```
UUU           UUU     AAAAAAAAA     FFFFFFFFFFFFFFF
UUU           UUU     AAAAAAAAA     FFFFFFFFFFFFFFF
UUU           UUU     AAAAAAAAA     FFFFFFFFFFFFFFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFFFFFFFFFFF
UUU           UUU   AAA         AAA   FFFFFFFFFFFF
UUU           UUU   AAA         AAA   FFFFFFFFFFFF
UUU           UUU   AAAAAAAAAAAAAAA   FFF
UUU           UUU   AAAAAAAAAAAAAAA   FFF
UUU           UUU   AAAAAAAAAAAAAAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUU           UUU   AAA         AAA   FFF
UUUUUUUUUUUUUUUUU   AAA         AAA   FFF
UUUUUUUUUUUUUUUUU   AAA         AAA   FFF
UUUUUUUUUUUUUUUUU   AAA         AAA   FFF
```

```
UU       UU    AAAAAA    FFFFFFFFFF  PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
UU       UU    AAAAAA    FFFFFFFFFF  PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
UU       UU   AA    AA   FF          PP      PP   AA    AA   RR      RR    SS        EE
UU       UU   AA    AA   FF          PP      PP   AA    AA   RR      RR    SS        EE
UU       UU   AA    AA   FF          PP      PP   AA    AA   RR      RR    SS        EE
UU       UU   AA    AA   FF          PP      PP   AA    AA   RR      RR    SS        EE
UU       UU   AA    AA   FFFFFFFF    PPPPPPPP     AA    AA   RRRRRRRR      SSSSSS    EEEEEEEE
UU       UU   AA    AA   FFFFFFFF    PPPPPPPP     AA    AA   RRRRRRRR      SSSSSS    EEEEEEEE
UU       UU   AAAAAAAAAA  FF          PP           AAAAAAAAAA  RR  RR              SS  EE
UU       UU   AAAAAAAAAA  FF          PP           AAAAAAAAAA  RR  RR              SS  EE
UU       UU   AA    AA   FF          PP           AA    AA    RR    RR            SS  EE
UU       UU   AA    AA   FF          PP           AA    AA    RR    RR            SS  EE
UUUUUUUUUU    AA    AA   FF          PP           AA    AA    RR      RR  SSSSSSSS    EEEEEEEEEE
UUUUUUUUUU    AA    AA   FF          PP           AA    AA    RR      RR  SSSSSSSS    EEEEEEEEEE

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 uafparse (
;     2          0002  0                 language (bliss32),
;001 ! JRL0038    0003  0                 ident = 'V04-001',
;     4-1        0004  0                 addressing_mode(external=general, nonexternal=general)) =
;     5          0005  1 begin
;     6          0006  1 !
;     7          0007  1 !*****************************************************************************
;     8          0008  1 !*                                                                           *
;     9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    11          0011  1 !*  ALL RIGHTS RESERVED.                                                     *
;    12          0012  1 !*                                                                           *
;    13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    14          0014  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
;    15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
;    16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;    17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
;    18          0018  1 !*  TRANSFERRED.                                                             *
;    19          0019  1 !*                                                                           *
;    20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
;    21          0021  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DICITAL EQUIPMENT         *
;    22          0022  1 !*  CORPORATION.                                                             *
;    23          0023  1 !*                                                                           *
;    24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS     *
;    25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*****************************************************************************
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:      System Management Utility Program
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:      Parsing subroutines for the AUTHORIZE utility
;    34          0034  1 !
;    35          0035  1 ! ENVIRONMENT:
;    36          0036  1 !
;    37          0037  1 ! AUTHOR:        Henry M. Levy    , CREATION DATE: 1-June-1977
;    38          0038  1 !
;    39          0039  1 ! MODIFIED BY:
;    40          0040  1 !
;001 ! JRL0038    0041  1 !       V04-001 JRL0038         John R. Lawson, Jr.     10-Oct-1984 09:04
;002 ! JRL0038    0042  1 !               Packet MHB0151 from FT2 has somehow disappeared, replace:
;003 ! JRL0038    0043  1 !
;004 ! JRL0038    0044  1 !               V03-016 MHB0151         Mark Bramhall           26-Apr-1984
;005 ! JRL0038    0045  1 !                       Fix external addressing mode.
;006 ! JRL0038    0046  1 !                       Change TPARSE tables PSECT naming to $TPA_xxx.
;007 ! JRL0038    0047  1 !                       Add disreconnect flag.
;008 ! JRL0038    0048  1 !                       Add security auditing for SYSUAF/NETUAF changes.
;009 ! JRL0038    0049  1 !
;010 ! JRL0038    0050  1 !               This packet was in FT2, but nothing later.
;011 ! JRL0038    0051  1 !
;    41          0052  1 !       V03-022 JRL0035         John R. Lawson, Jr.     07-Aug-1984 15:49
;    42          0053  1 !               Fix PWD_EXPIRED login flag to be set with passwords.
;    43          0054  1 !
;    44          0055  1 !       V03-021 JRL0022         John R. Lawson, Jr.     09-Jul-1984 13:40
;    45          0056  1 !               Supply missing brackets (if necessary) on /DIRECTORY.
;    46          0057  1 !
```

```
;  47          0058  1 ;       V03-020 JRL0011         John R. Lawson, Jr.     02-Jul-1984 14:40
;  48          0059  1 ;               Add literal WORD_UNSIGNED_LENGTH (=17) for unsigned
;  49          0060  1 ;               words in WS info.
;  50          0061  1 ;
;  51          0062  1 ;       V03-019 JRL0012         John R. Lawson, Jr.     21-Jun-1984 15:56
;  52          0063  1 ;               Add /GENERATE qualifier to generate random passwords.
;  53          0064  1 ;
;  54          0065  1 ;       V03-018 JRL0008         John R. Lawson, Jr.     20-Jun-1984 14:19
;  55          0066  1 ;               Add /PWDEXPIRED qualifier to allow pre-expired passwords.
;  56          0067  1 ;
;  57          0068  1 ;       V03-017 JRL0007         John R. Lawson, Jr.     20-Jun-1984 14:18
;  58          0069  1 ;               Add /PWDLIFETIME=NONE qualifier.
;  59          0070  1 ;
;  60          0071  1 ;       V03-016 JRL0002         John R. Lawson, Jr.     15-Jun-1984 12:54
;  61          0072  1 ;               Change all messages to calls to LIB$SIGNAL, place all
;  62          0073  1 ;               messages in a .MSG file
;  63          0074  1 ;
;  64          0075  1 ;       V03-015 LY0475          Larry Yetto             9-APR-1984 08:34
;  65          0076  1 ;               If /NOEXPIRATION or /NOPWDLIFE is specified then
;  66          0077  1 ;               zero the field in the UAF record.
;  67          0078  1 ;
;  68          0079  1 ;       V03-014 LY0467          Larry Yetto             22-MAR-1984 13:53
;  69          0080  1 ;               Add support for the rights data base functions
;  70          0081  1 ;
;  71          0082  1 ;       V03-013 ACG0397         Andrew C. Goldstein,    6-Feb-1984  16:25
;  72          0083  1 ;               Add DISREPORT flag to UAF; fix bugs in storing secondary
;  73          0084  1 ;               password
;  74          0085  1 ;
;  75          0086  1 ;       V03-012 ACG0388         Andrew C. Goldstein,    12-Jan-1984  18:52
;  76          0087  1 ;               Add command input to handle new UAF features;
;  77          0088  1 ;               general code cleanup
;  78          0089  1 ;
;  79          0090  1 ;       V03-011 ACG0385         Andrew C. Goldstein,    6-Jan-1984  14:18
;  80          0091  1 ;               V4 UAF format change
;  81          0092  1 ;
;  82          0093  1 ;       V03-010 TMK0001         Todd M. Katz            11-Oct-1983
;  83          0094  1 ;               Add JTQUOTA (job-wide logical name table creation quota)
;  84          0095  1 ;               qualifier.
;  85          0096  1 ;
;  86          0097  1 ;       V03-009 LMP0153         L. Mark Pilant,         13-Sep-1983  11:41
;  87          0098  1 ;               Add minimal support for alphanumeric UICs.
;  88          0099  1 ;
;  89          0100  1 ;       008     JWT0105         Jim Teague      04-Apr-1983
;  90          0101  1 ;               Finish up long filename stuff.
;  91          0102  1 ;
;  92          0103  1 ;       007     JWT0104         Jim Teague      29-Mar-1983
;  93          0104  1 ;               Add CLITABLES qualifier.
;  94          0105  1 ;
;  95          0106  1 ;       006     WMC0001         Wayne Cardoza   15-Mar-1983
;  96          0107  1 ;               Add MAXDETACH qualifier
;  97          0108  1 ;
;  98          0109  1 ;       005     JWT0076         Jim Teague      13-Dec-1982
;  99          0110  1 ;               Fix bug in parsing UIC specs.
; 100          0111  1 ;
; 101          0112  1 ;       004     JWT0046         Jim Teague      30-Jul-1982
; 102          0113  1 ;               Check null password string earlier in GETPASSWORD
; 103          0114  1 ;
```

```
;  104          0115  1 ;        003     JWT0030         Jim Teague      03-May-1982
;  105          0116  1 ;                Accomodate ufd directory specs
;  106          0117  1 ;
;  107          0118  1 ;        002     JWT0029         Jim Teague      20-Apr-1982
;  108          0119  1 ;                Completely disable /maxacctjobs
;  109          0120  1 ;
;  110          0121  1 ;        001     JWT0023         Jim Teague      17-Mar-1982
;  111          0122  1 ;                Enforce default directory syntax more strictly by
;  112          0123  1 ;                cranking the spec through tparse.
;  113          0124  1 ;
;  114          0125  1 ;--
```

```
;  116          0126  1 !
;  117          0127  1 ! Require files:
;  118          0128  1 !
;  119          0129  1 require
;  120          0130  1         'lib$:uafreq';
;  121          0226  1 !
;  122          0227  1 ! TABLE OF CONTENTS:
;  123          0228  1 !
;  124          0229  1
;  125          0230  1 forward routine
;  126          0231  1
;  127          0232  1         update_record,                  ! modify all specified fields
;  128          0233  1         getaccess,                      ! get access flag bits
;  129          0234  1         check_primary,                  ! verify "primary" keyword
;  130          0235  1         check_secondary,                ! verify "secondary" keyword
;  131          0236  1         set_range,                      ! set hourly restriction bits
;  132          0237  1         getcputim,                      ! get cpu time quota
;  133          0238  1         getdevice,                      ! get default device name
;  134          0239  1         getdirectory,                   ! get default directory
;  135          0240  1         checkvalue,                     ! check UFD group and member value
;  136          0241  1         checklength,                    ! check directory name length
;  137          0242  1         getflags,                       ! get flag bits
;  138          0243  1         getpflags,                      ! get old per day login flags
;  139          0244  1         getrestrict,                    ! get old hourly restriction values
;  140          0245  1         getprimedays,                   ! get primary day list
;  141          0246  1         getpriv,                        ! get process privileges
;  142          0247  1         getuic,                         ! get user identification code
;  143          0248  1         parse_uic,                      ! obtain UIC group and member
;  144          0249  1         parse_wild,                     ! parse wildcarded user specification
;  145          0250  1         parse_wild_uic,                 ! parse wildcarded UIC
;  146          0251  1         getstring,                      ! get string from input
;  147          0252  1         getval,                         ! get numeric value from user
;  148          0253  1         cvtnum,                         ! convert decimal ascii to binary
;  149          0254  1         UAF$GENERATE,                   ! generate necessary passwords
;  150          0255  1         GETPASSWORD,                    ! update password fields
;  151          0256  1         AUTHORIZE_GENERATE;             ! Interface to PL/I
;  152          0257  1
;  153          0258  1
;  154          0259  1 external routine
;  155          0260  1
;  156          0261  1         GENERATE_PASSWORDS,
;  157          0262  1         SYS$ASSIGN,
;  158          0263  1         SYS$DASSGN,
;  159          0264  1         SYS$QIOW,
;  160          0265  1         LIB$GET_VM,
;  161          0266  1         STR$UPCASE,
;001 !JRL0038  0267  1         lib$tparse,
;002 !JRL0038  0268  1         lib$lookup_key,
;  165-3        0269  1         cli$dcl_parse,
;  166          0270  1         cli$present,
;  167          0271  1         cli$get_value,
;  168          0272  1         faoout,                         ! output formatted message
;  169          0273  1         lgi$hpwd,                       ! hash password routine
;  170          0274  1         lib$cvt_dtime,                  ! convert ASCII to system delta time
;  171          0275  1         lib$cvt_time,                   ! convert ASCII to system abs time
;  172          0276  1         prv$setpriv;                    ! set privilege bits based on
;  173          0277  1                                         ! privilege names
```

```
;  174          0278  1
;  175          0279  1 !
;  176          0280  1 ! INCLUDE FILES:
;  177          0281  1 !
;  178          0282  1
;  179          0283  1 library 'SYS$LIBRARY:TPAMAC';
;  180          0284  1 library 'SYS$LIBRARY:LIB';
;  181          0285  1
;  182          0286  1 !
;  183          0287  1 ! MACROS:
;  184          0288  1 !
;  185          0289  1
;  186          0290  1 macro
;  187        M 0291  1     cstring[] = (uplit byte (%charcount (%string (%remaining)),
;  188          0292  1                  %string (%remaining)))%,
;  189          0293  1
;  190          0294  1 !
;  191          0295  1 ! Macro to create string descriptor for command parameters and
;  192          0296  1 !       qualifiers
;  193          0297  1 !
;  194        M 0298  1     sd[a] =
;  195          0299  1         bind %name ('SD_', a) = $descriptor (a)%;
;  196          0300  1
;  197          0301  1
;  198        P 0302  1      sd (
;  199        P 0303  1          'token1',       'token2',       'account',      'astlm',
;  200        P 0304  1          'biolm',        'bytlm',        'cli',          'cputime',
;  201        P 0305  1          'device',       'diolm',        'directory',    'enqlm',
;  202        P 0306  1          'fillm',        'flags',        'lgicmd',       'maxjobs',
;  203        P 0307  1          'owner',        'password',     'pbytlm',       'pflags',
;  204        P 0308  1          'pgflquota',    'p_restrict',   'prclm',        'priority',
;  205        P 0309  1          'privileges',   'sflags',       's_restrict',   'shrfillm',
;  206        P 0310  1          'tqelm',        'uic',          'wsdefault',    'wsextent',
;  207        P 0311  1          'wsquota',      'display',      'primedays',    'maxacctjobs',
;  208        P 0312  1          'maxdetach',    'clitables',    'jtquota',      'access',
;  209        P 0313  1          'batch',        'defprivileges','dialup',       'expiration',
;  210        P 0314  1          'interactive',  'local',        'network',      'pwdlifetime',
;  211        P 0315  1          'pwdminimum',   'quepriority',  'remote'
;  212          0316  1 );
;  213          0317  1
;  214          0318  1 !
;  215          0319  1 ! The definition table for flag bits for UAF$B_FLAGS.
;  216          0320  1 !
;  217          0321  1 literal
;001 !URL0038   0322  1         num_flags       = 14,           ! number of FLAG_TABLE entries
;  219-1        0323  1         num_opflags     = 2,            ! number of OPFLAG_TABLE entries
;  220          0324  1         num_pdflags     = 7;            ! number of entries in prime days table
;  221          0325  1
;  222          0326  1 own
;  223          0327  1     flag_table : vector [2 * num_flags + 1, long] initial (
;  224          0328  1         2 * num_flags,
;  225          0329  1         uplit (%asciz 'DISCTLY'),       $bitposition (uaf$v_disctly),
;  226          0330  1         uplit (%asciz 'DEFCLI'),        $bitposition (uaf$v_defcli),
;  227          0331  1         uplit (%asciz 'LOCKPWD'),       $bitposition (uaf$v_lockpwd),
;  228          0332  1         uplit (%asciz 'CAPTIVE'),       $bitposition (uaf$v_captive),
;  229          0333  1         uplit (%asciz 'DISUSER'),       $bitposition (uaf$v_disacnt),
;  230          0334  1         uplit (%asciz 'DISWELCOME'),    $bitposition (uaf$v_diswelcom),
```

```
;  231          0335  1          uplit (%ascic 'DISNEWMAIL'),    $bitposition (uaf$v_dismail),
;  232          0336  1          uplit (%ascic 'DISMAIL'),       $bitposition (uaf$v_nomail),
;  233          0337  1          uplit (%ascic 'GENPWD'),        $bitposition (uaf$v_genpwd),
;  234          0338  1          uplit (%ascic 'PWD_EXPIRED'),   $bitposition (uaf$v_pwd_expired),
;  235          0339  1          uplit (%ascic 'PWD2_EXPIRED'),  $bitposition (uaf$v_pwd2_expired),
;  236          0340  1          uplit (%ascic 'AUDIT'),         $bitposition (uaf$v_audit),
;001 !JRL0038   0341  1          uplit (%ascic 'DISREPORT'),     $bitposition (uaf$v_disreport),
;002 !JRL0038   0342  1          uplit (%ascic 'DISRECONNECT'),  $bitposition (uaf$v_disreconnect)
;  238-1        0343  1  ),
;  239          0344  1  !
;  240          0345  1  ! The definition table for flag bits for UAF$B_PDAYFLAGS and  UAF$B_SDAYFLAGS.
;  241          0346  1  !
;  242          0347  1      opflag_table : vector [2 * num_opflags + 1, long] initial (
;  243          0348  1          2 * num_opflags,
;  244          0349  1          uplit (%ascic 'DISDIALUP'),     uaf$k_disdialup,
;  245          0350  1          uplit (%ascic 'DISNETWORK'),    uaf$k_disnetwork        ! Disneyworld ??
;  246          0351  1  ),
;  247          0352  1  !
;  248          0353  1  ! The definition table for flag bits for UAF$B_PRIMEDAYS.
;  249          0354  1  !
;  250          0355  1      pdflag_table : vector [2 * num_pdflags + 1, long] initial (
;  251          0356  1          2 * num_pdflags,
;  252          0357  1          uplit (%ascic 'MONDAY'),        $bitposition (uaf$v_monday),
;  253          0358  1          uplit (%ascic 'TUESDAY'),       $bitposition (uaf$v_tuesday),
;  254          0359  1          uplit (%ascic 'WEDNESDAY'),     $bitposition (uaf$v_wednesday),
;  255          0360  1          uplit (%ascic 'THURSDAY'),      $bitposition (uaf$v_thursday),
;  256          0361  1          uplit (%ascic 'FRIDAY'),        $bitposition (uaf$v_friday),
;  257          0362  1          uplit (%ascic 'SATURDAY'),      $bitposition (uaf$v_saturday),
;  258          0363  1          uplit (%ascic 'SUNDAY'),        $bitposition (uaf$v_sunday)
;  259          0364  1  );
;  260          0365  1
;  261          0366  1
;  262          0367  1  !
;  263          0368  1  ! EQUATED SYMBOLS:
;  264          0369  1  !
;  265          0370  1
;  266          0371  1  literal
;  267          0372  1          false           = 0,            ! logical false
;  268          0373  1          true            = 1,            ! logical true
;  269          0374  1
;  270          0375  1          network_access  = 1^0,          ! mask representing /NETWORK_ACCESS
;  271          0376  1          batch_access    = 1^1,          ! mask representing /BATCH_ACCESS
;  272          0377  1          local_access    = 1^2,          ! mask representing /LOCAL_ACCESS
;  273          0378  1          dialup_access   = 1^3,          ! mask representing /DIALUP_ACCESS
;  274          0379  1          remote_access   = 1^4,          ! mask representing /REMOTE_ACCESS
;  275          0380  1          interactive_access =            ! mask representing /INTERACTIVE_ACCESS
;  276          0381  1                          local_access or dialup_access or remote_access,
;  277          0382  1          access_access   =               ! mask representing /ACCESS
;  278          0383  1                          interactive_access or network_access or batch_access,
;  279          0384  1
;  280          0385  1          byte_length     = 8,            ! bits per byte
;  281          0386  1          word_length     = 16,           ! bits per word
;  282          0387  1          word_unsigned_length
;  283          0388  1                          = 17,           ! for WS info
;  284          0389  1          long_length     = 32,           ! bits per longword
;  285          0390  1
;  286          0391  1
```

```
;  287     0392  1          blank               = ' ',              !  delimiters
;  288     0393  1          colon               = ':',
;  289     0394  1          comma               = ',',
;  290     0395  1          cr                  = 13,
;  291     0396  1          equals              = '=',
;  292     0397  1          lf                  = 10,
;  293     0398  1          lparen              = '(',
;  294     0399  1          rparen              = ')',
;  295     0400  1          slash               = '/',
;  296     0401  1          tab                 = '    ',
;  297     0402  1          zero                = 0,
;  298     0403  1
;  299     0404  1          counted_string      = 1,            ! counted string type indicator
;  300     0405  1          filled_string       = 2;            ! filled string type indicator
;  301     0406  1
;  302     0407  1  own
;  303     0408  1          status,
;  304     0409  1          cli_status,
;  305     0410  1
;  306     0411  1          tparse_block        : block [tpa$c_length0, byte]
;  307     0412  1                                preset ([tpa$l_count]   = tpa$k_count0,
;  308     0413  1                                        [tpa$l_options] = tpa$m_abbrev),
;  309     0414  1          converted_uic,
;  310     0415  1          address,
;  311     0416  1          left_bit,
;  312     0417  1          right_bit,
;  313     0418  1          primary_access      : bitvector [32],
;  314     0419  1          secondary_access    : bitvector [32],
;  315     0420  1          nc_flag             : initial (false),
;  316     0421  1          primary             : initial (false),
;  317     0422  1          secondary           : initial (false);
;  318     0423  1
;  319     0424  1  !
;  320     0425  1  ! ASCII numbers from '0' through hex 'F'
;  321     0426  1  !
;  322     0427  1
;  323     0428  1  bind
;  324     0429  1          numbers             = uplit byte ('0123456789ABCDEF'),
;  325     0430  1
;  326     0431  1  !
;  327     0432  1  ! 'NO' prefix descriptor for negating privileges
;  328     0433  1  !
;  329     0434  1          no_dsc              = $descriptor ('NO') : vector;
;  330     0435  1
;  331     0436  1  !
;  332     0437  1  !       External messages
;  333     0438  1  !
;  334     0439  1
;  335     0440  1  external routine
;  336     0441  1
;  337     0442  1     LIB$SICVAL: addressing_mode(general);
;  338     0443  1
;  339     0444  1  external literal
;  340     0445  1
;  341     0446  1     UAF$_BADVALUE,      UAF$_EXTRAPARM,      UAF$_INVDEV,
;  342     0447  1     UAF$_INVSTR,        UAF$_INVUSER_PEC,    UAF$_KEYNOTFND,
;  343     0448  1     UAF$_NAMETOOBIG,    UAF$_NOARG,          UAF$_NOQUOTE,
```

```
;  344          0449  1    UAF$_NOUSERSPEC,     UAF$_PRVNOTFND,          UAF$_PRVNOTUNQ,
;  345          0450  1    UAF$_PWDSYNTAX,      UAF$_PWDNCH,             UAF$_PWDNOL,
;  346          0451  1    UAF$_UICERR,         UAF$_VALTOOBIG,          UAF$_ZCONFLICT;
;  347          0452  1
;  348          0453  1
;  349          0454  1 !
;  350          0455  1 ! EXTERNAL REFERENCES:
;  351          0456  1 !
;  352          0457  1
;  353          0458  1
;  354          0459  1 external literal
;  355          0460  1
;  356          0461  1         encrypt,                        ! encryption algorithm to use
;  357          0462  1
;  358          0463  1         lib$_ambkey,                    ! status for ambiguous keyword
;  359          0464  1         lib$_unrkey,                    ! status for unrecognized keyword
;  360          0465  1
;  361          0466  1         cli$_abkeyw,                    ! message symbol for "ambiguous keyword"
;  362          0467  1         cli$_ivkeyw,                    ! message symbol for "unrecognized keyword"
;  363          0468  1         cli$_negated,                   ! qualifier explicitly negated
;  364          0469  1         CLI$_ABSENT,                    ! Qualifier not given
;  365          0470  1         prv$_notunq,
;  366          0471  1         prv$_invnam;
;  367          0472  1
;  368          0473  1 external                                ! Global storage from UAFMAIN
:001 !JRL0038  0474  1         uaf$gq_sysuaff  : block [,byte],        ! SYSUAF auditing flags
;  369          0475  1         uaf$gl_ctlmsk   : block,
;  370          0476  1         authorize_commands : addressing_mode ( general ),
;  371          0477  1         cmdlindsc       : block [,byte],
;  372          0478  1         call_count,
;  373          0479  1         symbol_str,                     ! Symbolic character set
;  374          0480  1         match_tokenlen,
;  375          0481  1         match_token,
;  376          0482  1         tokendsc        : block [,byte],
;  377          0483  1         tokenlen        : word,         ! Bound to TOKENDSC[DSC$W_LENGTH]
;  378          0484  1         tokenptr,                       ! Bound to TOKENDSC[DSC$A_POINTER]
;  379          0485  1         uic_flag,                       ! Wildcard parsing flags
;  380          0486  1         grp_wild,
;  381          0487  1         mem_wild,
;  382          0488  1         str_wild,
;  383          0489  1
;  384          0490  1         time_buf        : block [,byte], ! Buffer to receive current system time
;  385          0491  1         pwd_flag,                       ! User did not supply password flag
;  386          0492  1         recbuf          : block [,byte]; ! The current UAF record
;  387          0493  1
;  388          0494  1 !
;  389          0495  1 ! TPARSE state table to parse /<access> qualifiers
;  390          0496  1 !
;  391          0497  1
:001 !JRL0038  0498  1 $init_state (access_states, access_keys, $tpa);
;  393-1        0499  1
;  394        P 0500  1 $state   (
;  395        P 0501  1          ('PRIMARY', terminal, check_primary, true, primary),
;  396        P 0502  1          ('SECONDARY', terminal, check_secondary, true, secondary),
;  397        P 0503  1          (tpa$_decimal,,,, right_bit)
;  398          0504  1         );
;  399          0505  1
```

```
;  400          P 0506  1 $state   (,
;  401          P 0507  1           ('_'),
;  402          P 0508  1           (tpa$_eos, tpa$_exit, set_range)
;  403            0509  1          );
;  404            0510  1
;  405          P 0511  1 $state   (,
;  406          P 0512  1           (tpa$_decimal,, set_range)
;  407            0513  1          );
;  408            0514  1
;  409          P 0515  1 $state   (terminal,
;  410          P 0516  1           (tpa$_eos, tpa$_exit)
;  411            0517  1          );
;  412            0518  1
;  413            0519  1 !
;  414            0520  1 ! TPARSE state table for old /P_RESTRICT and /S_RESTRICT qualifiers.
;  415            0521  1 !
;  416            0522  1
;001 !JRL0038     0523  1 $init_state (restrict_states, restrict_keys, $tpa);
;  418-1          0524  1
;  419          P 0525  1 $state   (,
;  420          P 0526  1           (tpa$_decimal,,,, right_bit)
;  421            0527  1          );
;  422            0528  1
;  423          P 0529  1 $state   (,
;  424          P 0530  1           ('_'),
;  425          P 0531  1           (tpa$_eos, tpa$_exit, set_range)
;  426            0532  1          );
;  427            0533  1
;  428          P 0534  1 $state   (,
;  429          P 0535  1           (tpa$_decimal,, set_range)
;  430            0536  1          );
;  431            0537  1
;  432          P 0538  1 $state   (,
;  433          P 0539  1           (tpa$_eos, tpa$_exit)
;  434            0540  1          );
;  435            0541  1
;  436            0542  1 !
;  437            0543  1 ! TPARSE state table for UIC spec
;  438            0544  1 !
;  439            0545  1
;001 !JRL0038     0546  1 $init_state (uic_states, uic_keys, $tpa);
;  441-1          0547  1
;  442          P 0548  1 $state   (,
;  443          P 0549  1           (tpa$_ident, tpa$_exit,,, converted_uic)
;  444            0550  1          );
;  445            0551  1
;  446            0552  1 !
;  447            0553  1 ! TPARSE state table for directory string
;  448            0554  1 !
