FLAG, ARGLIST+8           ; 0374
                                51          78 AA41     D0  00076    MOVL    AUDIT_PACKETS-16[TYPE_INDEX], R1 ; 0378
                                57              6140    3C  0007B    MOVZWL  (R1)[JOB_TYPE_INDEX], PACKETS   ;
                                56          0C      AE  9E  0007F    MOVAB   ARGLIST+12, ARGLIST_PTR         ; 0380
                                11                  57  E9  00083    BLBC    PACKETS, 6$                     ; 0382
                                86 00020011         8F  D0  00086    MOVL    #131089, (ARGLIST_PTR)+         ; 0385
                                86 00000000G        00  D0  0008D    MOVL    PARENT_PID, (ARGLIST_PTR)+      ; 0389
                                            09      AE  96  00094    INCB    ARGLIST+9                       ; 0391
                        1E              57              01  E1  00097 6$:  BBC     #1, PACKETS, 7$             ; 0394
                                86 00030010         8F  D0  0009B    MOVL    #196624, (ARGLIST_PTR)+         ; 0397
                                50 00000000G        00  9A  000A2    MOVZBL  CTL$T_NODEADDR, R0              ; 0401
    08          00 00000000G    00                  50  2C  000A9    MOVC5   R0, CTL$T_NODEADDR+1, #0, #8, (ARGLIST_PTR) ; 0403
                                                        66  000B2    ;
                                56                  08  C0  000B3    ADDL2   #8, ARGLIST_PTR                 ; 0404
                                            09      AE  96  000B6    INCB    ARGLIST+9                       ; 0405
                        0E              57              02  E1  000B9 7$:  BBC     #2, PACKETS, 8$             ; 0408
```

```
                                86 00020013   8F  D0 000BD          MOVL    #131091, (ARGLIST_PTR)+                                 ; 0411
                                86       08   AC  D0 000C4          MOVL    LOGF_STATUS, (ARGLIST_PTR)+                             ; 0415
                                         09   AE  96 000C8          INCB    ARGLIST+9                                               ; 0417
                          11    57           03  E1 000CB 8$:      BBC     #3, PACKETS, 9$                                         ; 0420
                                86 0005000B   8F  D0 000CF          MOVL    #327691, (ARGLIST_PTR)+                                 ; 0423
                                86 00000000G  00  9E 000D6          MOVAB   FAIL_PASSWORD, (ARGLIST_PTR)+                           ; 0427
                                         09   AE  96 000DD          INCB    ARGLIST+9                                               ; 0429
                          0D    57           04  E1 000E0 9$:      BBC     #4, PACKETS, 10$                                        ; 0432
                                86 00050005   8F  D0 000E4          MOVL    #327685, (ARGLIST_PTR)+                                 ; 0435
                                86       68   9E  00EB             MOVAB   TERM_NAME, (ARGLIST_PTR)+                               ; 0439
                                         09   AE  96 000EE          INCB    ARGLIST+9                                               ; 0441
                          1D    57           05  E1 000F1 10$:     BBC     #5, PACKETS, 11$                                        ; 0444
                                51       04   A9  D0 000F5          MOVL    PHY_TERM_NAME+4, R1                                     ; 0445
                                50       04   A8  D0 000F9          MOVL    TERM_NAME+4, R0                                         ; 0446
                    68    00    61           69  2D 000FD          CMPC5   PHY_TERM_NAME, (R1), #0, TERM_NAME, (R0)                ; 0445
                                               60     00102
                                               0D  13  00103        BEQL    11$                                                     ;
                                86 00050005   8F  D0 00105          MOVL    #327685, (ARGLIST_PTR)+                                 ; 0449
                                86             69  9E 0010C          MOVAB   PHY_TERM_NAME, (ARGLIST_PTR)+                           ; 0453
                                         09   AE  96 0010F          INCB    ARGLIST+9                                               ; 0455
                          18    57           06  E1 00112 11$:     BBC     #6, PACKETS, 12$                                        ; 0458
                                86 00040009   8F  D0 00116          MOVL    #262153, (ARGLIST_PTR)+                                 ; 0461
                                86 00000000G  00  9A 0011D          MOVZBL  CTL$T_NODENAME, (ARGLIST_PTR)+                          ; 0465
                                86 00000000G  00  9E 00124          MOVAB   CTL$T_NODENAME+1, (ARGLIST_PTR)+                        ; 0467
                                         09   AE  96 0012B          INCB    ARGLIST+9                                               ; 0469
                                         57   95  0012E 12$:      TSTB    PACKETS                                                 ; 0472
                                         18   18  00130            BGEQ    13$                                                     ;
                                86 0004000A   8F  D0 00132          MOVL    #262154, (ARGLIST_PTR)+                                 ; 0475
                                86 00000000G  00  9A 00139          MOVZBL  CTL$T_REMOTEID, (ARGLIST_PTR)+                          ; 0479
                                86 00000000G  00  9E 00140          MOVAB   CTL$T_REMOTEID+1, (ARGLIST_PTR)+                        ; 0481
                                         09   AE  96 00147          INCB    ARGLIST+9                                               ; 0483
                          11    57           08  E1 0014A 13$:     BBC     #8, PACKETS, 14$                                        ; 0486
                                86 0005000A   8F  D0 0014E          MOVL    #327690, (ARGLIST_PTR)+                                 ; 0489
                                86 00000000G  00  9E 00155          MOVAB   CREATOR_USERNAME, (ARGLIST_PTR)+                        ; 0493
                                         09   AE  96 0015C          INCB    ARGLIST+9                                               ; 0495
                                50       04   AE  9E 0015F 14$:     MOVAB   ARGLIST+4, R0                                           ; 0498
                                56           50  C2 00163          SUBL2   R0, R6                                                  ;
                          6E    56           04  C7 00166          DIVL3   #4, R6, ARGLIST                                         ;
                                               5E  DD 0016A          PUSHL   SP                                                      ; 0501
                                   00000000G   00  9F 0016C          PUSHAB  NSA$EVENT_AUDIT                                         ;
                       00000000G   00          02  FB 00172          CALLS   #2, SYS$CMKRNL                                          ;
                                               04  00179          RET                                                             ; 0503

; Routine Size:  378 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  511          0504  1 END
;  512          0505  0 ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $PLIT$                         152  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                         378  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                                 Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        75         0        1000        00:01.4
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:AUDIT/OBJ=OBJ$:AUDIT MSRC$:AUDIT/UPDATE=(ENH$:AUDIT)

; Size:          378 code + 152 data bytes
; Run Time:          00:09.9
; Elapsed Time:      00:38.2
; Lines/CPU Min:      3060
; Lexemes/CPU-Min:  26260
; Memory Used:   205 pages
; Compilation Complete
```

PPDDEF
MDL

FILEIO
LIS

LOGIN

DETACHED
LIS

LOGINOUT
MAP

UTILDEF
REQ

LOGINCMD
CLD

HPWD
LIS

LGIDEF
MDL

AUDIT
LIS

BREAKIN
LIS