;  449            0555  1
;001 !JRL0038     0556  1 $init_state (dir_states, dir_keys, $tpa);
;  451-1          0557  1
;  452          P 0558  1 $state   (,
;  453          P 0559  1           ( (ufd), end_state),
;  454          P 0560  1           ( (subdir), end_state),
;  455          P 0561  1           ('['),
;  456          P 0562  1           ('<')
```

```
;  457          0563  1          );
;  458          0564  1
;  459        P 0565  1 $state    (,
;  460        P 0566  1           ( (ufd)),
;  461        P 0567  1           ( (subdir))
;  462          0568  1          );
;  463          0569  1
;  464        P 0570  1 $state    (,
;  465        P 0571  1           (']'),
;  466        P 0572  1           ('>')
;  467          0573  1          );
;  468          0574  1
;  469        P 0575  1 $state    (end_state,
;  470        P 0576  1           (tpa$_eos, tpa$_exit)
;  471          0577  1          );
;  472          0578  1
;  473        P 0579  1 $state    (ufd,
;  474        P 0580  1           (tpa$_octal,, checkvalue)
;  475          0581  1          );
;  476          0582  1
;  477        P 0583  1 $state    (,
;  478        P 0584  1           (',')
;  479          0585  1          );
;  480          0586  1
;  481        P 0587  1 $state    (,
;  482        P 0588  1           (tpa$_octal, tpa$_exit, checkvalue)
;  483          0589  1          );
;  484          0590  1
;  485        P 0591  1 $state    (subdir,
;  486        P 0592  1           (tpa$_symbol,, checklength)
;  487          0593  1          );
;  488          0594  1
;  489        P 0595  1 $state    (,
;  490        P 0596  1           ('.',subdir),
;  491        P 0597  1           (tpa$_lambda,tpa$_exit)
;  492          0598  1          );
```

```
;  494          0599  1 %sbttl 'update_record - modify all specified fields'
;  495          0600  1 global routine update_record =
;  496          0601  2 begin
;  497          0602  2
;  498          0603  2 !++
;  499          0604  2 !
;  500          0605  2 ! FUNCTIONAL DESCRIPTION:
;  501          0606  2 !
;  502          0607  2 !       Scan the remainder of the user's input command line for parameters
;  503          0608  2 !       and parameter values.  Pull off each parameter keyword and check
;  504          0609  2 !       that it is valid, then call the associated routine to process
;  505          0610  2 !       the supplied value or string.
;  506          0611  2 !
;  507          0612  2 ! INPUTS:
;  508          0613  2 !
;  509          0614  2 !       none
;  510          0615  2 !
;  511          0616  2 ! IMPLICIT INPUTS:
;  512          0617  2 !
;  513          0618  2 !
;  514          0619  2 ! OUTPUTS:
;  515          0620  2 !
;  516          0621  2 !       none
;  517          0622  2 !
;  518          0623  2 ! IMPLICIT OUTPUTS:
;  519          0624  2 !
;  520          0625  2 !       none
;  521          0626  2 !
;  522          0627  2 ! ROUTINE VALUE:
;  523          0628  2 !
;  524          0629  2 !       true -> all keywords and parameters were specified correctly
;  525          0630  2 !
;  526          0631  2 ! SIDE EFFECTS:
;  527          0632  2 !
;  528          0633  2 !       none
;  529          0634  2 !--
;  530          0635  2 status = false;
;  531          0636  2 primary = false;
;  532          0637  2 secondary = false;
;  533          0638  2 no_flag = false;
;  534          0639  2
;001  !JRL0038   0640  2 uaf$gq_sysuaff[0,0,32,0] = 0;
;002  !JRL0038   0641  2 uaf$gq_sysuaff[4,0,32,0] = 0;
;003  !JRL0038   0642  2 $assume(nsa_s_sysuaff, eql, 8);
;004  !JRL0038   0643  2
;  535          0644  2 !
;  536          0645  2 ! If this is part of a series of calls from WILD_USER, the command
;  537          0646  2 !    line must be reparsed after the first call
;  538          0647  2 !
;  539          0648  2 if .call_count neq 0
;  540          0649  2 then
;  541          0650  2     cli$dcl_parse (cmdlindsc, authorize_commands);
;  542          0651  2
;  543          0652  2
;  544          0653  2 if CLI$PRESENT(%ascid'GENERATE_PASSWORD')
;  545          0654  2 and (CLI$PRESENT(SD_PASSWORD) neq CLI$_ABSENT) then
;  546          0655  3    begin
```

```
;  547          0656  3
;  548          0657  3        LIB$SIGNAL(UAF$_ZCONFLICT);
;  549          0658  3
   550          0659  3    end
;  551          0660  2 else if CLI$PRESENT(%ascid'GENERATE_PASSWORD')
;  552          0661  2 or (CLI$PRESENT(SD_PASSWORD) neq CLI$_ABSENT) then
;  553          0662  3    begin
;  554          0663  3
;001 !JRL0038   0664  3        uaf$gq_sysuaff[nsa_v_password] = true;
;002 !JRL0038   0665  3
;  555          0666  3        if CLI$PRESENT(%ascid'GENERATE_PASSWORD') then
;  556          0667  3            UAF$GENERATE();
;  557          0668  3        if not GETPASSWORD() then
;  558          0669  3            return FALSE;
;  559          0670  3        RECBUF[UAF$V_PWD_EXPIRED] = FALSE;
;  560          0671  3        STATUS = TRUE;
;  561          0672  3
;  562          0673  2    end;
;  563          0674  2
;  564          0675  2
;  565          0676  2 if .uaf$gl_ctlmsk[uaf$v_rename]
;  566          0677  2 then
;  567          0678  2     return true;
;  568          0679  2
;  569          0680  2 !
;  570          0681  2 ! Check that we're not at the end of the command line. Continue
;  571          0682  2 ! processing parameters until the end is reached.
;  572          0683  2 !
;  573          0684  2
;  574          0685  2 cli_status = cli$present (sd_access);
;  575          0686  2 if .cli_status
;  576          0687  2 or .cli_status eql cli$_negated
;  577          0688  2 then
;  578          0689  3     begin
;001 !JRL0038   0690  3     uaf$gq_sysuaff[nsa_v_access] = true;
;  579          0691  3     if not getaccess (sd_access, access_access, .cli_status)
;  580          0692  3     then return false;
;  581          0693  3     status = true;
;  582          0694  2     end;
;  583          0695  2
;  584          0696  2 cli_status = cli$present (sd_interactive);
;  585          0697  2 if .cli_status
;  586          0698  2 or .cli_status eql cli$_negated
;  587          0699  2 then
;  588          0700  3     begin
;001 !JRL0038   0701  3     uaf$gq_sysuaff[nsa_v_interactive] = true;
;  589          0702  3     if not getaccess (sd_interactive, interactive_access, .cli_status)
;  590          0703  3     then return false;
;  591          0704  3     status = true;
;  592          0705  2     end;
;  593          0706  2
;  594          0707  2 cli_status = cli$present (sd_batch);
;  595          0708  2 if .cli_status
;  596          0709  2 or .cli_status eql cli$_negated
;  597          0710  2 then
;  598          0711  3     begin
;001 !JRL0038   0712  3     uaf$gq_sysuaff[nsa_v_batch] = true;
```

```
;  599          0713  3     if not getaccess (sd_batch, batch_access, .cli_status)
;  600          0714  3     then return false;
;  601          0715  3     status = true;
;  602          0716  2     end;
;  603          0717  2
;  604          0718  2 cli_status = cli$present (sd_dialup);
;  605          0719  2 if .cli_status
;  606          0720  2 or .cli_status eql cli$_negated
;  607          0721  2 then
;  608          0722  3     begin
;001 !JRL0038   0723  3     uaf$gq_sysuaff[nsa_v_dialup] = true;
;  609          0724  3     if not getaccess (sd_dialup, dialup_access, .cli_status)
;  610          0725  3     then return false;
;  611          0726  3     status = true;
;  612          0727  2     end;
;  613          0728  2
;  614          0729  2 cli_status = cli$present (sd_network);
;  615          0730  2 if .cli_status
;  616          0731  2 or .cli_status eql cli$_negated
;  617          0732  2 then
;  618          0733  3     begin
;001 !JRL0038   0734  3     uaf$gq_sysuaff[nsa_v_network] = true;
;  619          0735  3     if not getaccess (sd_network, network_access, .cli_status)
;  620          0736  3     then return false;
;  621          0737  3     status = true;
;  622          0738  2     end;
;  623          0739  2
;  624          0740  2 cli_status = cli$present (sd_local);
;  625          0741  2 if .cli_status
;  626          0742  2 or .cli_status eql cli$_negated
;  627          0743  2 then
;  628          0744  3     begin
;001 !JRL0038   0745  3     uaf$gq_sysuaff[nsa_v_local] = true;
;  629          0746  3     if not getaccess (sd_local, local_access, .cli_status)
;  630          0747  3     then return false;
;  631          0748  3     status = true;
;  632          0749  2     end;
;  633          0750  2
;  634          0751  2 cli_status = cli$present (sd_remote);
;  635          0752  2 if .cli_status
;  636          0753  2 or .cli_status eql cli$_negated
;  637          0754  2 then
;  638          0755  3     begin
;001 !JRL0038   0756  3     uaf$gq_sysuaff[nsa_v_remote] = true;
;  639          0757  3     if not getaccess (sd_remote, remote_access, .cli_status)
;  640          0758  3     then return false;
;  641          0759  3     status = true;
;  642          0760  2     end;
;  643          0761  2
;  644          0762  2 if cli$present (sd_account)
;  645          0763  2 then
;  646          0764  3     begin
;001 !JRL0038   0765  3     uaf$gq_sysuaff[nsa_v_account] = true;
;  647          0766  3     cli$get_value (sd_account, tokendsc);
;  648          0767  3 !***     if not getstring (recbuf[uaf$t_account], uaf$s_account, filled_string)
;  649          0768  3     if not getstring (recbuf[uaf$t_account], 8, filled_string)
;  650          0769  3     then return false;
```

```
;  651          0770  3      status = true;
;  652          0771  2      end;
;  653          0772  2
;  654          0773  2  if cli$present (sd_astlm)
;  655          0774  2  then
;  656          0775  3      begin
;001 !JRL0038   0776  3      uaf$gq_sysuaff[nsa_v_astlm] = true;
;  657          0777  3      cli$get_value (sd_astlm, tokendsc);
;  658          0778  3      if not getval (recbuf[uaf$w_astlm], word_length)
;  659          0779  3      then return false;
;  660          0780  3      status = true;
;  661          0781  2      end;
;  662          0782  2
;  663          0783  2  if cli$present (sd_biolm)
;  664          0784  2  then
;  665          0785  3      begin
;001 !JRL0038   0786  3      uaf$gq_sysuaff[nsa_v_biolm] = true;
;  666          0787  3      cli$get_value (sd_biolm, tokendsc);
;  667          0788  3      if not getval (recbuf[uaf$w_biolm], word_length)
;  668          0789  3      then return false;
;  669          0790  3      status = true;
;  670          0791  2      end;
;  671          0792  2
;  672          0793  2  if cli$present (sd_bytlm)
;  673          0794  2  then
;  674          0795  3      begin
;001 !JRL0038   0796  3      uaf$gq_sysuaff[nsa_v_bytlm] = true;
;  675          0797  3      cli$get_value (sd_bytlm, tokendsc);
;  676          0798  3      if not getval (recbuf[uaf$l_bytlm], long_length)
;  677          0799  3      then return false;
;  678          0800  3      status = true;
;  679          0801  2      end;
;  680          0802  2
;  681          0803  2  if cli$present (sd_cli)
;  682          0804  2  then
;  683          0805  3      begin
;001 !JRL0038   0806  3      uaf$gq_sysuaff[nsa_v_cli] = true;
;  684          0807  3      cli$get_value (sd_cli, tokendsc);
;  685          0808  3      if not getstring (recbuf[uaf$t_defcli], uaf$s_defcli, counted_string)
;  686          0809  3      then return false;
;  687          0810  3      uaf$gl_ctlmsk[uaf$v_clitab_pres] =
;  688          0811  3          (.vector [recbuf[uaf$t_clitables], 0; ,byte] neq 0);
;  689           812  3      uaf$gl_ctlmsk[uaf$v_cli] = true;
;  690          0813  3      status = true;
;  691          0814  2      end;
;  692          0815  2
;  693          0816  2  if cli$present (sd_clitables)
;  694          0817  2  then
;  695          0818  3      begin
;001 !JRL0038   0819  3      uaf$gq_sysuaff[nsa_v_clitables] = true;
;  696          0820  3      cli$get_value (sd_clitables, tokendsc);
;  697          0821  3      if not getstring (recbuf[uaf$t_clitables], uaf$s_clitables, counted_string)
;  698          0822  3      then return false;
;  699          0823  3      uaf$gl_ctlmsk[uaf$v_clitables] = true;
;  700          0824  3      status = true;
;  701          0825  2      end;
;  702          0826  2
```

```
;  703          0827  2 if cli$present (sd_cputime)
;  704          0828  2 then
;  705          0829  3     begin
;001 !JRL0038   0830  3     uaf$gq_sysuaff[nsa_v_cputime] = true;
;  706          0831  3     cli$get_value (sd_cputime, tokendsc);
;  707          0832  3     if not getcputim ()
;  708          0833  3     then return false;
;  709          0834  3     status = true;
;  710          0835  2     end;
;  711          0836  2
;  712          0837  2 if cli$present (sd_defprivileges)
;  713          0838  2 then
;  714          0839  3     begin
;001 !JRL0038   0840  3     uaf$gq_sysuaff[nsa_v_defprivileges] = true;
;  715          0841  3     if not getpriv (sd_defprivileges, recbuf[uaf$q_def_priv])
;  716          0842  3     then return false;
;  717          0843  3     status = true;
;  718          0844  2     end;
;  719          0845  2
;  720          0846  2 if cli$present (sd_device)
;  721          0847  2 then
;  722          0848  3     begin
;001 !JRL0038   0849  3     uaf$gq_sysuaff[nsa_v_device] = true;
;  723          0850  3     cli$get_value (sd_device, tokendsc);
;  724          0851  3     if not getdevice ()
;  725          0852  3     then return false;
;  726          0853  3     status = true;
;  727          0854  2     end;
;  728          0855  2
;  729          0856  2 if cli$present (sd_diolm)
;  730          0857  2 then
;  731          0858  3     begin
;001 !JRL0038   0859  3     uaf$gq_sysuaff[nsa_v_diolm] = true;
;  732          0860  3     cli$get_value (sd_diolm, tokendsc);
;  733          0861  3     if not getval (recbuf[uaf$w_diolm], word_length)
;  734          0862  3     then return false,
;  735          0863  3     status = true;
;  736          0864  2     end;
;  737          0865  2
;  738          0866  2 if cli$present (sd_directory)
;  739          0867  2 then
;  740          0868  3     begin
;001 !JRL0038   0869  3     uaf$gq_sysuaff[nsa_v_directory] = true;
;  741          0870  3     cli$get_value (sd_directory, tokendsc);
;  742          0871  3     if not getdirectory ()
;  743          0872  3     then return false;
;  744          0873  3     status = true;
;  745          0874  2     end;
;  746          0875  2
;  747          0876  2 if cli$present (sd_enqlm)
;  748          0877  2 then
;  749          0878  3     begin
;001 !JRL0038   0879  3     uaf$gq_sysuaff[nsa_v_enqlm] = true;
;  750          0880  3     cli$get_value (sd_enqlm, tokendsc);
;  751          0881  3     if not getval (recbuf[uaf$w_enqlm], word_length)
;  752          0882  3     then return false;
;  753          0883  3     status = true;
```

```
;  754          0884  2     end;
;  755          0885  2
;  756          0886  2 cli_status = cli$present (sd_expiration) ;
;  757          0887  2 if .cli_status eql cli$_negated
;  758          0888  2 then
;  759          0889  3     begin
;001 !JRL0038   0890  3     uaf$gq_sysuaff[nsa_v_expiration] = true;
;  760          0891  3     ch$fill ( 0, uaf$s_expiration, recbuf[uaf$q_expiration] ) ;
;  761          0892  3     status = true ;
;  762          0893  3     end
;  763          0894  2 else
;  764          0895  2     if .cli_status
;  765          0896  2     then
;  766          0897  3         begin
;001 !JRL0038   0898  3         uaf$gq_sysuaff[nsa_v_expiration] = true;
;  767          0899  3         cli$get_value (sd_expiration, tokendsc);
;  768          0900  4         if not (status = lib$cvt_time (tokendsc, recbuf[uaf$q_expiration]))
;  769          0901  3         then
;  770          0902  4             begin
;  771          0903  4             signal (.status);
;  772          0904  4             return false;
;  773          0905  3             end;
;  774          0906  3         status = true;
;  775          0907  2         end;
;  776          0908  2
;  777          0909  2 if cli$present (sd_fillm)
;  778          0910  2 then
;  779          0911  3     begin
;001 !JRL0038   0912  3     uaf$gq_sysuaff[nsa_v_fillm] = true;
;  780          0913  3     cli$get_value (sd_fillm, tokendsc);
;  781          0914  3     if not getval (recbuf[uaf$w_fillm], word_length)
;  782          0915  3     then return false;
;  783          0916  3     status = true;
;  784          0917  2     end;
;  785          0918  2
;  786          0919  2 if cli$present (sd_flags)
;  787          0920  2 then
;  788          0921  3     begin
;001 !JRL0038   0922  3     uaf$gq_sysuaff[nsa_v_flags] = true;
;  789          0923  3     if not getflags ()
;  790          0924  3     then return false;
;  791          0925  3     status = true;
;  792          0926  2     end;
;  793          0927  2
;  794          0928  2 if cli$present (sd_jtquota)
;  795          0929  2 then
;  796          0930  3     begin
;001 !JRL0038   0931  3     uaf$gq_sysuaff[nsa_v_jtquota] = true;
;  797          0932  3     cli$get_value (sd_jtquota, tokendsc);
;  798          0933  3     if not getval (recbuf[uaf$l_jtquota], long_length)
;  799          0934  3     then return false;
;  800          0935  3     status = true;
;  801          0936  2     end;
;  802          0937  2
;  803          0938  2 if cli$present (sd_lgicmd)
;  804          0939  2 then
;  805          0940  3     begin
```

```
:001 !JRL0038    0941  3      uaf$gq_sysuaff[nsa_v_lgicmd] = true;
:   806          0942  3      cli$get_value (sd_lgicmd, tokendsc);
:   807          0943  3      if not getstring (recbuf[uaf$t_lgicmd], uaf$s_lgicmd, counted_string)
:   808          0944  3      then return false;
:   809          0945  3      status = true;
:   810          0946  2      end;
:   811          0947  2
:   812          0948  2  if cli$present (sd_maxjobs)
:   813          0949  2  then
:   814          0950  3      begin
:001 !JRL0038    0951  3      uaf$gq_sysuaff[nsa_v_maxjobs] = true;
:   815          0952  3      cli$get_value (sd_maxjobs, tokendsc);
:   816          0953  3      if not getval (recbuf[uaf$w_maxjobs], word_length)
:   817          0954  3      then return false;
:   818          0955  3      status = true;
:   819          0956  2      end;
:   820          0957  2
:   821          0958  2  if cli$present (sd_maxacctjobs)
:   822          0959  2  then
:   823          0960  3      begin
:001 !JRL0038    0961  3      uaf$gq_sysuaff[nsa_v_maxacctjobs] = true;
:   824          0962  3      cli$get_value (sd_maxacctjobs, tokendsc);
:   825          0963  3      if not getval (recbuf[uaf$w_maxacctjobs], word_length)
:   826          0964  3      then return false;
:   827          0965  3      status = true;
:   828          0966  2      end;
:   829          0967  2
:   830          0968  2  if cli$present (sd_maxdetach)
:   831          0969  2  then
:   832          0970  3      begin
:001 !JRL0038    0971  3      uaf$gq_sysuaff[nsa_v_maxdetach] = true;
:   833          0972  3      cli$get_value (sd_maxdetach, tokendsc);
:   834          0973  3      if not getval (recbuf[uaf$w_maxdetach], word_length)
:   835          0974  3      then return false;
:   836          0975  3      status = true;
:   837          0976  2      end;
:   838          0977  2
:   839          0978  2  if cli$present (sd_owner)
:   840          0979  2  then
:   841          0980  3      begin
:001 !JRL0038    0981  3      uaf$gq_sysuaff[nsa_v_owner] = true;
:   842          0982  3      cli$get_value (sd_owner, tokendsc);
:   843          0983  3      if not getstring (recbuf[uaf$t_owner], uaf$s_owner, counted_string)
:   844          0984  3      then return false;
:   845          0985  3      status = true;
:   846          0986  2      end;
:   847          0987  2
:   848          0988  2  if cli$present (sd_pbytlm)
:   849          0989  2  then
:   850          0990  3      begin
:001 !JRL0038    0991  3      uaf$gq_sysuaff[nsa_v_pbytlm] = true;
:   851          0992  3      cli$get_value (sd_pbytlm, tokendsc);
:   852          0993  3      if not getval (recbuf[uaf$l_pbytlm], long_length)
:   853          0994  3      then return false;
:   854          0995  3      status = true;
:   855          0996  2      end;
:   856          0997  2
```

```
;  857          0998  2 if cli$present (sd_pgflquota)
;  858          0999  2 then
;  859          1000  3     begin
;001 JRL0038    1001  3     uaf$gq_sysuaff[nsa_v_pgflquota] = true;
;  860          1002  3     cli$get_value (sd_pgflquota, tokendsc);
;  861          1003  3     if not getval (recbuf[uaf$l_pgflquota], 3 * byte_length)
;  862          1004  3     then return false;
;  863          1005  3     status = true;
;  864          1006  2     end;
;  865          1007  2
;  866          1008  2 if cli$present (sd_p_restrict)
;  867          1009  2 then
;  868          1010  3     begin
;001 !JRL0038   1011  3     uaf$gq_sysuaff[nsa_v_p_restrict] = true;
;  869          1012  3     if not getrestrict (sd_p_restrict)
;  870          1013  3     then return false;
;  871          1014  3     status = true;
;  872          1015  2     end;
;  873          1016  2
;  874          1017  2 if cli$present (sd_s_restrict)
;  875          1018  2 then
;  876          1019  3     begin
;001 !JRL0038   1020  3     uaf$gq_sysuaff[nsa_v_s_restrict] = true;
;  877          1021  3     secondary = true;
;  878          1022  3     if not getrestrict (sd_s_restrict)
;  879          1023  3     then return false;
;  880          1024  3     status = true;
;  881          1025  2     end;
;  882          1026  2
;  883          1027  2 if cli$present (sd_pflags)
;  884          1028  2 then
;  885          1029  3     begin
;001 !JRL0038   1030  3     uaf$gq_sysuaff[nsa_v_pflags] = true;
;  886          1031  3     primary = true;
;  887          1032  3     if not getpflags (sd_pflags)
;  888          1033  3     then return false;
;  889          1034  3     status = true;
;  890          1035  2     end;
;  891          1036  2
;  892          1037  2 if cli$present (sd_sflags)
;  893          1038  2 then
;  894          1039  3     begin
;001 !JRL0038   1040  3     uaf$gq_sysuaff[nsa_v_sflags] = true;
;  895          1041  3     if not getpflags (sd_sflags)
;  896          1042  3     then return false;
;  897          1043  3     status = true;
;  898          1044  2     end;
;  899          1045  2
;  900          1046  2 if cli$present (sd_prclm)
;  901          1047  2 then
;  902          1048  3     begin
;001 !JRL0038   1049  3     uaf$gq_sysuaff[nsa_v_prclm] = true;
;  903          1050  3     cli$get_value (sd_prclm, tokendsc);
;  904          1051  3     if not getval (recbuf[uaf$w_prccnt], word_length)
;  905          1052  3     then return false;
;  906          1053  3     status = true;
;  907          1054  2     end;
```

```
;  908          1055  2
;  909          1056  2 if cli$present (sd_primedays)
;  910          1057  2 then
;  911          1058  3     begin
;001 !JRL0038   1059  3     uaf$gq_sysuaff[nsa_v_primedays] = true;
;  912          1060  3     if not getprimedays ()
;  913          1061  3     then return false;
;  914          1062  3     status = true;
;  915          1063  2     end;
;  916          1064  2
;  917          1065  2 if cli$present (sd_priority)
;  918          1066  2 then
;  919          1067  3     begin
;001 !JRL0038   1068  3     uaf$gq_sysuaff[nsa_v_priority] = true;
;  920          1069  3     cli$get_value (sd_priority, tokendsc);
;  921          1070  3     if not getval (recbuf[uaf$b_pri], byte_length)
;  922          1071  3     then return false;
;  923          1072  3     status = true;
;  924          1073  2     end;
;  925          1074  2
;  926          1075  2 if cli$present (sd_privileges)
;  927          1076  2 then
;  928          1077  3     begin
;001 !JRL0038   1078  3     uaf$gq_sysuaff[nsa_v_privileges] = true;
;  929          1079  3     if not getpriv (sd_privileges, recbuf[uaf$q_priv])
;  930          1080  3     then return false;
;  931          1081  3     status = true;
;  932          1082  2     end;
;  933          1083  2
;  934          1084  2 CLI_STATUS = CLI$PRESENT(%ascid'PWDEXPIRED');
;  935          1085  2 if .CLI_STATUS eql CLI$_NEGATED then
;  936          1086  3    begin
;  937          1087  3       local TIME: long;
;001 !JRL0038   1088  3       uaf$gq_sysuaff[nsa_v_pwdlifetime] = true;
;002 !JRL0038   1089  3       TIME = RECBUF[UAF$Q_PWD_DATE];
;003 !JRL0038   1090  3       .TIME = 0; (.TIME+%upval) = 0;
;004 !JRL0038   1091  3       uaf$gq_sysuaff[nsa_v_flags] = true;
;  940-2        1092  3       RECBUF[UAF$V_PWD_EXPIRED] = FALSE;
;  941          1093  3       STATUS = TRUE;
;  942          1094  3    end
;  943          1095  2 else if .CLI_STATUS then
;  944          1096  3    begin
;  945          1097  3       local TIME: long;
;001 !JRL0038   1098  3       uaf$gq_sysuaff[nsa_v_pwdlifetime] = true;
;002 !JRL0038   1099  3       TIME = RECBUF[UAF$Q_PWD_DATE];
;003 !JRL0038   1100  3       .TIME = -1; (.TIME+%upval) = -1;
;004 !JRL0038   1101  3       uaf$gq_sysuaff[nsa_v_flags] = true;
;  948-2        1102  3       RECBUF[UAF$V_PWD_EXPIRED] = FALSE;
;  949          1103  3       STATUS = TRUE;
;  950          1104  2    end;
;  951          1105  2
;  952          1106  2 cli_status = cli$present (sd_pwdlifetime) ;
;  953          1107  2 if .cli_status eql cli$_negated
;  954          1108  2 then
;  955          1109  3     begin
;001 !JRL0038   1110  3     uaf$gq_sysuaff[nsa_v_pwdlifetime] = true;
;002 !JRL0038   1111  3     ch$fill ( 0, uaf$s_pwd_lifetime, recbuf[uaf$q_pwd_lifetime] ) ;
```

```
;003 !JRL0038   1112  3      uaf$gq_sysuaff[nsa_v_flags] = true;
;   957-1       1113  3      RECBUF[UAF$V_PWD_EXPIRED] = FALSE;
;   958         1114  3      status = true ;
;   959         1115  3      end
;   960         1116  2  else
;   961         1117  2      if .cli_status
;   962         1118  2      then
;   963         1119  3          begin
;001 !JRL0038   1120  3          uaf$gq_sysuaff[nsa_v_pwdlifetime] = true;
;   964         1121  3          cli$get_value (sd_pwdlifetime, tokendsc);
;   965         1122  3          if ch$eql(.tokendsc[dsc$w_length], .tokendsc[dsc$a_pointer],
;   966         1123  3                4, uplit(%ascii'NONE') ) then
;   967         1124  4             begin
;   968         1125  4                ch$fill(0, uaf$s_pwd_lifetime, recbuf[uaf$q_pwd_lifetime] );
;   969         1126  4                status = true;
;   970         1127  4             end
;   971         1128  3          else
;   972         1129  4          if not (status = lib$cvt_dtime (tokendsc, recbuf[uaf$q_pwd_lifetime]))
;   973         1130  3          then
;   974         1131  4              begin
;   975         1132  4              signal (.status);
;   976         1133  4              return false;
;   977         1134  3              end;
;001 !JRL0038   1135  3          uaf$gq_sysuaff[nsa_v_flags] = true;
;   978         1136  3          RECBUF[UAF$V_PWD_EXPIRED] = FALSE;
;   979         1137  3          status = true;
;   980         1138  2          end;
;   981         1139  2
;   982         1140  2  if cli$present (sd_pwdminimum)
;   983         1141  2  then
;   984         1142  3      begin
;001 !JRL0038   1143  3      uaf$gq_sysuaff[nsa_v_pwdminimum] = true;
;   985         1144  3      cli$get_value (sd_pwdminimum, tokendsc);
;   986         1145  3      if not getval (recbuf[uaf$b_pwd_length], byte_length)
;   987         1146  3      then return false;
;   988         1147  3      status = true;
;   989         1148  2      end;
;   990         1149  2
;   991         1150  2  if cli$present (sd_quepriority)
;   992         1151  2  then
;   993         1152  3      begin
;001 !JRL0038   1153  3      uaf$gq_sysuaff[nsa_v_quepriority] = true;
;   994         1154  3      cli$get_value (sd_quepriority, tokendsc);
;   995         1155  3      if not getval (recbuf[uaf$b_quepri], byte_length)
;   996         1156  3      then return false;
;   997         1157  3      status = true;
;   998         1158  2      end;
;   999         1159  2
;  1000         1160  2  if cli$present (sd_shrfillm)
;  1001         1161  2  then
;  1002         1162  3      begin
;001 !JRL0038   1163  3      uaf$gq_sysuaff[nsa_v_shrfillm] = true;
;  1003         1164  3      cli$get_value (sd_shrfillm, tokendsc);
;  1004         1165  3      if not getval (recbuf[uaf$w_shrfillm], word_length)
;  1005         1166  3      then return false;
;  1006         1167  3      status = true;
;  1007         1168  2      end;
```

```
;  1008          1169  2
;  1009          1170  2 if cli$present (sd_tqelm)
;  1010          1171  2 then
;  1011          1172  3     begin
;001 !JRL0038    1173  3     uaf$gq_sysuaff[nsa_v_tqelm] = true;
;  1012          1174  3     cli$get_value (sd_tqelm, tokendsc);
;  1013          1175  3     if not getval (recbuf[uaf$w_tqcnt], word_length)
;  1014          1176  3     then return false;
;  1015          1177  3     status = true;
;  1016          1178  2     end;
;  1017          1179  2
;  1018          1180  2 if cli$present (sd_uic)
;  1019          1181  2 then
;  1020          1182  3     begin
;001 !JRL0038    1183  3     uaf$gq_sysuaff[nsa_v_uic] = true;
;  1021          1184  3     cli$get_value (sd_uic, tokendsc);
;  1022          1185  3     if not getuic ()
;  1023          1186  3     then return false;
;  1024          1187  3     status = true;
;  1025          1188  2     end;
;  1026          1189  2
;  1027          1190  2 if cli$present (sd_wsdefault)
;  1028          1191  2 then
;  1029          1192  3     begin
;001 !JRL0038    1193  3     uaf$gq_sysuaff[nsa_v_wsdefault] = true;
;  1030          1194  3     cli$get_value (sd_wsdefault, tokendsc);
;  1031          1195  3     if not getval (recbuf[uaf$l_dfwscnt], word_unsigned_length)
;  1032          1196  3     then return false;
;  1033          1197  3     (recbuf[uaf$l_dfwscnt])<16, 16> = 0; !** temp until longword is supported
;  1034          1198  3     status = true;
;  1035          1199  2     end;
;  1036          1200  2
;  1037          1201  2 if cli$present (sd_wsextent)
;  1038          1202  2 then
;  1039          1203  3     begin
;001 !JRL0038    1204  3     uaf$gq_sysuaff[nsa_v_wsextent] = true;
;  1040          1205  3     cli$get_value (sd_wsextent, tokendsc);
;  1041          1206  3     if not getval (recbuf[uaf$l_wsextent], word_unsigned_length)
;  1042          1207  3     then return false;
;  1043          1208  3     (recbuf[uaf$l_wsextent])<16, 16> = 0; !** temp until longword is supported
;  1044          1209  3     status = true;
;  1045          1210  2     end;
;  1046          1211  2
;  1047          1212  2 if cli$present (sd_wsquota)
;  1048          1213  2 then
;  1049          1214  3     begin
;001 !JRL0038    1215  3     uaf$gq_sysuaff[nsa_v_wsquota] = true;
;  1050          1216  3     cli$get_value (sd_wsquota, tokendsc);
;  1051          1217  3     if not getval (recbuf[uaf$l_wsquota], word_unsigned_length)
;  1052          1218  3     then return false;
;  1053          1219  3     (recbuf[uaf$l_wsquota])<16, 16> = 0; !** temp until longword is supported
;  1054          1220  3     status = true;
;  1055          1221  2     end;
;  1056          1222  2
;  1057          1223  2 if .uaf$gl_ctlmsk[uaf$v_copy] or .uaf$gl_ctlmsk[uaf$v_add]
;  1058          1224  2 then
;  1059          1225  2     return true;
```

```
; 1060          1226  2
; 1061          1227  2 return .status;
; 1062          1228  1 end;


                                                                .TITLE  UAFPARSE
                                                                .IDENT  \V04-001\

                                                                .PSECT  $TPA_KEY1,NOWRT,  SHR,  PIC,1

                                                        00000 ;TPA$KEYSTO
                                                              U.2:      .BLKB   0
                                    59  52  41  4D  49  52  50  00000 ;TPA$KEYST
                                                              U.4:      .ASCII  \PRIMARY\                       ;
                                                          FF  00007             .BYTE   -1                      ;
                                                              00008 ;TPA$KEYSTO
                                                              U.11:     .BLKB   0
                                59  52  41  44  4E  4F  43  45  53  00008 ;TPA$KEYST
                                                              U.13:     .ASCII  \SECONDARY\                     ;
                                                          FF  00011             .BYTE   -1                      ;
                                                          FF  00012 ;TPA$KEYFILL
                                                              U.21:     .BYTE   -1                              ;

                                                                .PSECT  $TPA_STATE,NOWRT,  SHR,  PIC,1

                                                        00000 ACCESS_STATES::
                                                                .BLKB   0
                                                    F100  00000 ;TPA$TYPE
                                                              U.5:      .WORD   -3840                           ;
                                               00000000V  00002 ;TPA$ACTION
                                                              U.6:      .LONG   <<CHECK_PRIMARY-U.6>-4>         ;
                                               00000000*  00006 ;TPA$ADDR
                                                              U.7:      .LONG   <<PRIMARY-U.7>-4>               ;
                                               00000001   0000A ;TPA$MASK
                                                              U.8:      .LONG   1                               ;
                                                   0000*  0000E ;TPA$TARGET
                                                              U.10:     .WORD   <<U.9-U.10>-2>                  ;
                                                   F101   00010 ;TPA$TYPE
                                                              U.14:     .WORD   -3839                           ;
                                               00000000V  00012 ;TPA$ACTION
                                                              U.15:     .LONG   <<CHECK_SECONDARY-U.15>-4>      ;
                                               00000000*  00016 ;TPA$ADDR
                                                              U.16:     .LONG   <<SECONDARY-U.16>-4>            ;
                                               00000001   0001A ;TPA$MASK
                                                              U.17:     .LONG   1                               ;
                                                   0000*  0001E ;TPA$TARGET
                                                              U.18:     .WORD   <<U.9-U.18>-2>                  ;
                                                   45F3   00020 ;TPA$TYPE
                                                              U.19:     .WORD   17907                           ;
                                               00000000*  00022 ;TPA$ADDR
                                                              U.20:     .LONG   <<RIGHT_BIT-U.20>-4>            ;
                                                   002D   00026 ;TPA$TYPE
                                                              U.22:     .WORD   45                              ;
                                                   95F7   00028 ;TPA$TYPE
                                                              U.23:     .WORD   -27145                          ;
                                               00000000V  0002A ;TPA$ACTION
                                                              U.24:     .LONG   <<SET_RANGE-U.24>-4>            ;
```

```
                FFFF  0002E ;TPA$TARGET
                            U.25:    .WORD   -1                                          ;
                85F3  00030 ;TPA$TYPE
                            U.26:    .WORD   -31245                                      ;
00000000V       00032 ;TPA$ACTION
                            U.27:    .LONG   <<SET_RANGE-U.27>-4>                        ;
                      00036 ;TERMINAL
                            U.9:     .BLKB   0
                15F7  00036 ;TPA$TYPE
                            U.28:    .WORD   5623                                        ;
                FFFF  00038 ;TPA$TARGET
                            U.29:    .WORD   -1                                          ;
                      0003A          .BLKB   2
                      0003C RESTRICT_STATES::
                                     .BLKB   0
                45F3  0003C ;TPA$TYPE
                            U.31:    .WORD   17907                                       ;
00000000*       0003E ;TPA$ADDR
                            U.32:    .LONG   <<RIGHT_BIT-U.32>-4>                        ;
                002D  00042 ;TPA$TYPE
                            U.33:    .WORD   45                                          ;
                95F7  00044 ;TPA$TYPE
                            U.34:    .WORD   -27145                                      ;
00000000V       00046 ;TPA$ACTION
                            U.35:    .LONG   <<SET_RANGE-U.35>-4>                        ;
                FFFF  0004A ;TPA$TARGET
                            U.36:    .WORD   -1                                          ;
                85F3  0004C ;TPA$TYPE
                            U.37:    .WORD   -31245                                      ;
00000000V       0004E ;TPA$ACTION
                            U.38:    .LONG   <<SET_RANGE-U.38>-4>                        ;
                15F7  00052 ;TPA$TYPE
                            U.39:    .WORD   5623                                        ;
                FFFF  00054 ;TPA$TARGET
                            U.40:    .WORD   -1                                          ;
                      00056          .BLKB   2
                      00058 UIC_STATES::
                                     .BLKB   0
                55EC  00058 ;TPA$TYPE
                            U.42:    .WORD   21996                                       ;
00000000*       0005A ;TPA$ADDR
                            U.43:    .LONG   <<CONVERTED_UIC-U.43>-4>                    ;
                FFFF  0005E ;TPA$TARGET
                            U.44:    .WORD   -1                                          ;
                      00060 DIR_STATES::
                                     .BLKB   0
                19F8  00060 ;TPA$TYPE
                            U.46:    .WORD   6648                                        ;
                0000* 00062 ;TPA$SUBEXP
                            U.48:    .WORD   <<U.47-U.48>-2>                             ;
                0000* 00064 ;TPA$TARGET
                            U.50:    .WORD   <<U.49-U.50>-2>                             ;
                19F8  00066 ;TPA$TYPE
                            U.51:    .WORD   6648                                        ;
                0000* 00068 ;TPA$SUBEXP
                            U.53:    .WORD   <<U.52-U.53>-2>                             ;
                0000* 0006A ;TPA$TARGET
```

```
                        U.54:   .WORD   <<U.49-U.54>-2>                 ;
005B  0006C             ;TPA$TYPE
                        U.55:   .WORD   91                              ;
043C  0006E             ;TPA$TYPE
                        U.56:   .WORD   1084                            ;
09F8  00070             ;TPA$TYPE
                        U.57:   .WORD   2552                            ;
0000* 00072             ;TPA$SUBEXP
                        U.58:   .WORD   <<U.47-U.58>-2>                 ;
0DF8  00074             ;TPA$TYPE
                        U.59:   .WORD   3576                            ;
0000* 00076             ;TPA$SUBEXP
                        U.60:   .WORD   <<U.52-U.60>-2>                 ;
005D  00078             ;TPA$TYPE
                        U.61:   .WORD   93                              ;
043E  0007A             ;TPA$TYPE
                        U.62:   .WORD   1086                            ;
      0007C             ;END_STATE
                        U.49:   .BLKB   0
15F7  0007C             ;TPA$TYPE
                        U.63:   .WORD   5623                            ;
FFFF  0007E             ;TPA$TARGET
                        U.64:   .WORD   -1                              ;
      00080             ;UFD
                        U.47:   .BLKB   0
85F4  00080             ;TPA$TYPE
                        U.65:   .WORD   -31244                          ;
00000000V 00082         ;TPA$ACTION
                        U.66:   .LONG   <<CHECKVALUE-U.66>-4>           ;
042C  00086             ;TPA$TYPE
                        U.67:   .WORD   1068                            ;
95F4  00088             ;TPA$TYPE
                        U.68:   .WORD   -27148                          ;
00000000V 0008A         ;TPA$ACTION
                        U.69:   .LONG   <<CHECKVALUE-U.69>-4>           ;
FFFF  0008E             ;TPA$TARGET
                        U.70:   .WORD   -1                              ;
      00090             ;SUBDIR
                        U.52:   .BLKB   0
85F1  00090             ;TPA$TYPE
                        U.71:   .WORD   -31247                          ;
00000000V 00092         ;TPA$ACTION
                        U.72:   .LONG   <<CHECKLENGTH-U.72>-4>          ;
102E  00096             ;TPA$TYPE
                        U.73:   .WORD   4142                            ;
0000* 00098             ;TPA$TARGET
                        U.74:   .WORD   <<U.52-U.74>-2>                 ;
15F6  0009A             ;TPA$TYPE
                        U.75:   .WORD   5622                            ;
FFFF  0009C             ;TPA$TARGET
                        U.76:   .WORD   -1                              ;

                                .PSECT  $TPA_KEY0,NOWRT,  SHR,  PIC,1

      00000 ACCESS_KEYS::
                                .BLKB   0
      00000             ;TPA$KEY0
```

```
                                                00000     U.1:    .BLKB   0
                                          0000* 00000 ;TPA$KEY
                                                00000     U.3:    .WORD   <U.2-U.1>                       ;
                                          0000* 00002 ;TPA$KEY
                                                00002     U.12:   .WORD   <U.11-U.1>                      ;
                                                00004 RESTRICT_KEYS::
                                                00004             .BLKB   0
                                                00004 ;TPA$KEY0
                                                00004     U.30:   .BLKB   0
                                                00004 UIC_KEYS::
                                                00004             .BLKB   0
                                                00004 ;TPA$KEY0
                                                00004     U.41:   .BLKB   0
                                                00004 DIR_KEYS::
                                                00004             .BLKB   0
                                                00004 ;TPA$KEY0
                                                00004     U.45:   .BLKB   0

                                                                  .PSECT  $PLIT$,NOWRT,NOEXE,2

                                    31  6E  65  6B  6F  74 00000 P.AAB:  .ASCII  \token1\                  ;
                                                00006             .BLKB   2
                                      00000006  0000R P.AAA:  .LONG   6                                 ;
                                      00000000' 0000C         .ADDRESS P.AAB                            ;
                                    32  6E  65  6B  6F  74 00010 P.AAD:  .ASCII  \token2\                 ;
                                                00016             .BLKB   2
                                      00000006  00018 P.AAC:  .LONG   6                                 ;
                                      00000000' 0001C         .ADDRESS P.AAD                            ;
                                74  6F  75  6F  63  63  61 00020 P.AAF:  .ASCII  \account\               ;
                                                00027             .BLKB   1
                                      00000007  00028 P.AAE:  .LONG   7                                 ;
                                      00000000' 0002C         .ADDRESS P.AAF                            ;
                                        6D  6C  74  73  61 00030 P.AAH:  .ASCII  \astlm\                 ;
                                                00035             .BLKB   3
                                      00000005  00038 P.AAG:  .LONG   5                                 ;
                                      00000000' 0003C         .ADDRESS P.AAH                            ;
                                        6D  6C  6F  69  62 00040 P.AAJ:  .ASCII  \biolm\                 ;
                                                00045             .BLKB   3
                                      00000005  00048 P.AAI:  .LONG   5                                 ;
                                      00000000' 0004C         .ADDRESS P.AAJ                            ;
                                        6D  6C  74  79  62 00050 P.AAL:  .ASCII  \bytlm\                 ;
                                                00055             .BLKB   3
                                      00000005  00058 P.AAK:  .LONG   5                                 ;
                                      00000000' 0005C         .ADDRESS P.AAL                            ;
                                            69  6C  63  00060 P.AAN:  .ASCII  \cli\                     ;
                                                00063             .BLKB   1
                                      00000003  00064 P.AAM:  .LONG   3                                 ;
                                      00000000' 00068         .ADDRESS P.AAN                            ;
                            65  6D  69  74  75  70  63 0006C P.AAP:  .ASCII  \cputime\                   ;
                                                00073             .BLKB   1
                                      00000007  00074 P.AAO:  .LONG   7                                 ;
                                      00000000' 00078         .ADDRESS P.AAP                            ;
                                    65  63  69  76  65  64 0007C P.AAR:  .ASCII  \device\               ;
                                                00082             BLKB    2
                                      00000006  00084 P.AAQ:  .LONG   6                                 ;
                                      00000000' 00088         .ADDRESS P.AAR                            ;
                                        6D  6C  6F  69  64 0008C P.AAT:  .ASCII  \diolm\                 ;
```

```
                                                00091                .BLKB   3
                                    00000005    00094 P.AAS:         .LONG   5
                                    00000000'   00098                .ADDRESS P.AAT
                        79  72  6F  74  63  65  72  69  64  0009C P.AAV:  .ASCII  \directory\
                                                000A5                .BLKB   3
                                    00000009    000A8 P.AAU:         .LONG   9
                                    00000000'   000AC                .ADDRESS P.AAV
                                6D  6C  71  6E  65  000B0 P.AAX:     .ASCII  \enqlm\
                                                000B5                .BLKB   3
                                    00000005    000B8 P.AAW:         .LONG   5
                                    00000000'   000BC                .ADDRESS P.AAX
                                6D  6C  6C  69  66  000C0 P.AAZ:     .ASCII  \fillm\
                                                000C5                .BLKB   3
                                    00000005    000C8 P.AAY:         .LONG   5
                                    00000000'   000CC                .ADDRESS P.AAZ
                                73  67  61  6C  66  000D0 P.ABB:     .ASCII  \flags\
                                                000D5                .BLKB   3
                                    00000005    000D8 P.ABA:         .LONG   5
                                    00000000'   000DC                .ADDRESS P.ABB
                            64  6D  63  69  67  5C  000E0 P.ABD:     .ASCII  \lgicmd\
                                                000E6                .BLKB   2
                                    00000006    000E8 P.ABC:         .LONG   6
                                    00000000'   000EC                .ADDRESS P.ABD
                        73  62  6F  6A  78  61  6D  000F0 P.ABF:     .ASCII  \maxjobs\
                                                000F7                .BLKB   1
                                    00000007    000F8 P.ABE:         .LONG   7
                                    00000000'   000FC                .ADDRESS P.ABF
                                72  65  6E  77  6F  00100 P.ABH:     .ASCII  \owner\
                                                00105                .BLKB   3
                                    00000005    00108 P.ABG:         .LONG   5
                                    00000000'   0010C                .ADDRESS P.ABH
                    64  72  6F  77  73  73  61  70  00110 P.ABJ:     .ASCII  \password\
                                    00000008    00118 P.ABI:         .LONG   8
                                    00000000'   0011C                .ADDRESS P.ABJ
                            6D  6C  74  79  62  70  00120 P.ABL:     .ASCII  \pbytlm\
                                                00126                .BLKB   2
                                    00000006    00128 P.ABK:         .LONG   6
                                    00000000'   0012C                .ADDRESS P.ABL
                            73  67  61  6C  66  70  00130 P.ABN:     .ASCII  \pflags\
                                                00136                .BLKB   2
                                    00000006    00138 P.ABM:         .LONG   6
                                    00000000'   0013C                .ADDRESS P.ABN
                61  74  6F  75  71  6C  66  67  70  00140 P.ABP:     .ASCII  \pgflquota\
                                                00149                .BLKB   3
                                    00000009    0014C P.ABO:         .LONG   9
                                    00000000'   00150                .ADDRESS P.ABP
            74  63  69  72  74  73  65  72  5F  70  00154 P.ABR:     .ASCII  \p_restrict\
                                                0015E                .BLKB   2
                                    0000000A    00160 P.ABQ:         .LONG   10
                                    00000000'   00164                .ADDRESS P.ABR
                                6D  6C  63  72  70  00168 P.ABT:     .ASCII  \prclm\
                                                0016D                 BLKB   3
                                    00000005    00170 P.ABS:         .LONG   5
                                    00000000'   00174                .ADDRESS P.ABT
                    79  74  69  72  6F  69  72  70  00178 P.ABV:     .ASCII  \priority\
                                    00000008    00180 P.ABU:         .LONG   8
                                    00000000'   00184                .ADDRESS P.ABV
```

```
                                                00188 P.ABX:  .ASCII  \privileges\
                                                00192         .BLKB   2
                                          0000000A  00194 P.ABW:  .LONG   10
                                          00000000' 00198         .ADDRESS P.ABX
                                    73 67 61 6C 66 73 0019C P.ABZ:  .ASCII  \sflags\
                                                001A2         .BLKB   2
                                          00000006  001A4 P.ABY:  .LONG   6
                                          00000000' 001A8         .ADDRESS P.ABZ
                                 74 63 69 72 74 73 65 72 5F 73 001AC P.ACB:  .ASCII  \s_restrict\
                                                001B6         .BLKB   2
                                          0000000A  001B8 P.ACA:  .LONG   10
                                          00000000' 001BC         .ADDRESS P.ACB
                                    6D 6C 6C 69 66 72 68 73 001C0 P.ACD:  .ASCII  \shrfillm\
                                          00000008  001C8 P.ACC:  .LONG   8
                                          00000000' 001CC         .ADDRESS P.ACD
                                       6D 6C 65 71 74 001D0 P.ACF:  .ASCII  \tqelm\
                                                001D5         .BLKB   3
                                          00000005  001D8 P.ACE:  .LONG   5
                                          00000000' 001DC         .ADDRESS P.ACF
                                          63 69 75 001E0 P.ACH:  .ASCII  \uic\
                                                001E3         .BLKB   1
                                          00000003  001E4 P.ACG:  .LONG   3
                                          00000000' 001E8         .ADDRESS P.ACH
                                 74 6C 75 61 66 65 64 73 77 001EC P.ACJ:  .ASCII  \wsdefault\
                                                001F5         .BLKB   3
                                          00000009  001F8 P.ACI:  .LONG   9
                                          00000000' 001FC         .ADDRESS P.ACJ
                                    74 6E 65 74 78 65 73 77 00200 P.ACL:  .ASCII  \wsextent\
                                          00000008  00208 P.ACK:  .LONG   8
                                          00000000' 0020C         .ADDRESS P.ACL
                                       61 74 6F 75 71 73 77 00210 P.ACN:  .ASCII  \wsquota\
                                                00217         .BLKB   1
                                          00000007  00218 P.ACM:  .LONG   7
                                          00000000' 0021C         .ADDRESS P.ACN
                                       79 61 6C 70 73 69 64 00220 P.ACP:  .ASCII  \display\
                                                00227         .BLKB   1
                                          00000007  00228 P.ACO:  .LONG   7
                                          00000000' 0022C         .ADDRESS P.ACP
                                 73 79 61 64 65 6D 69 72 70 00230 P.ACR:  .ASCII  \primedays\
                                                00239         .BLKB   3
                                          00000009  0023C P.ACQ:  .LONG   9
                                          00000000' 00240         .ADDRESS P.ACR
                              73 62 6F 6A 74 65 63 61 78 61 6D 00244 P.ACT:  .ASCII  \maxacctjobs\
                                                0024F         .BLKB   1
                                          0000000B  00250 P.ACS:  .LONG   11
                                          00000000' 00254         .ADDRESS P.ACT
                                 68 63 61 74 65 64 78 61 6D 00258 P.ACV:  .ASCII  \maxdetach\
                                                00261         .BLKB   3
                                          00000009  00264 P.ACU:  .LONG   9
                                          00000000' 00268         .ADDRESS P.ACV
                                 73 65 6C 62 61 74 69 6C 63 0026C P.ACX:  .ASCII  \clitables\
                                                00275         .BLKB   3
                                          00000009  00278 P.ACW:  .LONG   9
                                          00000000' 0027C         .ADDRESS P.ACX
                                       61 74 6F 75 71 74 6A 00280 P.ACZ:  .ASCII  \jtquota\
                                                00287         .BLKB   1
                                          00000007  00288 P.ACY:  .LONG   7
```

```
                                     00000000' 0028C         .ADDRESS P.ACZ
                         73 73 65 63 63 61  00290 P.ADB:  .ASCII  \access\
                                           00296           .BLKB   2
                                  00000006 00298 P.ADA:  .LONG   6
                                  00000000' 0029C          .ADDRESS P.ADB
                            68 63 74 61 62  002A0 P.ADD:  .ASCII  \batch\
                                           002A5           .BLKB   3
                                  00000005 002A8 P.ADC:  .LONG   5
                                  00000000' 002AC          .ADDRESS P.ADD
   73 65 67 65 6C 69 76 69 72 70 66 65 64  002B0 P.ADF:  .ASCII  \defprivileges\
                                           002BD           .BLKB   3
                                  0000000D 002C0 P.ADE:  .LONG   13
                                  00000000' 002C4          .ADDRESS P.ADF
                         70 75 6C 61 69 64  002C8 P.ADH:  .ASCII  \dialup\
                                           002CE           .BLKB   2
                                  00000006 002D0 P.ADG:  .LONG   6
                                  00000000' 002D4          .ADDRESS P.ADH
         6E 6F 69 74 61 72 69 70 78 65  002D8 P.ADJ:  .ASCII  \expiration\
                                           002E2           .BLKB   2
                                  0000000A 002E4 P.ADI:  .LONG   10
                                  00000000' 002E8          .ADDRESS P.ADJ
      65 76 69 74 63 61 72 65 74 6E 69  002EC P.ADL:  .ASCII  \interactive\
                                           002F7           .BLKB   1
                                  0000000B 002F8 P.ADK:  .LONG   11
                                  00000000' 002FC          .ADDRESS P.ADL
                            6C 61 63 6F 6C  00300 P.ADN:  .ASCII  \local\
                                           00305           .BLKB   3
                                  00000005 00308 P.ADM:  .LONG   5
                                  00000000' 0030C          .ADDRESS P.ADN
                   6B 72 6F 77 74 65 6E  00310 P.ADP:  .ASCII  \network\
                                           00317           .BLKB   1
                                  00000007 00318 P.ADO:  .LONG   7
                                  00000000' 0031C          .ADDRESS P.ADP
      65 6D 69 74 65 66 69 6C 64 77 70  00320 P.ADR:  .ASCII  \pwdlifetime\
                                           0032B           .BLKB   1
                                  0000000B 0032C P.ADQ:  .LONG   11
                                  00000000' 00330          .ADDRESS P.ADR
         6D 75 6D 69 6E 69 6D 64 77 70  00334 P.ADT:  .ASCII  \pwdminimum\
                                           0033E           .BLKB   2
                                  0000000A 00340 P.ADS:  .LONG   10
                                  00000000' 00344          .ADDRESS P.ADT
      79 74 69 72 6F 69 72 70 65 75 71  00348 P.ADV:  .ASCII  \quepriority\
                                           00353           .BLKB   1
                                  0000000B 00354 P.ADU:  .LONG   11
                                  00000000' 00358          .ADDRESS P.ADV
                         65 74 6F 6D 65 72  0035C P.ADX:  .ASCII  \remote\
                                           00362           .BLKB   2
                                  00000006 00364 P.ADW:  .LONG   6
                                  00000000' 00368          .ADDRESS P.ADX
                59 4C 54 43 53 49 44 07  0036C P.ADY:  .ASCII  <7>\DISCTLY\
                00 49 4C 43 46 45 44 06  00374 P.ADZ:  .ASCII  <6>\DEFCLI\<0>
                44 57 50 4B 43 4F 4C 07  0037C P.AEA:  .ASCII  <7>\LOCKPWD\
                45 56 49 54 50 41 43 07  00384 P.AEB:  .ASCII  <7>\CAPTIVE\
                52 45 53 55 53 49 44 07  0038C P.AEC:  .ASCII  <7>\DISUSER\
    00 45 4D 4F 43 4C 45 57 53 49 44 0A  00394 P.AED:  .ASCII  <10>\DISWELCOME\<0>
    00 4C 49 41 4D 57 45 4E 53 49 44 0A  003A0 P.AEE:  .ASCII  <10>\DISNEWMAIL\<0>
                4C 49 41 4D 53 49 44 07  003AC P.AEF:  .ASCII  <7>\DISMAIL\
```

```
                                                  00 44 57 50 4E 45 47 06  003B4 P.AEG:   .ASCII  <6>\GENPWD\<0>
                                        44 45 52 49 50 58 45 5F 44 57 50 0B  003BC P.AEH:   .ASCII  <11>\PWD_EXPIRED\
00 00 44 45 52 49 50 58 45 5F 32 44 57 50 0C  003C8 P.AEI:   .ASCII  <12>\PWD2_EXPIRED\<0><0><0>
                                                  00 003D7
                                                  00 00 54 49 44 55 41 05  003D8 P.AEJ:   .ASCII  <5>\AUDIT\<0><0>
                                        00 00 54 52 4F 50 45 52 53 49 44 09  003E0 P.AEK:   .ASCII  <9>\DISREPORT\<0><0>
00 00 54 43 45 4E 4E 4F 43 45 52 53 49 44 0C  003EC P.AEL:   .ASCII  <12>\DISRECONNECT\<0><0><0>
                                                  00 003FB
                                        00 00 50 55 4C 41 49 44 53 49 44 09  003FC P.AEM:   .ASCII  <9>\DISDIALUP\<0><0>
                                        00 4B 52 4F 57 54 45 4E 53 49 44 0A  00408 P.AEN:   .ASCII  <10>\DISNETWORK\<0>
                                                  00 59 41 44 4E 4F 4D 06  00414 P.AEO:   .ASCII  <6>\MONDAY\<0>
                                                  59 41 44 53 45 55 54 07  0041C P.AEP:   .ASCII  <7>\TUESDAY\
                                        00 00 59 41 44 53 45 4E 44 45 57 09  00424 P.AEQ:   .ASCII  <9>\WEDNESDAY\<0><0>
                                        00 00 00 59 41 44 53 52 55 48 54 08  00430 P.AER:   .ASCII  <8>\THURSDAY\<0><0><0>
                                                  00 59 41 44 49 52 46 06  0043C P.AES:   .ASCII  <6>\FRIDAY\<0>
                                        00 00 00 59 41 44 52 55 54 41 53 08  00444 P.AET:   .ASCII  <8>\SATURDAY\<0><0><0>
                                                  00 59 41 44 4E 55 53 06  00450 P.AEU:   .ASCII  <6>\SUNDAY\<0>
45 44 43 42 41 39 38 37 36 35 34 33 32 31 30  00458 P.AEV:   .ASCII  \0123456789ABCDEF\
                                                  46 00467
                                                  4F 4E  00468 P.AEX:   .ASCII  \NO\
                                                         0046A          .BLKB   2
                                                  00000002  0046C P.AEW:   .LONG   2
                                                  00000000'  00470       .ADDRESS P.AEX
4F 57 53 53 41 50 5F 45 54 41 52 45 4E 45 47  00474 P.AEZ:   .ASCII  \GENERATE_PASSWORD\<0><0><0>
                                              00 00 00 44 52  00483
                                                  010E0011  00488 P.AEY:   .LONG   17694737
                                                  00000000'  0048C       .ADDRESS P.AEZ
4F 57 53 53 41 50 5F 45 54 41 52 45 4E 45 47  00490 P.AFB:   .ASCII  \GENERATE_PASSWORD\<0><0><0>
                                              00 00 00 44 52  0049F
                                                  010E0011  004A4 P.AFA:   .LONG   17694737
                                                  00000000'  004A8       .ADDRESS P.AFB
4F 57 53 53 41 50 5F 45 54 41 52 45 4E 45 47  004AC P.AFD:   .ASCII  \GENERATE_PASSWORD\<0><0><0>
                                              00 00 00 44 52  004BB
                                                  010E0011  004C0 P.AFC:   .LONG   17694737
                                                  00000000'  004C4       .ADDRESS P.AFD
                                        00 00 44 45 52 49 50 58 45 44 57 50  004C8 P.AFF:   .ASCII  \PWDEXPIRED\<0><0>
                                                  010E000A  004D4 P.AFE:   .LONG   17694730
                                                  00000000'  004D8       .ADDRESS P.AFF
                                                  45 4E 4F 4E  004DC P.AFG:   .ASCII  \NONE\

                                                                         .PSECT  $OWN$,NOEXE,2

                                                  0000001C  00000 FLAG_TABLE:
                                                                         .LONG   28
                                                  00000000'  00004       .ADDRESS P.ADY
                                                  00000000  00008        .LONG   0
                                                  00000000'  0000C       .ADDRESS P.ADZ
                                                  00000001  00010        .LONG   1
                                                  00000000'  00014       .ADDRESS P.AEA
                                                  00000002  00018        .LONG   2
                                                  00000000'  0001C       .ADDRESS P.AEB
                                                  00000003  00020        .LONG   3
                                                  00000000'  00024       .ADDRESS P.AEC
                                                  00000004  00028        .LONG   4
                                                  00000000'  0002C       .ADDRESS P.AED
                                                  00000005  00030        .LONG   5
                                                  00000000'  00034       .ADDRESS P.AEE
```

```
                                00000006  00038          .LONG   6
                                00000000' 0003C          .ADDRESS P.AEF
                                00000007  00040          .LONG   7
                                00000000' 00044          .ADDRESS P.AEG
                                00000008  00048          .LONG   8
                                00000000' 0004C          .ADDRESS P.AEH
                                00000009  00050          .LONG   9
                                00000000' 00054          .ADDRESS P.AEI
                                0000000A  00058          .LONG   10
                                00000000' 0005C          .ADDRESS P.AEJ
                                0000000B  00060          .LONG   11
                                00000000' 00064          .ADDRESS P.AEK
                                0000000C  00068          .LONG   12
                                00000000' 0006C          .ADDRESS P.AEL
                                0000000D  00070          .LONG   13
                                00000004  00074 OPFLAG_TABLE:
                                                         .LONG   4
                                00000000' 00078          .ADDRESS P.AEM
                                00000001  0007C          .LONG   1
                                00000000' 00080          .ADDRESS P.AEN
                                00000002  00084          .LONG   2
                                0000000E  00088 PDFLAG_TABLE:
                                                         .LONG   14
                                00000000' 0008C          .ADDRESS P.AEO
                                00000000  00090          .LONG   0
                                00000000' 00094          .ADDRESS P.AEP
                                00000001  00098          .LONG   1
                                00000000' 0009C          .ADDRESS P.AEQ
                                00000002  000A0          .LONG   2
                                00000000' 000A4          .ADDRESS P.AER
                                00000003  000A8          .LONG   3
                                00000000' 000AC          .ADDRESS P.AES
                                00000004  000B0          .LONG   4
                                00000000' 000B4          .ADDRESS P.AET
                                00000005  000B8          .LONG   5
                                00000000' 000BC          .ADDRESS P.AEU
                                00000006  000C0          .LONG   6
                                          000C4 STATUS:  .BLKB   4
                                          000C8 CLI_STATUS:
                                                         .BLKB   4
                00000002  00000008        000CC TPARSE_BLOCK:
                                                         .LONG   8, 2
                                          000D4          .BLKB   28
                                          000F0 CONVERTED_UIC:
                                                         .BLKB   4
                                          000F4 ADDRESS: .BLKB   4
                                          000F8 LEFT_BIT:
                                                         .BLKB   4
                                          000FC RIGHT_BIT:
                                                         .BLKB   4
                                          00100 PRIMARY_ACCESS:
                                                         .BLKB   4
                                          00104 SECONDARY_ACCESS:
                                                         .BLKB   4
                                00000000  00108 NO_FLAG: .LONG   0
                                00000000  0010C PRIMARY: .LONG   0
                                00000000  00110 SECONDARY:
```

```
                .LONG   0                                               ;

SD_TOKEN1=              P.AAA
SD_TOKEN2=              P.AAC
SD_ACCOUNT=             P.AAE
SD_ASTLM=               P.AAG
SD_BIOLM=               P.AAI
SD_BYTLM=               P.AAK
SD_CLI=                 P.AAM
SD_CPUTIME=             P.AAO
SD_DEVICE=              P.AAQ
SD_DIOLM=               P.AAS
SD_DIRECTORY=           P.AAU
SD_ENQLM=               P.AAW
SD_FILLM=               P.AAY
SD_FLAGS=               P.ABA
SD_LGICMD=              P.ABC
SD_MAXJOBS=             P.ABE
SD_OWNER=               P.ABG
SD_PASSWORD=            P.ABI
SD_PBYTLM=              P.ABK
SD_PFLAGS=              P.ABM
SD_PGFLQUOTA=           P.ABO
SD_P_RESTRICT=          P.ABQ
SD_PRCLM=               P.ABS
SD_PRIORITY=            P.ABU
SD_PRIVILEGES=          P.ABW
SD_SFLAGS=              P.ABY
SD_S_RESTRICT=          P.ACA
SD_SHRFILLM=            P.ACC
SD_TQELM=               P.ACE
SD_UIC=                 P.ACG
SD_WSDEFAULT=           P.ACI
SD_WSEXTENT=            P.ACK
SD_WSQUOTA=             P.ACM
SD_DISPLAY=             P.ACO
SD_PRIMEDAYS=           P.ACQ
SD_MAXACCTJOBS=         P.ACS
SD_MAXDETACH=           P.ACU
SD_CLITABLES=           P.ACW
SD_JTQUOTA=             P.ACY
SD_ACCESS=              P.ADA
SD_BATCH=               P.ADC
SD_DEFPRIVILEGES=       P.ADE
SD_DIALUP=              P.ADG
SD_EXPIRATION=          P.ADI
SD_INTERACTIVE=         P.ADK
SD_LOCAL=               P.ADM
SD_NETWORK=             P.ADO
SD_PWDLIFETIME=         P.ADQ
SD_PWDMINIMUM=          P.ADS
SD_QUEPRIORITY=         P.ADU
SD_REMOTE=              P.ADW
NUMBERS=                P.AEV
NO_DSC=                 P.AEW
                .EXTRN  GENERATE_PASSWORDS
                .EXTRN  SYS$ASSIGN, SYS$DASSGN
```

```
                                                .EXTRN  SYS$QIOW, LIB$GET_VM
                                                .EXTRN  STR$UPCASE, LIB$TPARSE
                                                .EXTRN  LIB$LOOKUP_KEY, CLI$DCL_PARSE
                                                .EXTRN  CLI$PRESENT, CLI$GET_VALUE
                                                .EXTRN  FAO$OUT, LGI$HPWD
                                                .EXTRN  LIB$CVT_DTIME, LIB$CVT_TIME
                                                .EXTRN  PRV$SETPRIV, LIB$SIGNAL
                                                .EXTRN  UAF$_BADVALUE, UAF$_EXTRAPARM
                                                .EXTRN  UAF$_INVDEV, UAF$_INVSTR
                                                .EXTRN  UAF$_INVUSERSPEC
                                                .EXTRN  UAF$_KEYNOTFND, UAF$_NAMETOOBIG
                                                .EXTRN  UAF$_NOARG, UAF$_NOQUOTE
                                                .EXTRN  UAF$_NOUSERSPEC
                                                .EXTRN  UAF$_PRVNOTFND, UAF$_PRVNOTUNQ
                                                .EXTRN  UAF$_PWDSYNTAX, UAF$_PWDNCH
                                                .EXTRN  UAF$_PWDNOL, UAF$_UICERR
                                                .EXTRN  UAF$_VALTOOBIG, UAF$_ZCONFLICT
                                                .EXTRN  ENCRYPT, LIB$_AMBKEY
                                                .EXTRN  LIB$_UNKKEY, CLI$_ABKEYW
                                                .EXTRN  CLI$_IVKEYW, CLI$_NEGATED
                                                .EXTRN  CLI$_ABSENT, PRV$_NOTUNQ
                                                .EXTRN  PRV$_INVNAM, UAF$GQ_SYSUAFF
                                                .EXTRN  UAF$GL_CTLMSK, AUTHORIZE_COMMANDS
                                                .EXTRN  CMDLINDSC, CALL_COUNT
                                                .EXTRN  SYMBOL_STR, MATCH_TOKENLEN
                                                .EXTRN  MATCH_TOKEN, TOKENDSC
                                                .EXTRN  TOKENLEN, TOKENPTR
                                                .EXTRN  UIC_FLAG, GRP_WILD
                                                .EXTRN  MEM_WILD, STR_WILD
                                                .EXTRN  TIME_BUF, PWD_FLAG
                                                .EXTRN  RECBUF

                                                .PSECT  $CODE$,NOWRT,2

                                    0FFC 00000          .ENTRY  UPDATE_RECORD, Save R2,R3,R4,R5,R6,R7,R8,-      ; 0600
                                                                R9,R10,R11
                5B 00000000G    00  9E 00002          MOVAB   TOKENDSC, R11
                5A 00000000G    00  9E 00009          MOVAB   RECBUF+468, R10
                59 00000000G    00  9E 00010          MOVAB   CLI$PRESENT, R9
                58 00000000G    00  9E 00017          MOVAB   UAF$GQ_SYSUAFF, R8
                57 00000000'    00  9E 0001E          MOVAB   STATUS, R7
                56 00000000'    00  9E 00025          MOVAB   SD_PASSWORD, R6
                                67  D4 0002C          CLRL    STATUS                                          ; 0635
                      48        A7  7C 0002E          CLRQ    PRIMARY                                         ; 0636
                      44        A7  D4 00031          CLRL    NO_FLAG                                         ; 0637
                                68  7C 00034          CLRQ    UAF$GQ_SYSUAFF                                  ; 0640
                   00000000G    00  D5 00036          TSTL    CALL_COUNT                                      ; 0648
                                13  13 0003C          BEQL    1$
                   00000000G    00  9F 0003E          PUSHAB  AUTHORIZE_COMMANDS                              ; 0650
                   00000000G    00  9F 00044          PUSHAB  CMDLINDSC
00000000G       00               02  FB 0004A          CALLS   #2, CLI$DCL_PARSE
                      0370      C6  9F 00051 1$:      PUSHAB  P.AEY                                           ; 0653
                69              01  FB 00055          CALLS   #1, CLI$PRESENT
                1D              50  E9 00058          BLBC    R0, 2$
                                56  DD 0005B          PUSHL   R6                                              ; 0654
                69              01  FB 0005D          CALLS   #1, CLI$PRESENT
00000000G       8F              50  D1 00060          CMPL    R0, #CLI$_ABSENT
```

```
                                          0F   13 00067          BEQL     2$
                             00000000G    8F   DD 00069          PUSHL    #UAF$_ZCONFLICT                          ; 0657
            00000000G  00                 01   FB 0006F          CALLS    #1, LIB$SIGNAL
                                          3E   11 00076          BRB      5$                                       ; 0653
                                 038C     C6   9F 00078  2$:     PUSHAB   P.AFA                                    ; 0660
                       69                 01   FB 0007C          CALLS    #1, CLI$PRESENT
                       0E                 50   E8 0007F          BLBS     R0, 3$
                                          56   DD 00082          PUSHL    R6                                       ; 0661
                       69                 01   FB 00084          CALLS    #1, CLI$PRESENT
            00000000G  8F                 50   D1 00087          CMPL     R0, #CLI$_ABSENT
                                          26   13 0008E          BEQL     5$
                   03  A8                 08   88 00090  3$:     BISB2    #8, UAF$GQ_SYSUAFF+3                     ; 0664
                                 03A8     C6   9F 00094          PUSHAB   P.AFC                                    ; 0666
                       69                 01   FB 00098          CALLS    #1, CLI$PRESENT
                       07                 50   E9 0009B          BLBC     R0, 4$
            00000000V  00                 00   FB 0009E          CALLS    #0, UAF$GENERATE                         ; 0667
            00000000V  00                 00   FB 000A5  4$:     CALLS    #0, GETPASSWORD                          ; 0668
                       6A                 50   E9 000AC          BLBC     R0, 10$
                   01  AA                 02   8A 000AF          BICB2    #2, RECBUF+469                           ; 0670
                       67                 01   D0 000B3          MOVL     #1, STATUS                               ; 0671
                       03  00000000G      00   E9 000B6  5$:     BLBC     UAF$GL_CTLMSK, 6$                        ; 0676
                                       08BA    31 000BD          BRW      107$
                                 0180     C6   9F 000C0  6$:     PUSHAB   SD_ACCESS                                ; 0685
                       69                 01   FB 000C4          CALLS    #1, CLI$PRESENT
                   04  A7                 50   D0 000C7          MOVL     R0, CLI_STATUS
                       09                 50   E8 000CB          BLBS     R0, 7$                                   ; 0686
            00000000G  8F                 50   D1 000CE          CMPL     R0, #CLI$_NEGATED                        ; 0687
                                          18   12 000D5          BNEQ     8$
                       68                 01   88 000D7  7$:     BISB2    #1, UAF$GQ_SYSUAFF                       ; 0690
                                          50   DD 000DA          PUSHL    R0                                       ; 0691
                                          1F   DD 000DC          PUSHL    #31
                                 0180     C6   9F 000DE          PUSHAB   SD_ACCESS
            00000000V  00                 03   FB 000E2          CALLS    #3, GETACCESS
                       5C                 50   E9 000E9          BLBC     R0, 13$
                       67                 01   D0 000EC          MOVL     #1, STATUS                               ; 0693
                                 01E0     C6   9F 000EF  8$:     PUSHAB   SD_INTERACTIVE                           ; 0696
                       69                 01   FB 000F3          CALLS    #1, CLI$PRESENT
                   04  A7                 50   D0 000F6          MOVL     R0, CLI_STATUS
                       09                 50   E8 000FA          BLBS     R0, 9$                                   ; 0697
            00000000G  8F                 50   D1 000FD          CMPL     R0, #CLI$_NEGATED                        ; 0698
                                          19   12 00104          BNEQ     11$
                   02  A8                 04   88 00106  9$:     BISB2    #4, UAF$GQ_SYSUAFF+2                     ; 0701
                                          50   DD 0010A          PUSHL    R0                                       ; 0702
                                          1C   DD 0010C          PUSHL    #28
                                 01E0     C6   9F 0010E          PUSHAB   SD_INTERACTIVE
            00000000V  00                 03   FB 00112          CALLS    #3, GETACCESS
                       5C                 50   E9 00119  10$:    BLBC     R0, 16$
                       67                 01   D0 0011C          MOVL     #1, STATUS                               ; 0704
                                 0190     C6   9F 0011F  11$:    PUSHAB   SD_BATCH                                 ; 0707
                       69                 01   FB 00123          CALLS    #1, CLI$PRESENT
                   04  A7                 50   D0 00126          MOVL     R0, CLI_STATUS
                       09                 50   E8 0012A          BLBS     R0, 12$                                  ; 0708
            00000000G  8F                 50   D1 0012D          CMPL     R0, #CLI$_NEGATED                        ; 0709
                                          18   12 00134          BNEQ     14$
                       68                 08   88 00136  12$:    BISB2    #8, UAF$GQ_SYSUAFF                       ; 0712
                                          50   DD 00139          PUSHL    R0                                       ; 0713
                                          02   DD 0013B          PUSHL    #2
```

```
                                0190  C6  9F 0013D          PUSHAB  SD_BATCH                                    ;
                00000000V       00    03  FB 00141          CALLS   #3, GETACCESS                             ;
                                5D    50  E9 00148 13$:     BLBC    R0, 19$                                   ;
                                67    01  D0 0014B          MOVL    #1, STATUS                                ; 0715
                                0188  C6  9F 0014E 14$:     PUSHAB  SD_DIALUP                                 ; 0718
                                69    01  FB 00152          CALLS   #1, CLI$PRESENT                           ;
                          04    A7    50  D0 00155          MOVL    R0, CLI_STATUS                            ;
                                09    50  E8 00159          BLBS    R0, 15$                                   ; 0719
                00000000G       8F    50  D1 0015C          CMPL    R0, #CLI$_NEGATED                         ; 0720
                                      19  12 00163          BNEQ    17$                                       ;
                          01    A8          08  88 00165 15$:     BISB2   #8, UAF$GQ_SYSUAFF+1                      ; 0723
                                            50  DD 00169          PUSHL   R0                                        ; 0724
                                            08  DD 0016B          PUSHL   #8                                        ;
                                      01B8  C6  9F 0016D          PUSHAB  SD_DIALUP                                 ;
                00000000V       00    03  FB 00171          CALLS   #3, GETACCESS                             ;
                                5D    50  E9 00178 16$:     BLBC    R0, 22$                                   ;
                                67    01  D0 0017B          MOVL    #1, STATUS                                ; 0726
                                0200  C6  9F 0017E 17$:     PUSHAB  SD_NETWORK                                ; 0729
                                69    01  FB 00182          CALLS   #1, CLI$PRESENT                           ;
                          04    A7    50  D0 00185          MOVL    R0, CLI_STATUS                            ;
                                09    50  E8 00189          BLBS    R0, 18$                                   ; 0730
                00000000G       8F    50  D1 0018C          CMPL    R0, #CLI$_NEGATED                         ; 0731
                                      19  12 00193          BNEQ    20$                                       ;
                          03    A8          02  88 00195 18$:     BISB2   #2, UAF$GQ_SYSUAFF+3                      ; 0734
                                            50  DD 00199          PUSHL   R0                                        ; 0735
                                            01  DD 0019B          PUSHL   #1                                        ;
                                      0200  C6  9F 0019D          PUSHAB  SD_NETWORK                                ;
                00000000V       00    03  FB 001A1          CALLS   #3, GETACCESS                             ;
                                5E    50  E9 001A8 19$:     BLBC    R0, 25$                                   ;
                                67    01  D0 001AB          MOVL    #1, STATUS                                ; 0737
                                01F0  C6  9F 001AE 20$:     PUSHAB  SD_LOCAL                                  ; 0740
                                69    01  FB 001B2          CALLS   #1, CLI$PRESENT                           ;
                          04    A7    50  D0 001B5          MOVL    R0, CLI_STATUS                            ;
                                09    50  E8 001B9          BLBS    R0, 21$                                   ; 0741
                00000000G       8F    50  D1 001BC          CMPL    R0, #CLI$_NEGATED                         ; 0742
                                      19  12 001C3          BNEQ    23$                                       ;
                          02    A8          20  88 001C5 21$:     BISB2   #32, UAF$GQ_SYSUAFF+2                     ; 0745
                                            50  DD 001C9          PUSHL   R0                                        ; 0746
                                            04  DD 001CB          PUSHL   #4                                        ;
                                      01F0  C6  9F 001CD          PUSHAB  SD_LOCAL                                  ;
                00000000V       00    03  FB 001D1          CALLS   #3, GETACCESS                             ;
                                5D    50  E9 001D8 22$:     BLBC    R0, 27$                                   ;
                                67    01  D0 001DB          MOVL    #1, STATUS                                ; 0748
                                024C  C6  9F 001DE 23$:     PUSHAB  SD_REMOTE                                 ; 0751
                                69    01  FB 001E2          CALLS   #1, CLI$PRESENT                           ;
                          04    A7    50  D0 001E5          MOVL    R0, CLI_STATUS                            ;
                                09    50  E8 001E9          BLBS    R0, 24$                                   ; 0752
                00000000G       8F    50  D1 001EC          CMPL    R0, #CLI$_NEGATED                         ; 0753
                                      1A  12 001F3          BNEQ    26$                                       ;
                          04    A8    80    8F  88 001F5 24$:     BISB2   #128, UAF$GQ_SYSUAFF+4                    ; 0756
                                            50  DD 001FA          PUSHL   R0                                        ; 0757
                                            10  DD 001FC          PUSHL   #16                                       ;
                                      024C  C6  9F 001FE          PUSHAB  SD_REMOTE                                 ;
                00000000V       00    03  FB 00202          CALLS   #3, GETACCESS                             ;
                                58    50  E9 00209 25$:     BLBC    R0, 29$                                   ;
                                67    01  D0 0020C          MOVL    #1, STATUS                                ; 0759
                                FF10  C6  9F 0020F 26$:     PUSHAB  SD_ACCOUNT                                ; 0762
```

```
                69            01   FB 00213        CALLS   #1, CLI$PRESENT               :
                25            50   E9 00216        BLBC    R0, 28$                       :
                68            02   88 00219        BISB2   #2, UAF$GQ_SYSUAFF            : 0765
                              5B   DD 0021C        PUSHL   R11                           : 0766
                      FF10    C6   9F 0021E        PUSHAB  SD_ACCOUNT                    :
    00000000G   00            02   FB 00222        CALLS   #2, CLI$GET_VALUE             :
                              02   DD 00229        PUSHL   #2                            : 0768
                              08   DD 0022B        PUSHL   #8                            :
                      FE60    CA   9F 0022D        PUSHAB  RECBUF+52                     :
    00000000V   00            03   FB 00231        CALLS   #3, GETSTRING                 :
                55            50   E9 00238  27$:  BLBC    R0, 31$                       :
                67            01   D0 0023B        MOVL    #1, STATUS                    : 0770
                      FF20    C6   9F 0023E  28$:  PUSHAB  SD_ASTLM                      : 0773
                69            01   FB 00242        CALLS   #1, CLI$PRESENT               :
                22            50   E9 00245        BLBC    R0, 30$                       :
                68            04   88 00248        BISB2   #4, UAF$GQ_SYSUAFF            : 0776
                              5B   DD 0024B        PUSHL   R11                           : 0777
                      FF20    C6   9F 0024D        PUSHAB  SD_ASTLM                      :
    00000000G   00            02   FB 00251        CALLS   #2, CLI$GET_VALUE             :
                              10   DD 00258        PUSHL   #16                           : 0778
                        40    AA   9F 0025A        PUSHAB  RECBUF+532                    :
    00000000V   00            02   FB 0025D        CALLS   #2, GETVAL                    :
                55            50   E9 00264  29$:  BLBC    R0, 33$                       :
                67            01   D0 00267        MOVL    #1, STATUS                    : 0780
                      FF30    C6   9F 0026A  30$:  PUSHAB  SD_BIOLM                      : 0783
                69            01   FB 0026E        CALLS   #1, CLI$PRESENT               :
                22            50   E9 00271        BLBC    R0, 32$                       :
                68            10   88 00274        BISB2   #16, UAF$GQ_SYSUAFF           : 0786
                              5B   DD 00277        PUSHL   R11                           : 0787
                      FF30    C6   9F 00279        PUSHAB  SD_BIOLM                      :
    00000000G   00            02   FB 0027D        CALLS   #2, CLI$GET_VALUE             :
                              10   DD 00284        PUSHL   #16                           : 0788
                        3A    AA   9F 00286        PUSHAB  RECBUF+526                    :
    00000000V   00            02   FB 00289        CALLS   #2, GETVAL                    :
                59            50   E9 00290  31$:  BLBC    R0, 35$                       :
                67            01   D0 00293        MOVL    #1, STATUS                    : 0790
                      FF40    C6   9F 00296  32$:  PUSHAB  SD_BYTLM                      : 0793
                69            01   FB 0029A        CALLS   #1, CLI$PRESENT               :
                22            50   E9 0029D        BLBC    R0, 34$                       :
                68            20   88 002A0        BISB2   #32, UAF$GQ_SYSUAFF           : 0796
                              5B   DD 002A3        PUSHL   R11                           : 0797
                      FF40    C6   9F 002A5        PUSHAB  SD_BYTLM                      :
    00000000G   00            02   FB 002A9        CALLS   #2, CLI$GET_VALUE             :
                              20   DD 002B0        PUSHL   #32                           : 0798
                        5C    AA   9F 002B2        PUSHAB  RECBUF+560                    :
    00000000V   00            02   FB 002B5        CALLS   #2, GETVAL                    :
                77            50   E9 002BC  33$:  BLBC    R0, 38$                       :
                67            01   D0 002BF        MOVL    #1, STATUS                    : 0800
                      FF4C    C6   9F 002C2  34$:  PUSHAB  SD_CLI                        : 0803
                69            01   FB 002C6        CALLS   #1, CLI$PRESENT               :
                40            50   E9 002C9        BLBC    R0, 37$                       :
                68      40    8F   88 002CC        BISB2   #64, UAF$GQ_SYSUAFF           : 0806
                              5B   DD 002D0        PUSHL   R11                           : 0807
                      FF4C    C6   9F 002D2        PUSHAB  SD_CLI                        :
    00000000G   00            02   FB 002D6        CALLS   #2, CLI$GET_VALUE             :
                              01   DD 002DD        PUSHL   #1                            : 0808
                              20   DD 002DF        PUSHL   #32                           :
```

```
                                         FF40  CA  9F  002E1       PUSHAB  RECBUF+276                    ;
                       00000000V  00           03  FB  002E5       CALLS   #3, GETSTRING                 ;
                                  76           50  E9  002EC  35$: BLBC    R0, 40$                       ;
                                               50  D4  002EF       CLRL    R0                            ; 0811
                                         FF60  CA  95  002F1       TSTB    RECBUF+308                    ;
                                               02  13  002F5       BEQL    36$                           ;
                                               50  D6  002F7       INCL    R0                            ;
00000000G  00            01       05           50  F0  002F9  36$: INSV    R0, #5, #1, UAF$GL_CTLMSK     ;
                       00000000G  00           08  88  00302       BISB2   #8, UAF$GL_CTLMSK             ; 0812
                                  67           01  D0  00309       MOVL    #1, STATUS                    ; 0813
                                         0160  C6  9F  0030C  37$: PUSHAB  SD_CLITABLES                  ; 0816
                                  69           01  FB  00310       CALLS   #1, CLI$PRESENT               ;
                                  2D           50  E9  00313       BLBC    R0, 39$                       ;
                                  58       80  8F  88  00316       BISB2   #128, UAF$GQ_SYSUAFF          ; 0819
                                               5B  DD  0031A       PUSHL   R11                           ; 0820
                                         0160  C6  9F  0031C       PUSHAB  SD_CLITABLES                  ;
                       00000000G  00           02  FB  00320       CALLS   #2, CLI$GET_VALUE             ;
                                               01  DD  00327       PUSHL   #1                            ; 0821
                                               20  DD  00329       PUSHL   #32                           ;
                                         FF60  CA  9F  0032B       PUSHAB  RECBUF+308                    ;
                       00000000V  00           03  FB  0032F       CALLS   #3, GETSTRING                 ;
                                  76           50  E9  00336  38$: BLBC    R0, 43$                       ;
                       00000000G  00           10  88  00339       BISB2   #16, UAF$GL_CTLMSK            ; 0823
                                  67           01  D0  00340       MOVL    #1, STATUS                    ; 0824
                                         FF5C  C6  9F  00343  39$: PUSHAB  SD_CPUTIME                    ; 0827
                                  69           01  FB  00347       CALLS   #1, CLI$PRESENT               ;
                                  1E           50  E9  0034A       BLBC    R0, 41$                       ;
                             01   A8           01  88  0034D       BISB2   #1, UAF$GQ_SYSUAFF+1          ; 0830
                                               5B  DD  00351       PUSHL   R11                           ; 0831
                                         FF5C  C6  9F  00353       PUSHAB  SD_CPUTIME                    ;
                       00000000G  00           02  FB  00357       CALLS   #2, CLI$GET_VALUE             ;
                       00000000V  00           00  FB  0035E       CALLS   #0, GETCPUTIM                 ; 0832
                                  74           50  E9  00365  40$: BLBC    R0, 45$                       ;
                                  67           01  D0  00368       MOVL    #1, STATUS                    ; 0834
                                         01A8  C6  9F  0036B  41$: PUSHAB  SD_DEFPRIVILEGES              ; 0837
                                  69           01  FB  0036F       CALLS   #1, CLI$PRESENT               ;
                                  18           50  E9  00372       BLBC    R0, 42$                       ;
                             01   A8           02  88  00375       BISB2   #2, UAF$GQ_SYSUAFF+1          ; 0840
                                           D0  AA  9F  00379       PUSHAB  RECBUF+420                    ; 0841
                                         01A8  C6  9F  0037C       PUSHAB  SD_DEFPRIVILEGES              ;
                       00000000V  00           02  FB  00380       CALLS   #2, GETPRIV                   ;
                                  78           50  E9  00387       BLBC    R0, 47$                       ;
                                  67           01  D0  0038A       MOVL    #1, STATUS                    ; 0843
                                         FF6C  C6  9F  0038D  42$: PUSHAB  SD_DEVICE                     ; 0846
                                  69           01  FB  00391       CALLS   #1, CLI$PRESENT               ;
                                  1E           50  E9  00394       BLBC    R0, 44$                       ;
                             01   A8           04  88  00397       BISB2   #4, UAF$GQ_SYSUAFF+1          ; 0849
                                               5B  DD  0039B       PUSHL   R11                           ; 0850
                                         FF6C  C6  9F  0039D       PUSHAB  SD_DEVICE                     ;
                       00000000G  00           02  FB  003A1       CALLS   #2, CLI$GET_VALUE             ;
                       00000000V  00           00  FB  003A8       CALLS   #0, GETDEVICE                 ; 0851
                                  7C           50  E9  003AF  43$: BLBC    R0, 49$                       ;
                                  67           01  D0  003B2       MOVL    #1, STATUS                    ; 0853
                                         FF7C  C6  9F  003B5  44$: PUSHAB  SD_DIOLM                      ; 0856
                                  69           01  FB  003B9       CALLS   #1, CLI$PRESENT               ;
                                  23           50  E9  003BC       BLBC    R0, 46$                       ;
                             01   A8           10  88  003BF       BISB2   #16, UAF$GQ_SYSUAFF+1         ; 0859
```

```
                                     5B  DD 003C3         PUSHL   R11                                   ; 0860
                              FF7C   C6  9F 003C5         PUSHAB  SD_DIOLM                              ;
                000000000G  00       02  FB 003C9         CALLS   #2, CLI$GET_VALUE                     ;
                                     10  DD 003D0         PUSHL   #16                                   ; 0861
                              3C     AA  9F 003D2         PUSHAB  RECBUF+528                            ;
                00000000V   00       02  FB 003D5         CALLS   #2, GETVAL                            ;
                            4F       50  E9 003DC  45$:   BLBC    R0, 49$                               ;
                            67       01  D0 003DF         MOVL    #1, STATUS                            ; 0863
                              90     A6  9F 003E2  46$:   PUSHAB  SD_DIRECTORY                          ; 0866
                            69       01  FB 003E5         CALLS   #1, CLI$PRESENT                       ;
                            1D       50  E9 003E8         BLBC    R0, 48$                               ;
                     01     A8       20  88 003EB         BISB2   #32, UAF$GQ_SYSUAFF+1                 ; 0869
                                     5B  DD 003EF         PUSHL   R11                                   ; 0870
                              90     A6  9F 003F1         PUSHAB  SD_DIRECTORY                          ;
                000000000G  00       02  FB 003F4         CALLS   #2, CLI$GET_VALUE                     ;
                00000000V   00       00  FB 003FB         CALLS   #0, GETDIRECTORY                      ; 0871
                            29       50  E9 00402  47$:   BLBC    R0, 49$                               ;
                            67       01  D0 00405         MOVL    #1, STATUS                            ; 0873
                              A0     A6  9F 00408  48$:   PUSHAB  SD_ENQLM                              ; 0876
                            69       01  FB 0040B         CALLS   #1, CLI$PRESENT                       ;
                            23       50  E9 0040E         BLBC    R0, 5?$                               ;
                     01     A8  40   8F  88 00411         BISB2   #64, UAF$GQ_SYSUAFF+1                 ; 0879
                                     5B  DD 00416         PUSHL   R11                                   ; 0880
                              A0     A6  9F 00418         PUSHAB  SD_ENQLM                              ;
                000000000G  00       02  FB 0041B         CALLS   #2, CLI$GET_VALUE                     ;
                                     10  DD 00422         PUSHL   #16                                   ; 0881
                              42     AA  9F 00424         PUSHAB  RECBUF+534                            ;
                00000000V   00       02  FB 00427         CALLS   #2, GETVAL                            ;
                            77       50  E9 0042E  49$:   BLBC    R0, 54$                               ;
                            67       01  D0 00431         MOVL    #1, STATUS                            ; 0883
                            01CC     C6  9F 00434  50$:   PUSHAB  SD_EXPIRATION                         ; 0886
                            69       01  FB 00438         CALLS   #1, CLI$PRESENT                       ;
                     04     A7       50  D0 0043B         MOVL    R0, CLI_STATUS                        ;
                000000000G  8F       50  D1 0043F         CMPL    R0, #CLI$_NEGATED                     ; 0887
                                     0E  12 00446         BNEQ    51$                                   ;
                     01     A8  80   8F  88 00448         BISB2   #128, UAF$GQ_SYSUAFF+1                ; 0890
   08            00         6E       00  2C 0044D         MOVC5   #0, (SP), #0, #8, RECBUF+364          ; 0891
                              98     AA     00452                                                       ;
                                     2A  11 00454         BRB     52$                                   ; 0892
                            2A       50  E9 00456  51$:   BLBC    R0, 53$                               ; 0895
                     01     A8  80   8F  88 00459         BISB2   #128, UAF$GQ_SYSUAFF+1                ; 0898
                                     5B  DD 0045E         PUSHL   R11                                   ; 0899
                            01CC     C6  9F 00460         PUSHAB  SD_EXPIRATION                         ;
                000000000G  00       02  FB 00464         CALLS   #2, CLI$GET_VALUE                     ;
                              98     AA  9F 0046B         PUSHAB  RECBUF+364                            ; 0900
                                     5B  DD 0046E         PUSHL   R11                                   ;
                000000000G  00       02  FB 00470         CALLS   #2, LIB$CVT_TIME                      ;
                            67       50  D0 00477         MOVL    R0, STATUS                            ;
                            03       50  E8 0047A         BLBS    R0, 52$                               ;
                                   0364  31 0047D         BRW     92$                                   ;
                            67       01  D0 00480  52$:   MOVL    #1, STATUS                            ; 0906
                              B0     A6  9F 00483  53$:   PUSHAB  SD_FILLM                              ; 0909
                            69       01  FB 00486         CALLS   #1, CLI$PRESENT                       ;
                            22       50  E9 00489         BLBC    R0, 55$                               ;
                     02     A8       01  88 0048C         BISB2   #1, UAF$GQ_SYSUAFF+2                  ; 0912
                                     5B  DD 00490         PUSHL   R11                                   ; 0913
                              B0     A6  9F 00492         PUSHAB  SD_FILLM                              ;
```

```
                00000000G  00          02  FB  00495          CALLS   #2, CLI$GET_VALUE
                                       10  DD  0049C          PUSHL   #16                                  ;0914
                                    44 AA  9F  0049E          PUSHAB  RECBUF+536
                00000000V  00          02  FB  004A1          CALLS   #2, GETVAL
                                       74  50  E9  004A8 54$:  BLBC    R0, 58$
                                       67  01  D0  004AB      MOVL    #1, STATUS                          ;0916
                                    C0 AA  9F  004AE 55$:     PUSHAB  SD_FLAGS                            ;0919
                                       69  01  FB  004B1      CALLS   #1, CLI$PRESENT
                                       11  50  E9  004B4      BLBC    R0, 56$
                              02       A8  02  88  004B7      BISB2   #2, UAF$GQ_SYSUAFF+2                 ;0922
                00000000V  00          00  FB  004BB          CALLS   #0, GETFLAGS                        ;0923
                                       5A  50  E9  004C2      BLBC    R0, 58$
                                       67  01  D0  004C5      MOVL    #1, STATUS                          ;0925
                                  0170 C6  9F  004C8 56$:     PUSHAB  SD_JTQUOTA                          ;0928
                                       69  01  FB  004CC      CALLS   #1, CLI$PRESENT
                                       23  50  E9  004CF      BLBC    R0, 57$
                                 02    A8  08  88  004D2      BISB2   #8, UAF$GQ_SYSUAFF+2                 ;0931
                                       5B  DD  004D6          PUSHL   R11                                 ;0932
                                  0170 C6  9F  004D8          PUSHAB  SD_JTQUOTA
                00000000G  00          02  FB  004DC          CALLS   #2, CLI$GET_VALUE
                                       20  DD  004E3          PUSHL   #32                                 ;0933
                                    64 AA  9F  004E5          PUSHAB  RECBUF+568
                00000000V  00          02  FB  004E8          CALLS   #2, GETVAL
                                       59  50  E9  004EF      BLBC    R0, 60$
                                       67  01  D0  004F2      MOVL    #1, STATUS                          ;0935
                                    D0 A6  9F  004F5 57$:     PUSHAB  SD_LGICMD                           ;0938
                                       69  01  FB  004F8      CALLS   #1, CLI$PRESENT
                                       27  50  E9  004FB      BLBC    R0, 59$
                                 02    A8  10  88  004FE      BISB2   #16, UAF$GQ_SYSUAFF+2               ;0941
                                       5B  DD  00502          PUSHL   R11                                 ;0942
                                    D0 A6  9F  00504          PUSHAB  SD_LGICMD
                00000000G  00          02  FB  00507          CALLS   #2, CLI$GET_VALUE
                                       7E  01  DD  0050E      PUSHL   #1                                  ;0943
                                 40    8F  9A  00510          MOVZBL  #64, -(SP)
                              FF00     CA  9F  00514          PUSHAB  RECBUF+212
                00000000V  00          03  FB  00518          CALLS   #3, GETSTRING
                                       56  50  E9  0051F 58$:  BLBC    R0, 62$
                                       67  01  D0  00522      MOVL    #1, STATUS                          ;0945
                                    E0 A6  9F  00525 59$:     PUSHAB  SD_MAXJOBS                          ;0948
                                       69  01  FB  00528      CALLS   #1, CLI$PRESENT
                                       23  50  E9  0052B      BLBC    R0, 61$
                              02       A8  80  8F  88  0052E  BISB2   #128, UAF$GQ_SYSUAFF+2               ;0951
                                       5B  DD  00533          PUSHL   R11                                 ;0952
                                    E0 A6  9F  00535          PUSHAB  SD_MAXJOBS
                0000000UG  00          02  FB  00538          CALLS   #2, CLI$GET_VALUE
                                       10  DD  0053F          PUSHL   #16                                 ;0953
                                    32 AA  9F  00541          PUSHAB  RECBUF+518
                00000000V  00          02  FB  00544          CALLS   #2, GETVAL
                                       58  50  E9  0054B 60$:  BLBC    R0, 64$
                                       67  01  D0  0054E      MOVL    #1, STATUS                          ;0955
                                  0138 C6  9F  00551 61$:     PUSHAB  SD_MAXACCTJOBS                      ;0958
                                       69  01  FB  00555      CALLS   #1, CLI$PRESENT
                                       23  50  E9  00558      BLBC    R0, 63$
                                 03    A8  01  88  0055B      BISB2   #1, UAF$GQ_SYSUAFF+3                 ;0961
                                       5B  DD  0055F          PUSHL   R11                                 ;0962
                                  0138 C6  9F  00561          PUSHAB  SD_MAXACCTJOBS
                00000000G  00          02  FB  00565          CALLS   #2, CLI$GET_VALUE
```

```
                              10  DD 0056C        PUSHL   #16                               ; 0963
                      34      AA  9F 0056E        PUSHAB  RECBUF+520                        ;
        00000000V  00         02  FB 00571        CALLS   #2, GETVAL                        ;
                   59         50  E9 00578 62$:   BLBC    R0, 66$                           ;
                   67         01  D0 0057B        MOVL    #1, STATUS                        ; 0965
                    014C      C6  9F 0057E 63$:   PUSHAB  SD_MAXDETACH                      ; 0968
                   69         01  FB 00582        CALLS   #1, CLI$PRESENT                   ;
                   24         50  E9 00585        BLBC    R0, 65$                           ;
              02   A8  40     8F  88 00588        BISB2   #64, UAF$GQ_SYSUAFF+2             ; 0971
                              5B  DD 0058D        PUSHL   R11                               ; 0972
                    014C      C6  9F 0058F        PUSHAB  SD_MAXDETACH                      ;
        00000000G  00         02  FB 00593        CALLS   #2, CLI$GET_VALUE                 ;
                              10  DD 0059A        PUSHL   #16                               ; 0973
                      36      AA  9F 0059C        PUSHAB  RECBUF+522                        ;
        00000000V  00         02  FB 0059F        CALLS   #2, GETVAL                        ;
                   56         50  E9 005A6 64$:   BLBC    R0, 68$                           ;
                   67         01  D0 005A9        MOVL    #1, STATUS                        ; 0975
                      F0      A6  9F 005AC 65$:   PUSHAB  SD_OWNER                          ; 0978
                   69         01  FB 005AF        CALLS   #1, CLI$PRESENT                   ;
                   25         50  E9 005B2        BLBC    R0, 67$                           ;
              03   A8         04  88 005B5        BISB2   #4, UAF$GQ_SYSUAFF+3              ; 0981
                              5B  DD 005B9        PUSHL   R11                               ; 0982
                      F0      A6  9F 005BB        PUSHAB  SD_OWNER                          ;
        00000000G  00         02  FB 005BE        CALLS   #2, CLI$GET_VALUE                 ;
                              01  DD 005C5        PUSHL   #1                                ; 0983
                              20  DD 005C7        PUSHL   #32                               ;
                    FE80      CA  9F 005C9        PUSHAB  RECBUF+84                         ;
        00000000V  00         03  FB 005CD        CALLS   #3, GETSTRING                     ;
                   72         50  E9 005D4 66$:   BLBC    R0, 71$                           ;
                   67         01  D0 005D7        MOVL    #1, STATUS                        ; 0985
                      10      A6  9F 005DA 67$:   PUSHAB  SD_PBYTLM                         ; 0988
                   69         01  FB 005DD        CALLS   #1, CLI$PRESENT                   ;
                   22         50  E9 005E0        BLBC    R0, 69$                           ;
              03   A8         10  88 005E3        BISB2   #16, UAF$GQ_SYSUAFF+3             ; 0991
                              5B  DD 005E7        PUSHL   R11                               ; 0992
                      10      A6  9F 005E9        PUSHAB  SD_PBYTLM                         ;
        00000000G  00         02  FB 005EC        CALLS   #2, CLI$GET_VALUE                 ;
                              20  DD 005F3        PUSHL   #32                               ; 0993
                      60      AA  9F 005F5        PUSHAB  RECBUF+564                        ;
        00000000V  00         02  FB 005F8        CALLS   #2, GETVAL                        ;
                   6A         50  E9 005FF 68$:   BLBC    R0, 73$                           ;
                   67         01  D0 00602        MOVL    #1, STATUS                        ; 0995
                      34      A6  9F 00605 69$:   PUSHAB  SD_PGFLQUOTA                      ; 0998
                   69         01  FB 00608        CALLS   #1, CLI$PRESENT                   ;
                   23         50  E9 0060B        BLBC    R0, 70$                           ;
              03   A8  80     8F  88 0060E        BISB2   #128, UAF$GQ_SYSUAFF+3            ; 1001
                              5B  DD 00613        PUSHL   R11                               ; 1002
                      34      A6  9F 00615        PUSHAB  SD_PGFLQUOTA                      ;
        00000000G  00         02  FB 00618        CALLS   #2, CLI$GET_VALUE                 ;
                              18  DD 0061F        PUSHL   #24                               ; 1003
                      54      AA  9F 00621        PUSHAB  RECBUF+552                        ;
        00000000V  00         02  FB 00624        CALLS   #2, GETVAL                        ;
                   7E         50  E9 0062B        BLBC    R0, 76$                           ;
                   67         01  D0 0062F        MOVL    #1, STATUS                        ; 1005
                      48      A6  9F 00631 70$:   PUSHAB  SD_P_RESTRICT                     ; 1008
                   69         01  FB 00634        CALLS   #1, CLI$PRESENT                   ;
                   15         50  E9 00637        BLBC    R0, 72$                           ;
```

```
                03   A8        40   8F   88 0063A             BISB2   #64, UAF$GQ_SYSUAFF+3                    ; 1011
                          48   A6   9F   0063F                 PUSHAB  SD_P_RESTRICT                            ; 1012
          00000000V   00        01   FB   00642               CALLS   #1, GETRESTRICT
                      60        50   E9   00649 71$:           BLBC    R0, 76$
                      67        01   D0   0064C               MOVL    #1, STATUS                               ; 1014
                          00A0  C6   9F   0064F 72$:           PUSHAB  SD_S_RESTRICT                            ; 1017
                      69        01   FB   00653               CALLS   #1, CLI$PRESENT
                      19        50   E9   00656               BLBC    R0, 74$
                05    A8        02   88   00659               BISB2   #2, UAF$GQ_SYSUAFF+5                     ; 1020
                4C    A7        01   D0   0065D               MOVL    #1, SECONDARY                            ; 1021
                          00A0  C6   9F   00661               PUSHAB  SD_S_RESTRICT                            ; 1022
          00000000V   00        01   FB   00665               CALLS   #1, GETRESTRICT
                      68        50   E9   0066C 73$:           BLBC    R0, 78$
                      67        01   D0   0066F               MOVL    #1, STATUS                               ; 1024
                          20    A6   9F   00672 74$:           PUSHAB  SD_PFLAGS                                ; 1027
                      69        01   FB   00675               CALLS   #1, CLI$PRESENT
                      18        50   E9   00678               BLBC    R0, 75$
                03    A8        20   88   0067B               BISB2   #32, UAF$GQ_SYSUAFF+3                    ; 1030
                48    A7        01   D0   0067F               MOVL    #1, PRIMARY                              ; 1031
                          20    A6   9F   00683               PUSHAB  SD_PFLAGS                                ; 1032
          00000000V   00        01   FB   00686               CALLS   #1, GETPFLAGS
                      62        50   E9   0068D               BLBC    R0, 80$
                      67        01   D0   00690               MOVL    #1, STATUS                               ; 1034
                          008C  C6   9F   00693 75$:           PUSHAB  SD_SFLAGS                                ; 1037
                      69        01   FB   00697               CALLS   #1, CLI$PRESENT
                      15        50   E9   0069A               BLBC    R0, 77$
                05    A8        01   88   0069D               BISB2   #1, UAF$GQ_SYSUAFF+5                     ; 1040
                          008C  C6   9F   006A1               PUSHAB  SD_SFLAGS                                ; 1041
          00000000V   00        01   FB   006A5               CALLS   #1, GETPFLAGS
                      6E        50   E9   006AC 76$:           BLBC    R0, 82$
                      67        01   D0   006AF               MOVL    #1, STATUS                               ; 1043
                          58    A6   9F   006B2 77$:           PUSHAB  SD_PRCLM                                 ; 1046
                      69        01   FB   006B5               CALLS   #1, CLI$PRESENT
                      22        50   E9   006B8               BLBC    R0, 79$
                04    A8        01   88   006BB               BISB2   #1, UAF$GQ_SYSUAFF+4                     ; 1049
                                  5B   DD   006BF             PUSHL   R11                                      ; 1050
                          58    A6   9F   006C1               PUSHAB  SD_PRCLM
          00000000G   00        02   FB   006C4               CALLS   #2, CLI$GET_VALUE
                                  10   DD   006CB             PUSHL   #16                                      ; 1051
                          38    AA   9F   006CD               PUSHAB  RECBUF+524
          00000000V   00        02   FB   006D0               CALLS   #2, GETVAL
                      63        50   E9   006D7 78$:           BLBC    R0, 84$
                      67        01   D0   006DA               MOVL    #1, STATUS                               ; 1053
                        0124    C6   9F   006DD 79$:           PUSHAB  SD_PRIMEDAYS                             ; 1056
                      69        01   FB   006E1               CALLS   #1, CLI$PRESENT
                      11        50   E9   006E4               BLBC    R0, 81$
                04    A8        02   88   006E7               BISB2   #2, UAF$GQ_SYSUAFF+4                     ; 1059
          00000000V   00        00   FB   006EB               CALLS   #0, GETPRIMEDAYS                         ; 1060
                      48        50   E9   006F2 80$:           BLBC    R0, 84$
                      67        01   D0   006F5               MOVL    #1, STATUS                               ; 1062
                          68    A6   9F   006F8 81$:           PUSHAB  SD_PRIORITY                              ; 1065
                      69        01   FB   006FB               CALLS   #1, CLI$PRESENT
                      22        50   E9   006FE               BLBC    R0, 83$
                04    A8        04   88   00701               BISB2   #4, UAF$GQ_SYSUAFF+4                     ; 1068
                                  5B   DD   00705             PUSHL   R11                                      ; 1069
                          68    A6   9F   00707               PUSHAB  SD_PRIORITY
          00000000G   00        02   FB   0070A               CALLS   #2, CLI$GET_VALUE
```

```
                                    08  DD 00711          PUSHL   #8                                                  ; 1070
                                30  AA  9F 00713          PUSHAB  RECBUF+516
                00000000V       00      02 FB 00716      CALLS   #2, GETVAL
                                1D      50 E9 0071D 82$:  BLBC    R0, 84$
                                67      01 D0 00720       MOVL    #1, STATUS                                          ; 1072
                                7C  A6  9F 00723 83$:     PUSHAB  SD_PRIVILEGES                                       ; 1075
                                69      01 FB 00726       CALLS   #1, CLI$PRESENT
                                1A      50 E9 00729       BLBC    R0, 86$
                            04  A8      08 88 0072C       BISB2   #8, UAF$GQ_SYSUAFF+4                                ; 1078
                                C8  AA  9F 00730          PUSHAB  RECBUF+412                                          ; 1079
                                7C  A6  9F 00733          PUSHAB  SD_PRIVILEGES
                00000000V       00      02 FB 00736       CALLS   #2, GETPRIV
                                03      50 E8 00739 84$:  BLBS    R0, 85$
                                    023F 31 00740         BRW     109$
                                67      01 D0 00743 85$:  MOVL    #1, STATUS                                          ; 1081
                                03BC C6  9F 00746 86$:    PUSHAB  P.AFE                                               ; 1084
                                69      01 FB 0074A       CALLS   #1, CLI$PRESENT
                            04  A7      50 D0 0074D       MOVL    R0, CLI_STATUS
                00000000G   8F          50 D1 00751       CMPL    R0, #CLI$_NEGATED                                   ; 1085
                                        0C 12 00758       BNEQ    87$
                            04  A8      10 88 0075A       BISB2   #16, UAF$GQ_SYSUAFF+4                               ; 1088
                                50  A8  AA 9E 0075E       MOVAB   RECBUF+380, TIME                                    ; 1089
                                60      7C 00762          CLRQ    (TIME)                                              ; 1090
                                        12 11 00764       BRB     88$                                                 ; 1091
                                1A      50 E9 00766 87$:  BLBC    R0, 89$                                             ; 1095
                            04  A8      10 88 00769       BISB2   #16, UAF$GQ_SYSUAFF+4                               ; 1098
                                50  A8  AA 9E 0076D       MOVAB   RECBUF+380, TIME                                    ; 1099
                                60      01 CE 00771       MNEGL   #1, (TIME)                                          ; 1100
                            04  A0      01 CE 00774       MNEGL   #1, 4(TIME)
                            02  A8      02 88 00778 88$:  BISB2   #2, UAF$GQ_SYSUAFF+2                               ; 1101
                            01  AA      02 8A 0077C       BICB2   #2, RECBUF+469                                      ; 1102
                                67      01 D0 00780       MOVL    #1, STATUS                                          ; 1103
                                    0214 C6 9F 00783 89$: PUSHAB  SD_PWDLIFETIME                                      ; 1106
                                69      01 FB 00787       CALLS   #1, CLI$PRESENT
                            04  A7      50 D0 0078A       MOVL    R0, CLI_STATUS
                00000000G   8F          50 D1 0078E       CMPL    R0, #CLI$_NEGATED                                   ; 1107
                                        0D 12 00795       BNEQ    90$
                            04  A8      10 88 00797       BISB2   #16, UAF$GQ_SYSUAFF+4                               ; 1110
08              00          6E          00 2C 0079B       MOVC5   #0, (SP), #0, #8, RECBUF+372                        ; 1111
                                A0  AA      007A0
                                4C  11  007A2             BRB     94$                                                 ; 1112
                                54      50 E9 007A4 90$:  BLBC    R0, 95$                                             ; 1117
                            04  A8      10 88 007A7       BISB2   #16, UAF$GQ_SYSUAFF+4                               ; 1120
                                5B      DD 007AB          PUSHL   R11                                                 ; 1121
                                    0214 C6 9F 007AD      PUSHAB  SD_PWDLIFETIME
                00000000G       00      02 FB 007B1       CALLS   #2, CLI$GET_VALUE
                                50  04  AB D0 007B8       MOVL    TOKENDSC+4, R0                                      ; 1122
04              00              60      6B 2D 007BC       CMPC5   TOKENDSC, (R0), #0, #4, P.AFG
                                    03C4 C6      007C1
                                        0C 12 007C4       BNEQ    91$
08              00              6E          00 2C 007C6   MOVC5   #0, (SP), #0, #8, RECBUF+372                        ; 1125
                                A0  AA      007CB
                                67      01 D0 007CD       MOVL    #1, STATUS                                          ; 1126
                                        1E 11 007D0       BRB     94$                                                 ; 1122
                                A0  AA  9F 007D2 91$:     PUSHAB  RECBUF+372                                          ; 1129
                                5B      DD 007D5          PUSHL   R11
                00000000G       00      02 FB 007D7       CALLS   #2, LIB$CVT_DTIME
```

```
                      67                  50  D0 007DE           MOVL    R0, STATUS                         :
                      0C                  50  E8 007E1           BLBS    R0, 94$                            :
                                          67  DD 007E4 92$:      PUSHL   STATUS                             : 1132
          00000000G   00                  01  FB 007E6           CALLS   #1, LIB$SIGNAL                     :
                                     0192  31 007ED 93$:         BRW     109$                               : 1133
              02      A8                  02  88 007F0 94$:      BISB2   #2, UAF$GQ_SYSUAFF+2               : 1135
              01      AA                  02  8A 007F4           BICB2   #2, RECBUF+469                     : 1136
                      67                  01  D0 007F8           MOVL    #1, STATUS                         : 1137
                               0228   C6  9F 007FB 95$:          PUSHAB  SD_PWDMINIMUM                      : 1140
                      69                  01  FB 007FF           CALLS   #1, CLI$PRESENT                    :
                      23                  50  E9 00802           BLBC    R0, 96$                            :
              04      A8                  20  88 00805           BISB2   #32, UAF$GQ_SYSUAFF+4              : 1143
                                          5B  DD 00809           PUSHL   R11                                : 1144
                               0228   C6  9F 0080B               PUSHAB  SD_PWDMINIMUM                      :
          00000000G   00                  02  FB 0080F           CALLS   #2, CLI$GET_VALUE                  :
                                          08  DD 00816           PUSHL   #8                                 : 1145
                                 96   AA  9F 00818               PUSHAB  RECBUF+362                         :
          00000000V   00                  02  FB 0081B           CALLS   #2, GETVAL                         :
                      C8                  50  E9 00822           BLBC    R0, 93$                            :
                      67                  01  D0 00825           MOVL    #1, STATUS                         : 1147
                               023C   C6  9F 00828 96$:          PUSHAB  SD_QUEPRIORITY                     : 1150
                      69                  01  FB 0082C           CALLS   #1, CLI$PRESENT                    :
                      24                  50  E9 0082F           BLBC    R0, 97$                            :
              04      A8         40       8F  88 00832           BISB2   #64, UAF$GQ_SYSUAFF+4              : 1153
                                          5B  DD 00837           PUSHL   R11                                : 1154
                               023C   C6  9F 00839               PUSHAB  SD_QUEPRIORITY                     :
          00000000G   00                  02  FB 0083D           CALLS   #2, CLI$GET_VALUE                  :
                                          08  DD 00844           PUSHL   #8                                 : 1155
                                 31   AA  9F 00846               PUSHAB  RECBUF+517                         :
          00000000V   00                  02  FB 00849           CALLS   #2, GETVAL                         :
                      9A                  50  E9 00850           BLBC    R0, 93$                            :
                      67                  01  D0 00853           MOVL    #1, STATUS                         : 1157
                               0080   C6  9F 00856 97$:          PUSHAB  SD_SHRFILLM                        : 1160
                      69                  01  FB 0085A           CALLS   #1, CLI$PRESENT                    :
                      23                  50  E9 0085D           BLBC    R0, 98$                            :
              05      A8                  04  88 00860           BISB2   #4, UAF$GQ_SYSUAFF+5               : 1163
                                          5B  DD 00864           PUSHL   R11                                : 1164
                               0080   C6  9F 00866               PUSHAB  SD_SHRFILLM                        :
          00000000G   00                  02  FB 0086A           CALLS   #2, CLI$GET_VALUE                  :
                                          10  DD 00871           PUSHL   #16                                : 1165
                                 46   AA  9F 00873               PUSHAB  RECBUF+538                         :
          00000000V   00                  02  FB 00876           CALLS   #2, GETVAL                         :
                      7F                  50  E9 0087D           BLBC    R0, 101$                           :
                      67                  01  D0 00880           MOVL    #1, STATUS                         : 1167
                               00C0   C6  9F 00883 98$:          PUSHAB  SD_TQELM                           : 1170
                      69                  01  FB 00887           CALLS   #1, CLI$PRESENT                    :
                      23                  50  E9 0088A           BLBC    R0, 99$                            :
              05      A8                  08  88 0088D           BISB2   #8, UAF$GQ_SYSUAFF+5               : 1173
                                          5B  DD 00891           PUSHL   R11                                : 1174
                               00C0   C6  9F 00893               PUSHAB  SD_TQELM                           :
          00000000G   00                  02  FB 00897           CALLS   #2, CLI$GET_VALUE                  :
                                          10  DD 0089E           PUSHL   #16                                : 1175
                                 3E   AA  9F 008A0               PUSHAB  RECBUF+530                         :
          00000000V   00                  02  FB 008A3           CALLS   #2, GETVAL                         :
                      52                  50  E9 008AA           BLBC    R0, 101$                           :
                      67                  01  D0 008AD           MOVL    #1, STATUS                         : 1177
                               00CC   C6  9F 008B0 99$:          PUSHAB  SD_UIC                             : 1180
```

```
                69              01 FB 008B4             CALLS   #1, CLI$PRESENT                                  ;
                1E              50 E9 008B7             BLBC    R0, 100$                                        ;
              05 A8             10 88 008BA             BISB2   #16, UAF$GQ_SYSUAFF+5                           ; 1183
                                5B DD 008BE             PUSHL   R11                                             ; 1184
                          00CC  C6 9F 008C0             PUSHAB  SD_UIC
  00000000G     00              02 FB 008C4             CALLS   #2, CLI$GET_VALUE
  00000000V     00              00 FB 008CB             CALLS   #0, GETUIC                                      ; 1185
                5B              50 E9 008D2             BLBC    R0, 103$                                        ;
                67              01 D0 008D5             MOVL    #1, STATUS                                      ; 1187
                          00E0  C6 9F 008D8 100$:       PUSHAB  SD_WSDEFAULT                                    ; 1190
                69              01 FB 008DC             CALLS   #1, CLI$PRESENT
                26              50 E9 008DF             BLBC    R0, 102$                                        ;
              05 A8             20 88 008E2             BISB2   #32, UAF$GQ_SYSUAFF+5                           ; 1193
                                5B DD 008E6             PUSHL   R11                                             ; 1194
                          00E0  C6 9F 008E8             PUSHAB  SD_WSDEFAULT
  00000000G     00              02 FB 008EC             CALLS   #2, CLI$GET_VALUE
                                11 DD 008F3             PUSHL   #17                                             ; 1195
                            4C  AA 9F 008F5             PUSHAB  RECBUF+544
  00000000V     00              02 FB 008F8             CALLS   #2, GETVAL
                5F              50 E9 008FF 101$:       BLBC    R0, 105$                                        ;
                            4E  AA B4 00902             CLRW    RECBUF+546                                      ; 1197
                67              01 D0 00905             MOVL    #1, STATUS                                      ; 1198
                          00F0  C6 9F 00908 102$:       PUSHAB  SD_WSEXTENT                                     ; 1201
                69              01 FB 0090C             CALLS   #1, CLI$PRESENT
                27              50 E9 0090F             BLBC    R0, 104$                                        ;
              05 A8         40  8F 88 00912             BISB2   #64, UAF$GQ_SYSUAFF+5                           ; 1204
                                5B DD 00917             PUSHL   R11                                             ; 1205
                          00F0  C6 9F 00919             PUSHAB  SD_WSEXTENT
  00000000G     00              02 FB 0091D             CALLS   #2, CLI$GET_VALUE
                                11 DD 00924             PUSHL   #17                                             ; 1206
                            50  AA 9F 00926             PUSHAB  RECBUF+548
  00000000V     00              02 FB 00929             CALLS   #2, GETVAL
                4F              50 E9 00930 103$:       BLBC    R0, 109$                                        ;
                            52  AA B4 00933             CLR.    RECBUF+550                                      ; 1208
                67              01 D0 00936             MOVL    #1, STATUS                                      ; 1209
                        0100    C6 9F 00939 104$:       PUSHAB  SD_WSQUOTA                                      ; 1212
                69              01 FB 0093D             CALLS   #1, CLI$PRESENT
                27              50 E9 00940             BLBC    R0, 106$                                        ;
              05 A8         80  8F 88 00943             BISB2   #128, UAF$GQ_SYSUAFF+5                          ; 1215
                                5B DD 00948             PUSHL   R11                                             ; 1216
                        0100    C6 9F 0094A             PUSHAB  SD_WSQUOTA
  00000000G     00              02 FB 0094E             CALLS   #2, CLI$GET_VALUE
                                11 DD 00955             PUSHL   #17                                             ; 1217
                            48  AA 9F 00957             PUSHAB  RECBUF+540
  00000000V     00              02 FB 0095A             CALLS   #2, GETVAL
                1E              50 E9 00961 105$:       BLBC    R0, 109$                                        ;
                            4A  AA B4 00964             CLRW    RECBUF+542                                      ; 1219
                67              01 D0 00967             MOVL    #1, STATUS                                      ; 1220
08 00000000G    00              02 E0 0096A 106$:       BBS     #2, UAF$GL_CTLMSK, 107$                         ; 1223
04 00000000G    00              01 E1 00972             BBC     #1, UAF$GL_CTLMSK, 108$
                50              01 D0 0097A 107$:       MOVL    #1, R0                                          ; 1225
                                04 0097D                RET
                50              67 D0 0097E 108$:       MOVL    STATUS, R0                                      ; 1227
                                04 00981                RET
                                50 D4 00982 109$:       CLRL    R0                                              ; 1228
                                04 00984                RET
```

; Routine Size: 2437 bytes, Routine Base: $CODE$ + 0000

```
;1064     1229  1 %sbttl 'getaccess - get hourly and daily access control flags'
;1065     1230  1 routine getaccess (qual_name, access_type, qual_status) =
;1066     1231  2 begin
;1067     1232  2
;1068     1233  2 !++
;1069     1234  2 !
;1070     1235  2 ! FUNCTIONAL DESCRIPTION:
;1071     1236  2 !
;1072     1237  2 !       Parse the value list for hourly login restrictions on the
;1073     1238  2 !       selected access modes.
;1074     1239  2 !
;1075     1240  2 ! INPUTS:
;1076     1241  2 !
;1077     1242  2 !       qual_name - descriptor of qualifier name to use
;1078     1243  2 !       access_type - bitvector representing the access modes to be affected
;1079     1244  2 !       qual_status - status from cli$present for qualifier
;1080     1245  2 !
;1081     1246  2 ! IMPLICIT INPUTS:
;1082     1247  2 !
;1083     1248  2 !       none
;1084     1249  2 !
;1085     1250  2 ! OUTPUTS:
;1086     1251  2 !
;1087     1252  2 !       none
;1088     1253  2 !
;1089     1254  2 ! IMPLICIT OUTPUTS:
;1090     1255  2 !
;1091     1256  2 !       none
;1092     1257  2 !
;1093     1258  2 ! ROUTINE VALUE:
;1094     1259  2 !
;1095     1260  2 !       true -> all keywords and parameters were specified correctly
;1096     1261  2 !
;1097     1262  2 ! SIDE EFFECTS:
;1098     1263  2 !
;1099     1264  2 !       none
;1100     1265  2 !--
;1101     1266  2
;1102     1267  2 map
;1103     1268  2         access_type     : bitvector;
;1104     1269  2
;1105     1270  2 STRUCTURE
;1106     1271  2     threebytevector [i; n, ext=0] =
;1107     1272  2         [n*3]
;1108     1273  2         (threebytevector+i*3)<0, 24, ext>;
;1109     1274  2
;1110     1275  2 $ASSUME ($BYTEOFFSET (uaf$b_network_access_s), EQL, $BYTEOFFSET (uaf$b_network_access_p)+3);
;1111     1276  2 $ASSUME ($BYTEOFFSET (uaf$b_batch_access_p), EQL, $BYTEOFFSET (uaf$b_network_access_s)+3);
;1112     1277  2 $ASSUME ($BYTEOFFSET (uaf$b_batch_access_s), EQL, $BYTEOFFSET (uaf$b_batch_access_p)+3);
;1113     1278  2 $ASSUME ($BYTEOFFSET (uaf$b_local_access_p), EQL, $BYTEOFFSET (uaf$b_batch_access_s)+3);
;1114     1279  2 $ASSUME ($BYTEOFFSET (uaf$b_local_access_s), EQL, $BYTEOFFSET (uaf$b_local_access_p)+3);
;1115     1280  2 $ASSUME ($BYTEOFFSET (uaf$b_dialup_access_p), EQL, $BYTEOFFSET (uaf$b_local_access_s)+3);
;1116     1281  2 $ASSUME ($BYTEOFFSET (uaf$b_dialup_access_s), EQL, $BYTEOFFSET (uaf$b_dialup_access_p)+3);
;1117     1282  2 $ASSUME ($BYTEOFFSET (uaf$b_remote_access_p), EQL, $BYTEOFFSET (uaf$b_dialup_access_s)+3);
;1118     1283  2 $ASSUME ($BYTEOFFSET (uaf$b_remote_access_s), EQL, $BYTEOFFSET (uaf$b_remote_access_p)+3);
;1119     1284  2
;1120     1285  2 bind
```

```
;  1121          1286  2          access_vector   = recbuf[uaf$b_network_access_p]
;  1122          1287  2                          : threebytevector;
;  1123          1288  2
;  1124          1289  2 !
;  1125          1290  2 ! Initialize output values.
;  1126          1291  2 !
;  1127          1292  2 primary_access = -1;
;  1128          1293  2 secondary_access = -1;
;  1129          1294  2 if not .qual_status then no_flag = true;
;  1130          1295  2
;  1131          1296  2 !
;  1132          1297  2 ! Fetch value list items and parse them.
;  1133          1298  2 !
;  1134          1299  2 while true
;  1135          1300  2 do
;  1136          1301  3     begin
;  1137          1302  3     if not cli$get_value (.qual_name, tokendsc)
;  1138          1303  3     then exitloop;
;  1139          1304  3
;  1140          1305  3     tparse_block[tpa$l_stringcnt] = .tokenlen;
;  1141          1306  3     tparse_block[tpa$l_stringptr] = .tokenptr;
;  1142          1307  3     if not lib$tparse (tparse_block, access_states, access_keys)
;  1143          1308  3     then return LIB$SIGNAL(UAF$_BADVALUE, 2, .tokenlen, .tokenptr);
;  1144          1309  2     end;
;  1145          1310  2
;  1146          1311  2 !
;  1147          1312  2 ! Default omitted values.
;  1148          1313  2 !
;  1149          1314  2 if not .primary and not .secondary
;  1150          1315  2 then secondary_access = .primary_access;
;  1151          1316  2 if .primary and not .secondary
;  1152          1317  2 then secondary_access = 0;
;  1153          1318  2 if .secondary and not .primary and .primary_access eql -1
;  1154          1319  2 then primary_access = 0;
;  1155          1320  2
;  1156          1321  2 if not .no_flag
;  1157          1322  2 then
;  1158          1323  3     begin
;  1159          1324  3     primary_access = not .primary_access;
;  1160          1325  3     secondary_access = not .secondary_access;
;  1161          1326  2     end;
;  1162          1327  2
;  1163          1328  2 !
;  1164          1329  2 ! Write the specified fields in the UAF record.
;  1165          1330  2 !
;  1166          1331  2 incr j from 0 to 4
;  1167          1332  2 do
;  1168          1333  3     begin
;  1169          1334  3     if .access_type[.j]
;  1170          1335  3     then
;  1171          1336  4         begin
;  1172          1337  4         access_vector [.j*2+0] = .primary_access;
;  1173          1338  4         access_vector [.j*2+1] = .secondary_access;
;  1174          1339  3         end;
;  1175          1340  2     end;
;  1176          1341  2
;  1177          1342  2 true
```

```
; 1178          1343  1 end;


                                    003C 00000 GETACCESS:
                                                        .WORD   Save R2,R3,R4,R5                        ; 1230
                        55 00000000G   00  9E 00002     MOVAB   TOKENPTR, R5                            ;
                        54 00000000G   00  9E 00009     MOVAB   TOKENLEN, R4                            ;
                        53 00000000'   00  9E 00010     MOVAB   PRIMARY_ACCESS, R3                      ;
                        63             01  CE 00017     MNEGL   #1, PRIMARY_ACCESS                      ; 1292
                     04 A3             01  CE 0001A     MNEGL   #1, SECONDARY_ACCESS                    ; 1293
                     04        0C      AC  E8 0001E     BLBS    QUAL_STATUS, T$                         ; 1294
                     08 A3             01  D0 00022     MOVL    #1, NO_FLAG                             ;
                           00000000G   00  9F 00026 1$:  PUSHAB  TOKENDSC                                ; 1302
                                 04    AC  DD 0002C     PUSHL   QUAL_NAME                               ;
                00000000G  00          02  FB 0002F     CALLS   #2, CLI$GET_VALUE                       ;
                           36          50  E9 00036     BLBC    R0, 2$                                  ;
                     D4 A3             64  3C 00039     MOVZWL  TOKENLEN, TPARSE_BLOCK+8                ; 1305
                     D8 A3             65  D0 0003D     MOVL    TOKENPTR, TPARSE_BLOCK+12               ; 1306
                           00000000'   00  9F 00041     PUSHAB  ACCESS_KEYS                             ; 1307
                           00000000'   00  9F 00047     PUSHAB  ACCESS_STATES                           ;
                                 CC    A3  9F 0004D     PUSHAB  TPARSE_BLOCK                            ;
                00000000G  00          03  FB 00050     CALLS   #3, LIB$TPARSE                          ;
                           CC          50  E8 00057     BLBS    R0, 1$                                  ;
                                       65  DD 0005A     PUSHL   TOKENPTR                                ; 1308
                           7E          64  3C 0005C     MOVZWL  TOKENLEN, -(SP)                         ;
                                       02  DD 0005F     PUSHL   #2                                      ;
                           00000000G   8F  DD 00061     PUSHL   #UAF$_BADVALUE                          ;
                00000000G  00          04  FB 00067     CALLS   #4  LIB$SIGNAL                          ;
                                       04  00006E      RET                                             ;
                           50        0C  A3  D0 0006F 2$:  MOVL    PRIMARY, R0                         ; 1314
                           0B          50  E8 00073     BLBS    R0, 4$                                  ;
                           04        10  A3  E8 00076     BLBS    SECONDARY, 3$                           ;
                                 04    A3  63  D0 0007A     MOVL    PRIMARY_ACCESS, SECONDARY_ACCESS        ; 1315
                           07          50  E9 0007E 3$:  BLBC    R0, 5$                                  ; 1316
                           07        10  A3  E8 00081 4$:  BLBS    SECONDARY, 6$                           ;
                                     04  A3  D4 00085     CLRL    SECONDARY_ACCESS                        ; 1317
                           0E        10  A3  E9 00088 5$:  BLBC    SECONDARY, 7$                           ; 1318
                           0B          50  E8 0008C 6$:  BLBS    R0, 7$                                  ;
                     FFFFFFFF      8F  63  D1 0008F     CMPL    PRIMARY_ACCESS, #-1                     ;
                                       02  12 00096     BNEQ    7$                                      ;
                                       63  D4 00098     CLRL    PRIMARY_ACCESS                          ; 1319
                           08        08  A3  E8 0009A 7$:  BLBS    NO_FLAG, 8$                             ; 1321
                                       63  D2 0009E     MCOML   PRIMARY_ACCESS, PRIMARY_ACCESS          ; 1324
                                 04    A3  04  A3  D2 000A1 MCOML   SECONDARY_ACCESS, SECONDARY_ACCESS      ; 1325
                                       50  D4 000A6 8$:  CLRL    J                                       ; 1331
                        21       08  AC      50  E1 000A8 9$:  BBC     J, ACCESS_TYPE, 10$                     ; 1334
                        52          50      01  78 000AD     ASHL    #1, J, R2                               ; 1337
                        51          52      03  C5 000B1     MULL3   #3, R2, R1                              ;
00000000G0041           18          00      63  F0 000B5     INSV    PRIMARY_ACCESS, #0, #24, ACCESS_VECTOR[R1]      ;
                        51          52      03  C5 000BF     MULL3   #3, R2, R1                              ; 1338
00000000G0041           18          00  04  A3  F0 000C3     INSV    SECONDARY_ACCESS, #0, #24, ACCESS_VECTOR+3-     ;
                                                                [R1]
                        D6          50      04  F3 000CE 10$:  AOBLEQ  #4, J, 9$                               ; 1331
                                    50      01  D0 000D2     MOVL    #1, R0                                  ; 1343
```

```
                                                04 000D5          RET                                                 ;

; Routine Size:  214 bytes,    Routine Base:  $CODE$ + 0985
```

```
;  1180          1344  1 %sbttl 'check_primary - check presence of PRIMARY keyword'
;  1181          1345  1 routine check_primary =
;  1182          1346  2 begin
;  1183          1347  2 not .primary and not .secondary
;  1184          1348  1 end;


                                        0000 00000 CHECK_PRIMARY:
                                                           .WORD   Save nothing                                    ;  1345
                 50 00000000'  00 00000000'  00  C9 00002          BISL3   SECONDARY, PRIMARY, R0                  ;  1347
                               50            50  D2 0000E          MCOML   R0, R0                                  ;  1346
                                                 04 00011          RET                                             ;  1348

; Routine Size:  18 bytes,    Routine Base:  $CODE$ + 0A5B
```

```
; 1186          1349  1 %sbttl 'check_secondary - check presence of SECONDARY keyword'
; 1187          1350  1 routine check_secondary =
; 1188          1351  2 begin
; 1189          1352  2 not .secondary
; 1190          1353  1 end;


                                        0000 00000 CHECK_SECONDARY:
                                                           .WORD   Save nothing
                               50 00000000'  00  D2 00002          MCOML   SECONDARY, R0                   ; 1350
                                                 04 00009          RET                                     ; 1351
                                                                                                           ; 1353

; Routine Size:  10 bytes,    Routine Base:  $CODE$ + 0A6D
```

```
;  1192     1354  1 %sbttl 'set_range - set hourly restriction bits'
;  1193     1355  1 global routine set_range  =
;  1194     1356  2 begin
;  1195     1357  2
;  1196     1358  2 !++
;  1197     1359  2 !
;  1198     1360  2 ! FUNCTIONAL DESCRIPTION:
;  1199     1361  2 !
;  1200     1362  2 !       Set or clear appropriate hourly login restriction bits
;  1201     1363  2 !
;  1202     1364  2 ! INPUTS:
;  1203     1365  2 !
;  1204     1366  2 !       none
;  1205     1367  2 !
;  1206     1368  2 ! IMPLICIT INPUTS:
;  1207     1369  2 !
;  1208     1370  2 !
;  1209     1371  2 ! OUTPUTS:
;  1210     1372  2 !
;  1211     1373  2 !       none
;  1212     1374  2 !
;  1213     1375  2 ! ROUTINE VALUE:
;  1214     1376  2 !
;  1215     1377  2 !       true -> successful completion
;  1216     1378  2 !       false -> error in flag name
;  1217     1379  2 !--
;  1218     1380  2
;  1219     1381  2 builtin
;  1220     1382  2         ap;
;  1221     1383  2
;  1222     1384  2 map
;  1223     1385  2         ap              : ref block [,byte];
;  1224     1386  2
;  1225     1387  2 local
;  1226     1388  2         address,
;  1227     1389  2         width,
;  1228     1390  2         leftmost,
;  1229     1391  2         wrap;
;  1230     1392  2
;  1231     1393  2 !
;  1232     1394  2 ! Point to the flags longword to use.
;  1233     1395  2 !
;  1234     1396  2 if .secondary
;  1235     1397  2 then address = secondary_access
;  1236     1398  2 else address = primary_access;
;  1237     1399  2
;  1238     1400  2 !
;  1239     1401  2 ! Get the ending bit - note that this is the same as the starting
;  1240     1402  2 ! bit for a single hour.
;  1241     1403  2 !
   1242     1404  2 left_bit = .ap[tpa$l_number];
;  1243     1405  2
;  1244     1406  2 !
;  1245     1407  2 ! Make sure hour is not out of range
;  1246     1408  2 !
;  1247     1409  2 if .left_bit  gtru 23
;  1248     1410  2 or .right_bit gtru 23
```

```
;  1249          1411  2 then
;  1250          1412  2     return false;
;  1251          1413  2
;  1252          1414  2 !
;  1253          1415  2 ! Clear out the default field value if this is the first real value.
;  1254          1416  2 !
;  1255          1417  2 if ..address eql -1
;  1256          1418  2 then .address = 0;
;  1257          1419  2
;  1258          1420  2 wrap = false;
;  1259          1421  2 leftmost = .left_bit;
;  1260          1422  2
;  1261          1423  2 !
;  1262          1424  2 ! If the starting bit is greater than the ending bit, allow field to wrap
;  1263          1425  2 !
;  1264          1426  2 if .left_bit lss .right_bit
;  1265          1427  2 then
;  1266          1428  3     begin
;  1267          1429  3     leftmost = 23;
;  1268          1430  3     wrap = true;
;  1269          1431  2     end;
;  1270          1432  2
;  1271          1433  2 !
;  1272          1434  2 ! Set field width(s) and set appropriate bits
;  1273          1435  2 !
;  1274          1436  2 width = .leftmost - .right_bit + 1;
;  1275          1437  2
;  1276          1438  2 (.address)<.right_bit, .width> = -1;
;  1277          1439  2
;  1278          1440  2 if .wrap
;  1279          1441  2 then (.address)<0, .left_bit+1> = -1;
;  1280          1442  2
;  1281          1443  2 return true;
;  1282          1444  2
;  1283          1445  1 end;


                                003C 00000          .ENTRY  SET_RANGE, Save R2,R3,R4,R5                    ; 1355
                     55 00000000'   00   9E 00002          MOVAB   LEFT_BIT, R5
                     06         18  A5   E9 00009          BLBC    SECONDARY, 1$                          ; 1396
                     53         0C  A5   9E 0000D          MOVAB   SECONDARY_ACCESS, ADDRESS              ; 1397
                                    04   11 00011          BRB     2$
                     53         08  A5   9E 00013 1$:      MOVAB   PRIMARY_ACCESS, ADDRESS                ; 1395
                     65         1C  AC   D0 00017 2$:      MOVL    28(AP), LEFT_BIT                       ; 1404
                     54             65   D0 0001B          MOVL    LEFT_BIT, R4                           ; 1409
                     17             54   D1 0001E          CMPL    R4, #23
                                    47   1A 00021          BGTRU   6$
                     17         04  A5   D1 00023          CMPL    RIGHT_BIT, #23                         ; 1410
                                    41   1A 00027          BGTRU   6$
                FFFFFFFF    8F         63   D1 00029          CMPL    (ADDRESS), #-1                         ; 1417
                                    02   12 00030          BNEQ    3$
                                    63   D4 00032          CLRL    (ADDRESS)                              ; 1418
                                    52   D4 00034 3$:      CLRL    WRAP                                   ; 1420
                     50             54   D0 00036          MOVL    R4, LEFTMOST                           ; 1421
```

```
                              51        04   A5  D0 00039          MOVL    RIGHT_BIT, R1                        ; 1426
                              51             54  D1 0003D          CMPL    R4, RT                               ;
                                             0g  18 00040          BGEQ    4$                                   ;
                              50             17  D0 00042          MOVL    #23, LEFTMOST                        ; 1429
                              52             01  D0 00045          MOVL    #1, WRAP                             ; 1430
                              50             51  C2 00048 4$:      SUBL2   R1, R0                               ; 1436
                                             50  D6 0004B          INCL    WIDTH                                ;
        63           50       51 FFFFFFFF    8F  F0 0004D          INSV    #-1, R1, WIDTH, (ADDRESS)            ; 1438
                              0D             52  E9 00056          BLBC    WRAP, 5$                             ; 1440
                              50        01   A4  9E 00059          MOVAB   1(R4), R0                            ; 1441
        63           50       00 FFFFFFFF    8F  F0 0005D          INSV    #-1, #0, R0, (ADDRESS)               ;
                              50             01  D0 00066 5$:      MOVL    #1, R0                               ; 1443
                                                 04 00069          RET                                          ;
                                             50  D4 0006A 6$:      CLRL    R0                                   ; 1445
                                                 04 0006C          RET                                          ;

; Routine Size:  109 bytes,    Routine Base:  $CODE$ + 0A77
```

```
;  1285          1446  1 %sbttl 'getcputim - get cpu time quota'
;  1286          1447  1 global routine getcputim =
;  1287          1448  2 begin
;  1288          1449  2
;  1289          1450  2 !++
;  1290          1451  2 !
;  1291          1452  2 ! FUNCTIONAL DESCRIPTION:
;  1292          1453  2 !
;  1293          1454  2 !       Fill in the CPU time limit in the proper field in RECBUF.
;  1294          1455  2 !
;  1295          1456  2 ! INPUTS:
;  1296          1457  2 !
;  1297          1458  2 !       none
;  1298          1459  2 !
;  1299          1460  2 ! OUTPUTS:
;  1300          1461  2 !
;  1301          1462  2 !       none
;  1302          1463  2 !
;  1303          1464  2 ! ROUTINE VALUE:
;  1304          1465  2 !
;  1305          1466  2 !       true -> success
;  1306          1467  2 !       false -> invalid value supplied
;  1307          1468  2 !
;  1308          1469  2 !--
;  1309          1470  2
;  1310          1471  2 local
;  1311          1472  2         delta_time      : vector [2, long], ! system delta time temporary
;  1312          1473  2         cpu_time,                           ! CPU time work variables
;  1313          1474  2         remainder;
;  1314          1475  2
;  1315          1476  2 builtin
;  1316          1477  2         ediv;
;  1317          1478  2
;  1318          1479  2 !
;  1319          1480  2 ! Convert ASCII to system delta time
;  1320          1481  2 !
;  1321          1482  2
;  1322          1483  2 if not lib$cvt_dtime (tokendsc, delta_time)
;  1323          1484  2 then
;  1324          1485  2     return LIB$SIGNAL(UAF$_BADVALUE, 2, .tokenlen, .tokenptr);
;  1325          1486  2
;  1326          1487  2 !
;  1327          1488  2 ! Convert system delta time to a longword value which is in .01 sec. units
;  1328          1489  2 !
;  1329          1490  2
;  1330          1491  2 if (ediv (%ref (-200000), delta_time, cpu_time, remainder) and psl$m_v) neq 0
;  1331          1492  2 then
;  1332          1493  2     return LIB$SIGNAL(UAF$_VALTOOBIG, 2, .tokenlen, .tokenptr);
;  1333          1494  2
;  1334          1495  2 recbuf[uaf$l_cputim] = (.cpu_time ^ 1) + (if .remainder eql 0 then 0 else 1);
;  1335          1496  2
;  1336          1497  2 true
;  1337          1498  1 end;
```

```
                                                003C 00000          .ENTRY  GETCPUTIM, Save R2,R3,R4,R5                     ; 1447
                                    55 00000000G    00  9E 00002       MOVAB   TOKENLEN, R5
                                    54 00000000G    00  9E 00009       MOVAB   TOKENPTR, R4
                                    5E                  08  C2 00010       SUBL2   #8, SP
                                                        5E  DD 00013       PUSHL   SP                                      ; 1483
                                       00000000G        00  9F 00015       PUSHAB  TOKENDSC
                                00000000G   00          02  FB 0001B       CALLS   #2, LIB$CVT_DTIME
                                            0F          50  E8 00022       BLBS    R0, 1$
                                                        64  DD 00025       PUSHL   TOKENPTR                                ; 1485
                                            7E          65  3C 00027       MOVZWL  TOKENLEN, -(SP)
                                                        02  DD 0002A       PUSHL   #2
                                       00000000G        8F  DD 0002C       PUSHL   #UAF$_BADVALUE
                                                        1C  11 00032       BRB     2$
                53              52          6E FFFCF2C0 8F  7B 00034 1$:    EDIV    #-200000, DELTA_TIME, CPU_TIME, REMAINDER   ; 1491
                                                        50  DC 0003D       MOVPSL  R0
                                15              50      01  E1 0003F       BBC     #1, R0, 3$
                                                        64  DD 00043       PUSHL   TOKENPTR                                ; 1493
                                            7E          65  3C 00045       MOVZWL  TOKENLEN, -(SP)
                                                        02  DD 00048       PUSHL   #2
                                       00000000G        8F  DD 0004A       PUSHL   #UAF$_VALTOOBIG
                                00000000G   00          04  FB 00050 2$:    CALLS   #4, LIB$SIGNAL
                                                        04  00057       RET
                                                        53  D5 00058 3$:    TSTL    REMAINDER                               ; 1495
                                                        04  12 0005A       BNEQ    4$
                                                        50  D4 0005C       CLRL    R0
                                                        03  11 0005E       BRB     5$
                                            50          01  D0 00060 4$:    MOVL    #1, R0
                                00000000G   00        6042  3E 00063 5$:    MOVAW   (R0)[CPU_TIME], RECBUF+556
                                            50          01  D0 0006B       MOVL    #1, R0                                  ; 1498
                                                        04  0006E       RET                                             ;

; Routine Size:  111 bytes,    Routine Base:  $CODE$ + 0AE4
```

```
;  1339     1499  1 %sbttl 'getdevice - get default device name'
;  1340     1500  1 global routine getdevice =
;  1341     1501  2 begin
;  1342     1502  2
;  1343     1503  2 !++
;  1344     1504  2 !
;  1345     1505  2 ! FUNCTIONAL DESCRIPTION:
;  1346     1506  2 !
;  1347     1507  2 !       Fill in the default device name in RECBUF.
;  1348     1508  2 !       Will also append an ending colon if omitted.
;  1349     1509  2 !
;  1350     1510  2 ! INPUTS:
;  1351     1511  2 !
;  1352     1512  2 !       none
;  1353     1513  2 !
;  1354     1514  2 ! OUTPUTS:
;  1355     1515  2 !
;  1356     1516  2 !       none
;  1357     1517  2 !
;  1358     1518  2 ! ROUTINE VALUE:
;  1359     1519  2 !
;  1360     1520  2 !       true -> update successful
;  1361     1521  2 !       false -> error in specification
;  1362     1522  2 !--
;  1363     1523  2
;  1364     1524  2 local
;  1365     1525  2         strlen;
;  1366     1526  2
;  1367     1527  2 if not getstring (recbuf[uaf$t_defdev], uaf$s_defdev, counted_string)
;  1368     1528  2 then
;  1369     1529  2     return false;
;  1370     1530  2
;  1371     1531  2 !
;  1372     1532  2 ! Check to see that string has ending ':'
;  1373     1533  2 ! and add one if not.
;  1374     1534  2 ! If string length is zero, then field has been removed so
;  1375     1535  2 ! there's no need for the check.
;  1376     1536  2 !
;  1377     1537  2
;  1378     1538  2 strlen = .(recbuf[uaf$t_defdev])<0,8>;                    ! get string length
;  1379     1539  2
;  1380     1540  2 if .strlen eql 0
;  1381     1541  2 then
;  1382     1542  2     return true;
;  1383     1543  2
;  1384     1544  2 if .(recbuf[uaf$t_defdev] + .strlen)<0,8> neq colon
;  1385     1545  2 then
;  1386     1546  3     begin
;  1387     1547  3     if .strlen geq uaf$s_defdev-1
;  1388     1548  3     then
;  1389     1549  3         return LIB$SIGNAL(UAF$_INVDEV, 2, .tokenlen, .tokenptr);
;  1390     1550  3     (recbuf[uaf$t_defdev]+.strlen+1)<0,8> = colon;
;  1391     1551  3     (recbuf[uaf$t_defdev])<0,8> = .strlen + 1;
;  1392     1552  2     end;
;  1393     1553  2 return true;
;  1394     1554  1 end;
```

```
                                    000C 00000          .ENTRY  GETDEVICE, Save R2,R3                   ;1500
                             53 00000000G   00   9E 00002       MOVAB   RECBUF+116, R3
                                            01   DD 00009       PUSHL   #1                              ;1527
                                            20   DD 0000B       PUSHL   #32
                                            53   DD 0000D       PUSHL   R3
                        00000000V   00      03   FB 0000F       CALLS   #3, GETSTRING
                                    3A      50   E9 00016       BLBC    R0, 3$
                                    52      63   9A 00019       MOVZBL  RECBUF+116, STRLEN              ;1538
                                            31   13 0001C       BEQL    2$                              ;1540
                                    3A    6342   91 0001E       CMPB    RECBUF+116[STRLEN], #58         ;1544
                                            2B   13 00022       BEQL    2$
                                    1F      52   D1 00024       CMPL    STRLEN, #31                     ;1547
                                            1D   19 00027       BLSS    1$
                                 00000000G  00   DD 00029       PUSHL   TOKENPTR                        ;1549
                                    7E 00000000G   00   3C 0002F       MOVZWL  TOKENLEN, -(SP)
                                            02   DD 00036       PUSHL   #2
                                 00000000G  8F   DD 00038       PUSHL   #UAF$_INVDEV
                        00000000G   00      04   FB 0003E       CALLS   #4, LIB$SIGNAL
                                            04 00045           RET
                              01 A342       3A   90 00046 1$:   MOVB    #58, RECBUF+117[STRLEN]         ;1550
                          63      52        01   81 0004B       ADDB3   #1, STRLEN, RECBUF+116          ;1551
                                  50        01   D0 0004F 2$:   MOVL    #1, R0                          ;1553
                                            04 00052           RET
                                            50   D4 00053 3$:   CLRL    R0                              ;1554
                                            04 00055           RET

; Routine Size:  86 bytes,    Routine Base:  $CODE$ + 0B53
```

```
;1396    1555  1 %sbttl 'getdirectory - get default directory'
;1397    1556  1 global routine getdirectory =
;1398    1557  2 begin
;1399    1558  2 !++
;1400    1559  2 !
;1401    1560  2 ! FUNCTIONAL DESCRIPTION:
;1402    1561  2 !
;1403    1562  2 !       This routine parses the user default directory and places
;1404    1563  2 !       it into the buffer if the form is correct.  If delimiters
;1405    1564  2 !       are not present, they are added.
;1406    1565  2 !
;1407    1566  2 ! INPUTS:
;1408    1567  2 !
;1409    1568  2 !       none
;1410    1569  2 !
;1411    1570  2 ! OUTPUTS:
;1412    1571  2 !
;1413    1572  2 !       none
;1414    1573  2 !
;1415    1574  2 ! IMPLICIT INPUTS:
;1416    1575  2 !
;1417    1576  2 !       TOKENLEN = length of default directory string
;1418    1577  2 !       TOKENPTR = pointer to default directory string
;1419    1578  2 !
;1420    1579  2 !--
;1421    1580  2 local BRACKETLESS;
;1422    1581  2
;1423    1582  2 !
;1424    1583  2 ! Crank directory spec through tparse
;1425    1584  2 !
;1426    1585  2 tparse_block [tpa$l_stringcnt] = .tokenlen;
;1427    1586  2 tparse_block [tpa$l_stringptr] = .tokenptr;
;1428    1587  2
;1429    1588  2 if not lib$tparse (tparse_block, dir_states, dir_keys)
;1430    1589  2 then
;1431    1590  2     return LIB$SIGNAL(UAF$_BADVALUE, 2, .tokenlen, .tokenptr);
;1432    1591  2
;1433    1592  2 !
;1434    1593  2 ! Did the string have enclosing brackets (matching is guaranteed
;1435    1594  2 ! by the command parser) ??
;1436    1595  2 !
;1437    1596  2 BRACKETLESS = (.(.tokenptr)<0,8> neq '[') and (.(.tokenptr)<0,8> neq '<');
;1438    1597  2
;1439    1598  2 !
;1440    1599  2 ! Make sure that the string isn't too long
;1441    1600  2 !
;1442    1601  3 if (((.tokenlen+1)  geq  uaf$s_defdir) and not .BRACKETLESS)
;1443    1602  3 or ((.tokenlen+3) geq uaf$s_defdir)
;1444    1603  2 then
;1445    1604  2     return LIB$SIGNAL(UAF$_INVSTR, 2, .tokenlen, .tokenptr);
;1446    1605  2
;1447    1606  2 !
;1448    1607  2 ! Move the string into the UAF record
;1449    1608  2 !
;1450    1609  2 if .BRACKETLESS then
;1451    1610  3    begin
;1452    1611  3       vector[recbuf[uaf$t_defdir], 0;, byte] = .tokenlen + 2;
```

```
;  1453     1612  3        vector[recbuf[uaf$t_defdir], 1;, byte] = '[';
;  1454     1613  3        ch$copy(.tokenlen, .tokenptr, ' ',
;  1455     1614  3           uaf$s_defdir-2, recbuf[uaf$t_defdir]+2);
;  1456     1615  3        vector[recbuf[uaf$t_defdir], .tokenlen+2;, byte] = ']';
;  1457     1616  3     end
;  1458     1617  2  else
;  1459     1618  3     begin
;  1460     1619  3        (recbuf[uaf$t_defdir])<0,8> = .tokenlen;
;  1461     1620  3        ch$copy(.tokenlen, .tokenptr, ' ',
;  1462     1621  3           uaf$s_defdir-1, recbuf[uaf$t_defdir]+1);
;  1463     1622  2     end;
;  1464     1623  2
;  1465     1624  2  return true;
;  1466     1625  2
;  1467     1626  1  end;

                                                07FC 00000          .ENTRY  GETDIRECTORY, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1556
                                                                            R10
                                     5A 00000000'   00   9E 00002   MOVAB  TPARSE_BLOCK+8, R10
                                     59 00000000G   00   9E 00009   MOVAB  TOKENLEN, R9
                                     58 00000000G   00   9E 0C010   MOVAB  TOKENPTR, R8
                                     57 00000000G   00   9E 00017   MOVAB  RECBUF+148, R7
                                     6A             69   3C 0001E   MOVZWL TOKENLEN, TPARSE_BLOCK+8                    ; 1585
                                  04 AA             68   D0 00021   MOVL   TOKENPTR, TPARSE_BLOCK+12                   ; 1586
                                        00000000'   00   9F 00025   PUSHAB DIR_KEYS                                    ; 1588
                                        00000000'   00   9F 0002B   PUSHAB DIR_STATES
                                              F8    AA   9F 00031   PUSHAB TPARSE_BLOCK
                                00000000G   00          03   FB 00034   CALLS  #3, LIB$TPARSE
                                            0F          50   E8 0003B   BLBS   R0, 1$
                                                        68   DD 0003E   PUSHL  TOKENPTR                                ; 1590
                                            7E          69   3C 00040   MOVZWL TOKENLEN, -(SP)
                                                        02   DD 00043   PUSHL  #2
                                                00000000G   8F   DD 00045   PUSHL  #UAF$_BADVALUE
                                                        3F   11 0004B   BRB    6$
                                            52          68   D0 0004D 1$:   MOVL   TOKENPTR, R2                        ; 1596
                                                        51   D4 00050   CLRL   R1
                                     5B     8F          62   91 00052   CMPB   (R2), #91
                                                        02   13 00056   BEQL   2$
                                                        51   D6 00058   INCL   R1
                                                        50   D4 0005A 2$:   CLRL   R0
                                            3C          62   91 0005C   CMPB   (R2), #60
                                                        02   13 0005F   BEQL   3$
                                                        50   D6 00061   INCL   R0
                                            53          51   D2 00063 3$:   MCOML  R1, BRACKETLESS
                                     53     50          53   CB 00066   BICL3  BRACKETLESS, R0, BRACKETLESS
                                            51          69   3C 0006A   MOVZWL TOKENLEN, R1                         ; 1601
                                            50       01 A1   9E 0006D   MOVAB  1(R1), R0
                                            3F          50   D1 00071   CMPL   R0, #63
                                                        03   15 00074   BLEQ   4$
                                            09          53   E9 00076   BLBC   BRACKETLESS, 5$
                                            50       03 A1   9E 00079 4$:   MOVAB  3(R1), R0                         ; 1602
                                            3F          50   D1 0007D   CMPL   R0, #63
                                                        12   15 00080   BLEQ   7$
```

```
                                                  06  BB 00082 5$:      PUSHR   #^M<R1,R2>                                  ; 1604
                                                  02  DD 00084          PUSHL   #2                                          ;
                                      0C000000G   8F  DD 00086          PUSHL   #UAF$_INVSTR                                ;
                  00000000G  00                   04  FB 0008C 6$:      CALLS   #4, LIB$SIGNAL                              ;
                                                      04 00093          RET                                                 ;
                                 50               69  3C 00094 7$:      MOVZWL  TOKENLEN, R0                                ; 1611
                                 51               68  D0 00097          MOVL    TOKENPTR, R1                                ; 1613
                                 1A               53  E9 0009A          BLBC    BRACKETLESS, 8$                             ; 1609
                                 56       02      A0  9E 0009D          MOVAB   2(R0), R6                                   ; 1611
                                 67               56  90 000A1          MOVB    R6, RECBUF+148                              ;
                           01    A7       5B      8F  90 000A4          MOVB    #91, RECBUF+149                             ; 1612
            3E       20           61               50  2C 000A9          MOVC5   R0, (R1), #32, #62, RECBUF+150              ; 1614
                                                  02  A7 000AE                                                              ;
                                6746       5D      8F  90 000B0          MOVB    #93, RECBUF+148[R6]                         ; 1615
                                                  0A  11 000B5          BRB     9$                                          ; 1609
                                 67               50  90 000B7 8$:      MOVB    R0, RECBUF+148                              ; 1619
            3F       20           61               50  2C 000BA          MOVC5   R0, (R1), #32, #63, RECBUF+149              ; 1621
                                                  01  A7 000BF                                                              ;
                                 50               01  D0 000C1 9$:      MOVL    #1, R0                                      ; 1624
                                                      04 000C4          RET                                                 ; 1626

; Routine Size:  197 bytes,    Routine Base:  $CODE$ + 0BA9
```

```
;  1469          1627  1 %sbttl 'checkvalue - check UFD group and member value'
;  1470          1628  1 routine checkvalue =
;  1471          1629  2 begin
;  1472          1630  2 builtin ap;
;  1473          1631  2 map ap: ref block [,byte];
;  1474          1632  2 if .ap[tpa$l_number] gtr %o'377' then return false else return true;
;  1475          1633  1 end;


                                        0C00 00000 CHECKVALUE:
                                                           .WORD   Save nothing                    ;  1628
                    000000FF    8F      1C    AC  D1 00002       CMPL    28(AP), #255              ;  1632
                                              03  15 0000A       BLEQ    1$                        ;
                                              50  D4 0000C       CLRL    R0                        ;
                                                  04 0000E       RET                               ;
                                50            01  D0 0000F 1$:   MOVL    #1, R0                    ;
                                                  04 00012       RET                               ;  1633

; Routine Size:  19 bytes,    Routine Base:  $CODE$ + 0C6E
```

```
; 1477        1634  1 %sbttl 'checklength - check directory name length'
; 1478        1635  1 routine checklength =
; 1479        1636  2 begin
; 1480        1637  2 builtin ap;
; 1481        1638  2 map ap: ref block [,byte];
; 1482        1639  2 if .ap[tpa$l_tokencnt] gtr 39 then return false else return true;
; 1483        1640  1 end;


                                    0000 00000 CHECKLENGTH:
                                                       .WORD   Save nothing                    ; 1635
                              27        10   AC  D1 00002       CMPL    16(AP), #39             ; 1639
                                             03  15 00006       BLEQ    1$                      ;
                                             50  D4 00008       CLRL    R0                      ;
                                                 04 0000A       RET                             ;
                              50             01  D0 0000B 1$:   MOVL    #1, R0                  ;
                                                 04 0000E       RET                             ; 1640

; Routine Size:  15 bytes,    Routine Base:  $CODE$ + 0CB1
```

```
;1485    1641  1 %sbttl 'getflags - get flag bits'
;1486    1642  1 global routine getflags =
;1487    1643  2 begin
;1488    1644  2
;1489    1645  2 !++
;1490    1646  2 !
;1491    1647  2 ! FUNCTIONAL DESCRIPTION:
;1492    1648  2 !
;1493    1649  2 !       Get login flag bit settings.
;1494    1650  2 !
;1495    1651  2 ! INPUTS:
;1496    1652  2 !
;1497    1653  2 !       none
;1498    1654  2 !
;1499    1655  2 ! IMPLICIT INPUTS:
;1500    1656  2 !
;1501    1657  2 !       FLAG_TABLE - table of flag bit numbers and ascii names
;1502    1658  2 !
;1503    1659  2 ! OUTPUTS:
;1504    1660  2 !
;1505    1661  2 !       none
;1506    1662  2 !
;1507    1663  2 ! ROUTINE VALUE:
;1508    1664  2 !
;1509    1665  2 !       true -> successful completion
;1510    1666  2 !       false -> error in flag name
;1511    1667  2 !--
;1512    1668  2
;1513    1669  2 local
;1514    1670  2         status,
;1515    1671  2         keyword_dsc     : vector [2],
;1516    1672  2         no_flag,
;1517    1673  2         code,                           ! return from find_val
;1518    1674  2         ientry;
;1519    1675  2
;1520    1676  2 !
;1521    1677  2 ! Retrieve flag values as long as there are any present
;1522    1678  2 !
;1523    1679  2
;1524    1680  2 while cli$get_value (sd_flags, tokendsc) do
;1525    1681  2     !
;1526    1682  2     ! Initialize keyword descriptor
;1527    1683  2     !
;1528    1684  3     begin
;1529    1685  3     keyword_dsc [0] = .tokenlen;
;1530    1686  3     keyword_dsc [1] = .tokenptr;
;1531    1687  3     no_flag = false;
;1532    1688  3
;1533    1689  3     !
;1534    1690  3     ! If 'NO' prefix specified, set no_flag and adjust the
;1535    1691  3     !    keyword descriptor to remove the 'NO'
;1536    1692  3     !
;1537    1693  3     if ch$eql (2, .no_dsc [1], 2, .keyword_dsc [1])
;1538    1694  3     then
;1539    1695  4         begin
;1540    1696  4         keyword_dsc [0] = .keyword_dsc [0] - 2;
;1541    1697  4         keyword_dsc [1] = .keyword_dsc [1] + 2;
```

```
;  1542          1698  4          no_flag = true;
;  1543          1699  3          end;
;  1544          1700  3
;  1545          1701  4      if (status = lib$lookup_key (keyword_dsc, flag_table, code))
;  1546          1702  3      then
;  1547          1703  3          !
;  1548          1704  3          ! valid flag, set bit
;  1549          1705  3          !
;  1550          1706  3          (recbuf[uaf$l_flags])<.code,1> = not .no_flag
;  1551          1707  3      else
;  1552          1708  3          select .status of
;  1553          1709  3              set
;  1554          1710  3              !
;  1555          1711  3              ! ambiguous keyword?
;  1556          1712  3              !
;  1557          1713  3              [lib$_ambkey] : signal (cli$_abkeyw);
;  1558          1714  3              !
;  1559          1715  3              ! unrecognized keyword?
;  1560          1716  3              !
;  1561          1717  3              [lib$_unrkey] : signal (cli$_ivkeyw);
;  1562          1718  3              !
;  1563          1719  3              ! whatever, always return an error if we're in here
;  1564          1720  3              !
;  1565          1721  3              [always]      : return false;
;  1566          1722  3              tes;
;  1567          1723  3
;  1568          1724  2      end;
;  1569          1725  2
;  1570          1726  2 return true;
;  1571          1727  1 end;


                                 001C 00000          .ENTRY  GETFLAGS, Save R2,R3,R4                 ; 1642
                      54 00000000G   00   9E 00002          MOVAB   LIB$SIGNAL, R4                  ;
                      5E             0C   C2 00009          SUBL2   #12, SP                         ;
                         00000000G   00   9F 0000C 1$:       PUSHAB  TOKENDSC                        ; 1680
                         00000000'   00   9F 00012          PUSHAB  SD_FLAGS                        ;
00000000G             00             02   FB 00018          CALLS   #2, CLI$GET_VALUE               ;
                      76             50   E9 0001F          BLBC    R0, 5$                          ;
            04        AE 00000000G   00   3C 00022          MOVZWL  TOKENLEN, KEYWORD_DSC           ; 1685
            08        AE 00000000G   00   D0 0002A          MOVL    TOKENPTR, KEYWORD_DSC+4         ; 1686
                                     52   D4 00032          CLRL    NO_FLAG                         ; 1687
                      50 00000000'   00   D0 00034          MOVL    NO_DSC+4, R0                    ; 1693
            08        BE             60   B1 0003B          CMPW    (R0), @KEYWORD_DSC+4            ;
                                     0B   12 0003F          BNEQ    2$                              ;
            04        AE             02   C2 00041          SUBL2   #2, KEYWORD_DSC                 ; 1696
            08        AE             02   C0 00045          ADDL2   #2, KEYWORD_DSC+4               ; 1697
                      52             01   D0 00049          MOVL    #1, NO_FLAG                     ; 1698
                                     5E   DD 0004C 2$:       PUSHL   SP                              ; 1701
                         C0000000'   00   9F 0004E          PUSHAB  FLAG_TABLE                      ;
                                0C   AE   9F 00054          PUSHAB  KEYWORD_DSC                     ;
00000000G             00             03   FB 00057          CALLS   #3, LIB$LOOKUP_KEY              ;
                      53             50   D0 0005E          MOVL    R0, STATUS                      ;
                      0E             53   E9 00061          BLBC    STATUS, 3$                      ;
```

```
                                                52  D2 00064          MCOML   NO_FLAG, R0                              ; 1706
00000000G  00            01                     50  F0 00067          INSV    R0, CODE, #1, RECBUF+468
                                                9A  11 00070          BRB     1$
                                    00000000G   53  D1 00072 3$:       CMPL    STATUS, #LIB$_AMBKEY                     ; 1713
                                                09  12 00079          BNEQ    4$
                                  00000000G     8F  DD 0007B          PUSHL   #CLI$_ABKEYW
                                                01  FB 00081          CALLS   #1, LIB$SIGNAL
                                    00000000G   53  D1 00084 4$:       CMPL    STATUS, #LIB$_UNRKEY                     ; 1717
                                                0F  12 0008B          BNEQ    6$
                                  00000000G     8F  DD 0008D          PUSHL   #CLI$_IVKEYW
                                                01  FB 00093          CALLS   #1, LIB$SIGNAL
                                                04  11 00096          BRB     6$                                       ; 1721
                                                01  D0 00098 5$:       MOVL    #1, R0                                   ; 1726
                                                04  0009B             RET
                                                50  D4 0009C 6$:       CLRL    R0                                       ; 1727
                                                04  0009E             RET

; Routine Size:  159 bytes,    Routine Base:  $CODE$ + 0C90
```

```
;  1573          1728  1 %sbttl 'getpflags - get old per day login flags'
;  1574          1729  1 global routine getpflags (qual_name) =
;  1575          1730  2 begin
;  1576          1731  2
;  1577          1732  2 !++
;  1578          1733  2 !
;  1579          1734  2 ! FUNCTIONAL DESCRIPTION:
;  1580          1735  2 !
;  1581          1736  2 !       Get prime day login flag bit settings.
;  1582          1737  2 !
;  1583          1738  2 ! INPUTS:
;  1584          1739  2 !
;  1585          1740  2 !       qual_name - address of descriptor for qualifier name
;  1586          1741  2 !
;  1587          1742  2 ! IMPLICIT INPUTS:
;  1588          1743  2 !
;  1589          1744  2 !       OPFLAG_TABLE - table of flag bit numbers and ascii names
;  1590          1745  2 !       PRIMARY - true if getting primary day, false if secondary
;  1591          1746  2 !
;  1592          1747  2 ! OUTPUTS:
;  1593          1748  2 !
;  1594          1749  2 !       none
;  1595          1750  2 !
;  1596          1751  2 ! ROUTINE VALUE:
;  1597          1752  2 !
;  1598          1753  2 !       true -> successful completion
;  1599          1754  2 !       false -> error in flag name
;  1600          1755  2 !--
;  1601          1756  2
;  1602          1757  2 local
;  1603          1758  2         status,
;  1604          1759  2         keyword_dsc     : vector [2],
;  1605          1760  2         no_flag,
;  1606          1761  2         code,                                   ! return from find_val
;  1607          1762  2         ientry;
;  1608          1763  2
;  1609          1764  2 !
;  1610          1765  2 ! Retrieve keywords as long as any are present
;  1611          1766  2 !
;  1612          1767  2 while cli$get_value (.qual_name, tokendsc) do
;  1613          1768  2
;  1614          1769  3     begin
;  1615          1770  3     keyword_dsc [0] = .tokenlen;
;  1616          1771  3     keyword_dsc [1] = .tokenptr;
;  1617          1772  3     no_flag = 0;
;  1618          1773  3
;  1619          1774  3     !
;  1620          1775  3     ! Test for presence of 'NO'prefix.  If present, adjust descriptor
;  1621          1776  3     !    to remove it, and set no_flag.
;  1622          1777  3     !
;  1623          1778  3     if ch$eql (2, .no_dsc [1], 2, .keyword_dsc [1])
;  1624          1779  3     then
;  1625          1780  3
;  1626          1781  4         begin
;  1627          1782  4         keyword_dsc [0] = .keyword_dsc [0] - 2;
;  1628          1783  4         keyword_dsc [1] = .keyword_dsc [1] + 2;
;  1629          1784  4         no_flag = -1;
```

```
;  1630     1785  3          end,
;  1631     1786  3
;  1632     1787  4      if (status = lib$lookup_key (keyword_dsc, opflag_table, code))
;  1633     1788  3      then
;  1634     1789  3          !
;  1635     1790  3          ! valid flag, set bit
;  1636     1791  3          !
;  1637     1792  3          case .code from uaf$k_disdialup to uaf$k_disnetwork of
;  1638     1793  3              set
;  1639     1794  3              [uaf$k_disdialup]:
;  1640     1795  4                  begin
;  1641     1796  4                  if .primary
;  1642     1797  4                  then recbuf[uaf$b_dialup_access_p] = not .no_flag
;  1643     1798  4                  else recbuf[uaf$b_dialup_access_s] = not .no_flag;
;  1644     1799  3                  end;
;  1645     1800  3              [uaf$k_disnetwork]:
;  1646     1801  4                  begin
;  1647     1802  4                  if .primary
;  1648     1803  4                  then recbuf[uaf$b_remote_access_p] = not .no_flag
;  1649     1804  4                  else recbuf[uaf$b_remote_access_s] = not .no_flag;
;  1650     1805  3                  end;
;  1651     1806  3              tes
;  1652     1807  3      else
;  1653     1808  3          select .status of
;  1654     1809  3              set
;  1655     1810  3              !
;  1656     1811  3              ! ambiguous keyword?
;  1657     1812  3              !
;  1658     1813  3              [lib$_ambkey] : signal (cli$_abkeyw);
;  1659     1814  3              !
;  1660     1815  3              ! unrecognized keyword?
;  1661     1816  3              !
;  1662     1817  3              [lib$_unrkey] : signal (cli$_ivkeyw);
;  1663     1818  3              !
;  1664     1819  3              ! whatever, always return an error if we're in here
;  1665     1820  3              !
;  1666     1821  3              [always]      : return false;
;  1667     1822  3              tes;
;  1668     1823  3
;  1669     1824  2      end;
;  1670     1825  2
;  1671     1826  2 return true;
;  1672     1827  1 end;


                                    003C 00000          .ENTRY  GETPFLAGS, Save R2,R3,R4,R5                 ; 1729
                          55 00000000G  00   9E 00002   MOVAB   LIB$SIGNAL, R5                               ;
                          54 00000000G  00   9E 00009   MOVAB   RECBUF+490, R4                              ;
                          5E            0C   C2 00010   SUBL2   #12, SP                                    ;
                             00000000G  00   9F 00013 1$: PUSHAB  TOKENDSC                                   ; 1767
                                   04   AC   DD 00019   PUSHL   QUAL_NAME                                   ;
                00000000G  00          02   FB 0001C   CALLS   #2, CLI$GET_VALUE                           ;
                           03          50   E8 00023   BLBS    R0, 2$                                      ;
                                  009F 31   31 00026   BRW     12$                                         ;
```

```
                                 04   AE 00000000G   00   3C 00029 2$:     MOVZWL  TOKENLEN, KEYWORD_DSC                    ;1770
                                 08   AE 00000000G   00   D0 00031         MOVL    TOKENPTR, KEYWORD_DSC+4                  ;1771
                                                     52   D4 00039         CLRL    NO_FLAG                                  ;1772
                                      50 00000000'   00   D0 0003B         MOVL    NO_DSC+4, R0                             ;1778
                                 08   BE             60   B1 00042         CMPW    (R0), @KEYWORD_DSC+4
                                                     0B   12 00046         BNEQ    3$
                                 04   AE             02   C2 00048         SUBL2   #2, KEYWORD_DSC                          ;1782
                                 08   AE             02   C0 0004C         ADDL2   #2, KEYWORD_DSC+4                        ;1783
                                      52             01   CE 00050         MNEGL   #1, NO_FLAG                              ;1784
                                                     5E   DD 00053 3$:      PUSHL   SP                                       ;1787
                                           00000000'   00   9F 00055         PUSHAB  OPFLAG_TABLE
                                                 0C    AE   9F 0005B         PUSHAB  KEYWORD_DSC
                                00000000G        00         03   FB 0005E    CALLS   #3, LIB$LOOKUP_KEY
                                                 53         50   D0 00065    MOVL    R0, STATUS
                                                 37         53   E9 00068    BLBC    STATUS, 10$
                                                 51 00000000'   00   D0 0006B    MOVL    PRIMARY, R1                          ;1796
                                                 50             52   D2 00072    MCOML   NO_FLAG, R0                          ;1797
                                    01           01             6E   CF 00075    CASEL   CODE, #1, #1                         ;1792
                                              0016                0004      00079 4$:     .WORD   5$-4$,-
                                                                                          8$-4$
                                                 07             51   E9 0007D 5$:     BLBC    R1, 7$                           ;1796
                                64        18     00             50   F0 00080         INSV    R0, #0, #24, RECBUF+490          ;1797
                                                                 8C   11 00085 6$:     BRB     1$
                             03 A4        18     00             50   F0 00087 7$:     INSV    R0, #0, #24, RECBUF+493          ;1798
                                                                 84   11 0008D         BRB     1$                               ;1792
                                                 08             51   E9 0008F 8$:     BLBC    R1, 9$                           ;1802
                             06 A4        18     00             50   F0 00092         INSV    R0, #0, #24, RECBUF+496          ;1803
                                                                 EB   11 00098         BRB     6$
                             09 A4        18     00             50   F0 0009A 9$:     INSV    R0, #0, #24, RECBUF+499          ;1804
                                                                 E3   11 000A0         BRB     6$                               ;1792
                                00000000G        8F             53   D1 000A2 10$:    CMPL    STATUS, #LIB$_AMBKEY             ;1813
                                                                 09   12 000A9         BNEQ    11$
                                           00000000G             8F   DD 000AB         PUSHL   #CLI$_ABKEYW
                                                 65              01   FB 000B1         CALLS   #1, LIB$SIGNAL
                                00000000G        8F             53   D1 000B4 11$:    CMPL    STATUS, #LIB$_UNRKEY             ;1817
                                                                 0F   12 000BB         BNEQ    13$
                                           00000000G             8F   DD 000BD         PUSHL   #CLI$_IVKEYW
                                                 65              01   FB 000C3         CALLS   #1, LIB$SIGNAL
                                                                 04   11 000C6         BRB     13$                              ;1821
                                                 50              01   D0 000C8 12$:    MOVL    #1, R0                           ;1826
                                                                 04   000CB         RET
                                                                 50   D4 000CC 13$:    CLRL    R0                               ;1827
                                                                 04   000CE         RET

; Routine Size:  207 bytes,    Routine Base:  $CODE$ + 0D2F
```

```
;1674     1828  1 %sbttl 'getrestrict - parse hourly restriction ranges'
;1675     1829  1 global routine getrestrict (qual_name) =
;1676     1830  2 begin
;1677     1831  2
;1678     1832  2 !++
;1679     1833  2 !
;1680     1834  2 ! FUNCTIONAL DESCRIPTION:
;1681     1835  2 !
;1682     1836  2 !       Parse hourly login restrictions for primary or secondary days
;1683     1837  2 !
;1684     1838  2 ! INPUTS:
;1685     1839  2 !
;1686     1840  2 !       qual_name - descriptor of qualifier name to use
;1687     1841  2 !
;1688     1842  2 ! IMPLICIT INPUTS:
;1689     1843  2 !
;1690     1844  2 !
;1691     1845  2 ! OUTPUTS:
;1692     1846  2 !
;1693     1847  2 !       none
;1694     1848  2 !
;1695     1849  2 ! ROUTINE VALUE:
;1696     1850  2 !
;1697     1851  2 !       true -> successful completion
;1698     1852  2 !       false -> error in flag name
;1699     1853  2 !--
;1700     1854  2
;1701     1855  2 !
;1702     1856  2 ! Initialize output values.
;1703     1857  2 !
;1704     1858  2 primary_access = 0;
;1705     1859  2 secondary_access = 0;
;1706     1860  2
;1707     1861  2 !
;1708     1862  2 ! Fetch value list items and parse them.
;1709     1863  2 !
;1710     1864  2 while true
;1711     1865  2 do
;1712     1866  3     begin
;1713     1867  3     if not cli$get_value (.qual_name, tokendsc)
;1714     1868  3     then exitloop;
;1715     1869  3
;1716     1870  3     tparse_block[tpa$l_stringcnt] = .tokenlen;
;1717     1871  3     tparse_block[tpa$l_stringptr] = .tokenptr;
;1718     1872  3     if not lib$tparse (tparse_block, access_states, access_keys)
;1719     1873  3     then return LIB$SIGNAL(UAF$_BADVALUE, 2, .tokenlen, .tokenptr);
;1720     1874  2     end;
;1721     1875  2
;1722     1876  2 !
;1723     1877  2 ! Write the modified fields into the UAF record.
;1724     1878  2 !
;1725     1879  2 if not .secondary
;1726     1880  2 then
;1727     1881  3     begin
;1728     1882  3     recbuf[uaf$b_network_access_p] = .primary_access;
;1729     1883  3     recbuf[uaf$b_batch_access_p] = .primary_access;
;1730     1884  3     recbuf[uaf$b_local_access_p] = .primary_access;
```

```
;  1731          1885  3      recbuf[uaf$b_dialup_access_p] = .primary_access;
;  1732          1886  3      recbuf[uaf$b_remote_access_p] = .primary_access;
;  1733          1887  3      end
;  1734          1888  2  else
;  1735          1889  3      begin
;  1736          1890  3      recbuf[uaf$b_network_access_s] = .secondary_access;
;  1737          1891  3      recbuf[uaf$b_batch_access_s] = .secondary_access;
;  1738          1892  3      recbuf[uaf$b_local_access_s] = .secondary_access;
;  1739          1893  3      recbuf[uaf$b_dialup_access_s] = .secondary_access;
;  1740          1894  3      recbuf[uaf$b_remote_access_s] = .secondary_access;
;  1741          1895  2      end;
;  1742          1896  2
;  1743          1897  2  true
;  1744          1898  1  end;

                                                  003C 00000          .ENTRY  GETRESTRICT, Save R2,R3,R4,R5                ; 1829
                                        55 00000000G  00   9E 00002    MOVAB   TOKENPTR, R5                                ;
                                        54 00000000G  00   9E 00009    MOVAB   TOKENLEN, R4                                ;
                                        53 00000000'  00   9E 00010    MOVAB   PRIMARY_ACCESS, R3                          ;
                                        52 00000000G  00   9E 00017    MOVAB   RECBUF+472, R2                              ;
                                                      63   7C 0001E    CLRQ    PRIMARY_ACCESS                              ; 1858
                                           00000000G  00   9F 00020 1$: PUSHAB  TOKENDSC                                  ; 1867
                                                 04   AC   DD 00026    PUSHL   QUAL_NAME                                   ;
                                  00000000G      00        02   FB 00029    CALLS   #2, CLI$GET_VALUE                   ;
                                                 36        50   E9 00030    BLBC    R0, 2$                              ;
                                           D4    A3        64   3C 00033    MOVZWL  TOKENLEN, TPARSE_BLOCK+8            ; 1870
                                           D8    A3        65   D0 00037    MOVL    TOKENPTR, TPARSE_BLOCK+12           ; 1871
                                                    00000000'  00   9F 0003B    PUSHAB  ACCESS_KEYS                         ; 1872
                                                    00000000'  00   9F 00041    PUSHAB  ACCESS_STATES                       ;
                                                          CC   A3   9F 00047    PUSHAB  TPARSE_BLOCK                        ;
                                  00000000G      00        03   FB 0004A    CALLS   #3, LIB$TPARSE                      ;
                                                 CC        50   E8 00051    BLBS    R0, 1$                              ;
                                                           65   DD 00054    PUSHL   TOKENPTR                            ; 1873
                                                 7E        64   3C 00056    MOVZWL  TOKENLEN, -(SP)                     ;
                                                           02   DD 00059    PUSHL   #2                                  ;
                                                    00000000G  8F   DD 0005B    PUSHL   #UAF$_BADVALUE                      ;
                                  00000000G      00        04   FB 00061    CALLS   #4, LIB$SIGNAL                      ;
                                                           04   00068    RET                                         ;
                                                 22        10   A3   E8 00069 2$: BLBS    SECONDARY, 3$                   ; 1879
                                                 50             63   D0 0006D    MOVL    PRIMARY_ACCESS, R0          ; 1882
                                           62    18             00   F0 00070    INSV    R0, #0, #24, RECBUF+472     ;
                                     06    A2    18             00   F0 00075    INSV    R0, #0, #24, RECBUF+478     ; 1883
                                     0C    A2    18             00   F0 0007B    INSV    R0, #0, #24, RECBUF+484     ; 1884
                                     12    A2    18             00   F0 00081    INSV    R0, #0, #24, RECBUF+490     ; 1885
                                     18    A2    18             00   F0 00087    INSV    R0, #0, #24, RECBUF+496     ; 1886
                                                 22        11   00   11 0008B    BRB     4$                          ; 1879
                                                 50        04   A3   D0 0008F 3$: MOVL    SECONDARY_ACCESS, R0        ; 1890
                                     03    A2    18             00   F0 00093    INSV    R0, #0, #24, RECBUF+475     ;
                                     09    A2    18             00   F0 00099    INSV    R0, #0, #24, RECBUF+481     ; 1891
                                     0F    A2    18             00   F0 0009F    INSV    R0, #0, #24, RECBUF+487     ; 1892
                                     15    A2    18             00   F0 000A5    INSV    R0, #0, #24, RECBUF+493     ; 1893
                                     1B    A2    18             00   F0 000AB    INSV    R0, #0, #24, RECBUF+499     ; 1894
                                                 50             01   D0 000B1 4$: MOVL    #1, R0                      ; 1898
```

```
                                        04 000B4          RET                                                    ;

; Routine Size:  181 bytes,    Routine Base:  $CODE$ + 0DFE
```

```
;1746    1899  1 %sbttl 'getprimedays - get primary day list'
;1747    1900  1 global routine getprimedays =
;1748    1901  2 begin
;1749    1902  2
;1750    1903  2 !++
;1751    1904  2 !
;1752    1905  2 ! FUNCTIONAL DESCRIPTION:
;1753    1906  2 !
;1754    1907  2 !       Get list of day considered prime days.
;1755    1908  2 !
;1756    1909  2 ! INPUTS:
;1757    1910  2 !
;1758    1911  2 !       none
;1759    1912  2 !
;1760    1913  2 ! IMPLICIT INPUTS:
;1761    1914  2 !
;1762    1915  2 !       PDFLAG_TABLE - table of flag bit numbers and ascii names
;1763    1916  2 !
;1764    1917  2 ! OUTPUTS:
;1765    1918  2 !
;1766    1919  2 !       none
;1767    1920  2 !
;1768    1921  2 ! ROUTINE VALUE:
;1769    1922  2 !
;1770    1923  2 !       true -> successful completion
;1771    1924  2 !       false -> error in flag name
;1772    1925  2 !--
;1773    1926  2
;1774    1927  2 local
;1775    1928  2         status,
;1776    1929  2         keyword_dsc     : vector [2],
;1777    1930  2         no_flag,
;1778    1931  2         code,                                   ! return from find_val
;1779    1932  2         ientry;
;1780    1933  2
;1781    1934  2 !
;1782    1935  2 ! Retrieve keywords as long as they are present
;1783    1936  2 !
;1784    1937  2 while cli$get_value (sd_primedays, tokendsc) do
;1785    1938  2
;1786    1939  3     begin
;1787    1940  3     keyword_dsc [0] = .tokenlen;
;1788    1941  3     keyword_dsc [1] = .tokenptr;
;1789    1942  3     no_flag = false;
;1790    1943  3
;1791    1944  3     !
;1792    1945  3     ! Test for presence of the 'NO' prefix.  If present, adjust the
;1793    1946  3     !    descriptor to remove it, and set nc_flag
;1794    1947  3     !
;1795    1948  3     if ch$eql (2, .no_dsc [1], 2, .keyword_dsc [1])
;1796    1949  3     then
;1797    1950  4         begin
;1798    1951  4         keyword_dsc [0] = .keyword_dsc [0] - 2;
;1799    1952  4         keyword_dsc [1] = .keyword_dsc [1] + 2;
;1800    1953  4         no_flag = true;
;1801    1954  3         end;
;1802    1955  3
```

```
;  1803          1956  4     if (status = lib$lookup_key (keyword_dsc, pdflag_table, code))
;  1804          1957  3     then
;  1805          1958  3
;  1806          1959  3         ! valid flag, set bit
;  1807          1960  3         !
;  1808          1961  3         (recbuf[uaf$b_primedays])<.code,1> =  .no_flag
;  1809          1962  3     else
;  1810          1963  3         select .status of
;  1811          1964  3             set
;  1812          1965  3             !
;  1813          1966  3             ! ambiguous keyword?
;  1814          1967  3             !
;  1815          1968  3             [lib$_ambkey] : signal (cli$_abkeyw);
;  1816          1969  3             !
;  1817          1970  3             ! unrecognized keyword?
;  1818          1971  3             !
;  1819          1972  3             [lib$_unrkey] : signal (cli$_ivkeyw);
;  1820          1973  3             !
;  1821          1974  3             ! whatever, always return an error if we're in here
;  1822          1975  3             !
;  1823          1976  3             [always]      : return false;
;  1824          1977  3             tes;
;  1825          1978  3
;  1826          1979  2     end;
;  1827          1980  2
;  1828          1981  2 return true;
;  1829          1982  1 end;


                                        001C 00000          .ENTRY  GETPRIMEDAYS, Save R2,R3,R4                 ; 1900
                                54 00000000G    00   9E 00002      MOVAB   LIB$SIGNAL, R4
                                5E              0C   C2 00009      SUBL2   #12, SP
                                   00000000G    00   9F 0000C 1$:  PUSHAB  TOKENDSC                            ; 1937
                                   00000000'    00   9F 00012      PUSHAB  SD_PRIMEDAYS
                    00000000G   00              02   FB 00018      CALLS   #2, CLI$GET_VALUE
                                73              50   E9 0001F      BLBC    R0, 5$
                          04    AE 00000000G    00   3C 00022      MOVZWL  TOKENLEN, KEYWORD_DSC               ; 1940
                          08    AE 00000000G    00   D0 00026      MOVL    TOKENPTR, KEYWORD_DSC+4             ; 1941
                                                52   D4 0002A      CLRL    NO_FLAG                             ; 1942
                                50 00000000'    00   D0 0002C      MOVL    NO_DSC+4, R0                        ; 1948
                          08    BE              60   B1 00030      CMPW    (R0), @KEYWORD_DSC+4
                                                0B   12 00037      BNEQ    2$
                          04    AE              02   C2 00039      SUBL2   #2, KEYWORD_DSC                     ; 1951
                          08    AE              02   C0 0003D      ADDL2   #2, KEYWORD_DSC+4                   ; 1952
                                52              01   D0 00041      MOVL    #1, NO_FLAG                         ; 1953
                                                5E   DD 00044 2$:  PUSHL   SP                                  ; 1956
                                   00000000'    00   9F 00046      PUSHAB  PDFLAG_TABLE
                                         0C     AE   9F 0004A      PUSHAB  KEYWORD_DSC
                    00000000G   00              03   FB 0004D      CALLS   #3, LIB$LOOKUP_KEY
                                53              50   D0 00054      MOVL    R0, STATUS
                                0C              53   E9 00057      BLBC    STATUS, 3$
00000000G   00          01                      52   F0 0005A      INSV    NO_FLAG, CODE, #1, RECBUF+514       ; 1961
                                                9D   11 00061      BRB     1$
                    00000000G   8F              53   D1 00063 3$:  CMPL    STATUS, #LIB$_AMBKEY                ; 1968
```

```
                                    09  12 00076          BNEQ    4$
                          00000000G 8F  DD 00078          PUSHL   #CLI$_ARKEYW
                                64  01  FB 0007E          CALLS   #1, LIB$SIGNAL
                    00000000G   8F  53  D1 00081 4$:      CMPL    STATUS, #LIB$_UNRKEY                    1972
                                    0F  12 00088          BNEQ    6$
                          00000000G 8F  DD 0008A          PUSHL   #CLI$_IVKEYW
                                64  01  FB 00090          CALLS   #1, LIB$SIGNAL
                                    04  11 00093          BRB     6$                                      1976
                                50  01  D0 00095 5$:      MOVL    #1, R0                                  1981
                                    04  00098          RET
                                    50  D4 00099 6$:      CLRL    R0                                      1982
                                    04  0009B          RET

; Routine Size:  156 bytes,    Routine Base:  $CODE$ + 0EB3
```

```
;  1831     1983  1 %sbttl 'getpriv - get process privileges'
;  1832     1984  1 global routine getpriv (qual_name, privadr) =
;  1833     1985  2 begin
;  1834     1986  2
;  1835     1987  2 !++
;  1836     1988  2 !
;  1837     1989  2 ! FUNCTIONAL DESCRIPTION:
;  1838     1990  2 !
;  1839     1991  2 !       Process privilege mask specification and set or
;  1840     1992  2 !       clear proper bits in privilege quadword in RECBUF.
;  1841     1993  2 !
;  1842     1994  2 ! INPUTS:
;  1843     1995  2 !
;  1844     1996  2 !       qual_name - descriptor of qualifier name to use
;  1845     1997  2 !       privadr - address to store privilege bits
;  1846     1998  2 !
;  1847     1999  2 ! IMPLICIT INPUTS:
;  1848     2000  2 !
;  1849     2001  2 !       none
;  1850     2002  2 !
;  1851     2003  2 ! OUTPUTS:
;  1852     2004  2 !
;  1853     2005  2 !       none
;  1854     2006  2 !
;  1855     2007  2 ! IMPLICIT OUTPUTS:
;  1856     2008  2 !
;  1857     2009  2 !       none
;  1858     2010  2 !
;  1859     2011  2 ! ROUTINE VALUE:
;  1860     2012  2 !
;  1861     2013  2 !       true -> input processed successfully
;  1862     2014  2 !       false -> error in privilege specification
;  1863     2015  2 !
;  1864     2016  2 ! SIDE EFFECTS:
;  1865     2017  2 !
;  1866     2018  2 !       none
;  1867     2019  2 !--
;  1868     2020  2
;  1869     2021  2 !
;  1870     2022  2 ! Loop through if this is a list and check all of the names.
;  1871     2023  2 ! Set or clear appropriate bits depending on the 'NO' prefix.
;  1872     2024  2 !
;  1873     2025  2
;  1874     2026  2 while cli$get_value (.qual_name, tokendsc) do
;  1875     2027  3     begin
;  1876     2028  3
;  1877     2029  3     select prv$setpriv (tokendsc, .privadr) of
;  1878     2030  3
;  1879     2031  3         set
;  1880     2032  3         !
;  1881     2033  3         ! Privilege name not found
;  1882     2034  3         !
;  1883     2035  3         [prv$_invnam]:
;  1884     2036  3             return LIB$SIGNAL(UAF$_PRVNOTFND, 2, .tokenlen, .tokenptr);
;  1885     2037  3
;  1886     2038  3         !
;  1887     2039  3         ! Privilege name not unique
```

```
;  1888          2040  3          !
;  1889          2041  3          [prv$_notunq]:
;  1890          2042  3              return LIB$SIGNAL(UAF$_PRVNOTUNQ, 2, .tokenlen, .tokenptr);
;  1891          2043  3
;  1892          2044  3      tes;
;  1893          2045  3
;  1894          2046  2      end;
;  1895          2047  2
;  1896          2048  2 return true;
;  1897          2049  1 end;


                                                003C 00000          .ENTRY  GETPRIV, Save R2,R3,R4,R5                ; 1984
                                    55 00000000G    00   9E 00002          MOVAB   TOKENDSC, R5
                                    54 00000000G    00   9E 00009          MOVAB   TOKENLEN, R4
                                    53 00000000G    00   9E 00010          MOVAB   TOKENPTR, R3
                                                    55   DD 00017 1$:       PUSHL   R5                               ; 2026
                                               04   AC   DD 00019          PUSHL   QUAL_NAME
                                00000000G      00        02   FB 0001C     CALLS   #2, CLI$GET_VALUE
                                               45        50   E9 00023     BLBC    R0, 4$
                                               08   AC   DD 00026          PUSHL   PRIVADR                          ; 2029
                                                    55   DD 00029          PUSHL   R5
                                00000000G      00        02   FB 0002B     CALLS   #2, PRV$SETPRIV
                                               52        50   D0 00032     MOVL    R0, R2
                                00000000G      8F        52   D1 00035     CMPL    R2, #PRV$_INVNAM                 ; 2035
                                                         0F   12 0003C     BNEQ    2$
                                                         63   DD 0003E     PUSHL   TOKENPTR                         ; 2036
                                               7E        64   3C 00040     MOVZWL  TOKENLEN, -(SP)
                                                         02   DD 00043     PUSHL   #2
                                          00000000G      8F   DD 00045     PUSHL   #UAF$_PRVNOTFND
                                                         16   11 0004B     BRB     3$
                                00000000G      8F        52   D1 0004D 2$:       CMPL    R2, #PPV$_NOTUNQ         ; 2041
                                                         C1   12 00054     BNEQ    1$
                                                         63   DD 00056     PUSHL   TOKENPTR                         ; 2042
                                               7E        64   3C 00058     MOVZWL  TOKENLEN, -(SP)
                                                         02   DD 0005B     PUSHL   #2
                                          00000000G      8F   DD 0005D     PUSHL   #UAF$_PRVNOTUNQ
                                00000000G      00        04   FB 00063 3$:       CALLS   #4, LIB$SIGNAL
                                                              04 0006A     RET
                                               50        01   D0 0006B 4$:       MOVL    #1, R0                   ; 2048
                                                              04 0006E     RET                                      ; 2049

; Routine Size:  111 bytes,    Routine Base:  $CODE$ + 0F4F
```

```
;  1899          2050  1 %sbttl 'getuic - get user identification code'
;  1900          2051  1 global routine getuic =
;  1901          2052  2 begin
;  1902          2053  2
;  1903          2054  2 !++
;  1904          2055  2 !
;  1905          2056  2 ! FUNCTIONAL DESCRIPTION:
;  1906          2057  2 !
;  1907          2058  2 !       Process UIC specification
;  1908          2059  2 !
;  1909          2060  2 ! INPUTS:
;  1910          2061  2 !
;  1911          2062  2 !       none
;  1912          2063  2 !
;  1913          2064  2 ! OUTPUTS:
;  1914          2065  2 !
;  1915          2066  2 !       none
;  1916          2067  2 !
;  1917          2068  2 ! ROUTINE VALUE:
;  1918          2069  2 !
;  1919          2070  2 !       none
;  1920          2071  2 !--
;  1921          2072  2
;  1922          2073  2 !
;  1923          2074  2 ! Parse UIC and return error if invalid.
;  1924          2075  2 ! Do not allow wild card group or member numbers
;  1925          2076  2 !
;  1926          2077  2
;  1927          2078  2 if not parse_uic (recbuf[uaf$w_grp], recbuf[uaf$w_mem], false)
;  1928          2079  2 then return LIB$SIGNAL(UAF$_UICERR, 2, .tokenlen, .tokenptr);
;  1929          2080  2
;  1930          2081  2 true
;  1931          2082  1 end;


                                                  0000 00000         .ENTRY  GETUIC, Save nothing                    ;  2051
                                              7E  D4 00002         CLRL    -(SP)                                   ;  2078
                                  00000000G   00  9F 00004         PUSHAB  RECBUF+36
                                  00000000G   00  9F 0000A         PUSHAB  RECBUF+38
                      00000000V   00            03  FB 00010         CALLS   #3, PARSE_UIC
                                  1D            50  E8 00017         BLBS    R0, 1$
                                  00000000G   00  DD 0001A         PUSHL   TOKENPTR                                ;  2079
                                  7E 00000000G  00  3C 00020         MOVZWL  TOKENLEN, -(SP)
                                              02  DD 00027         PUSHL   #2
                                  00000000G   8F  DD 00029         PUSHL   #UAF$_UICERR
                      00000000G   00            04  FB 0002F         CALLS   #4, LIB$SIGNAL
                                                04 00036         RET
                                  50            01  D0 00037 1$:     MOVL    #1, R0                              ;  2082
                                                04 0003A         RET

; Routine Size:  59 bytes,    Routine Base:  $CODE$ + 0FBE
```

```
;  1933          2083  1 %sbttl 'parse_uic - obtain UIC group and member'
;  1934          2084  1 global routine parse_uic (retgrp, retmem, allow_wild) =
;  1935          2085  2 begin
;  1936          2086  2
;  1937          2087  2 !++
;  1938          2088  2 !
;  1939          2089  2 ! FUNCTIONAL DESCRIPTION:
;  1940          2090  2 !
;  1941          2091  2 !       Routine to parse and validate a UIC or directory string.
;  1942          2092  2 !       The UIC is returned as group and member elements.
;  1943          2093  2 !
;  1944          2094  2 ! INPUTS:
;  1945          2095  2 !
;  1946          2096  2 !       RETGRP          - address of a word to receive the group number
;  1947          2097  2 !       RETMEM          - address of a word to receive the member number
;  1948          2098  2 !       ALLOW_WILD      - TRUE => allow wild card group/member numbers
;  1949          2099  2 !
;  1950          2100  2 ! IMPLICIT INPUTS:
;  1951          2101  2 !
;  1952          2102  2 !       TOKENPTR - address of first character past delimiter
;  1953          2103  2 !
;  1954          2104  2 ! OUTPUTS:
;  1955          2105  2 !
;  1956          2106  2 !       RETGRP and RETMEM receive group and member numbers if
;  1957          2107  2 !       no errors are encountered.
;  1958          2108  2 !
;  1959          2109  2 ! IMPLICIT OUTPUTS:
;  1960          2110  2 !
;  1961          2111  2 !       none
;  1962          2112  2 !
;  1963          2113  2 ! ROUTINE VAUE:
;  1964          2114  2 !
;  1965          2115  2 !       true -> no errors found
;  1966          2116  2 !       false -> error in UIC specification
;  1967          2117  2 !
;  1968          2118  2 ! SIDE EFFECTS:
;  1969          2119  2 !
;  1970          2120  2 !       None
;  1971          2121  2 !--
;  1972          2122  2
;  1973          2123  2 !
;  1974          2124  2 ! Crank UIC through tparse
;  1975          2125  2 !
;  1976          2126  2 tparse_block [tpa$l_stringcnt] = .tokenlen;
;  1977          2127  2 tparse_block [tpa$l_stringptr] = .tokenptr;
;  1978          2128  2
;  1979          2129  2 if not lib$tparse (tparse_block, uic_states, uic_keys)
;  1980          2130  2 then return false;
;  1981          2131  2 if .converted_uic<00,16> gtru uic$k_wild_member
;  1982          2132  2 or .converted_uic<16,16> gtru uic$k_wild_group
;  1983          2133  2 then return false;
;  1984          2134  2 if not .allow_wild
;  1985          2135  2 then
;  1986          2136  2     if .converted_uic<00,16> eqlu uic$k_wild_member
;  1987          2137  2     or .converted_uic<16,16> eqlu uic$k_wild_group
;  1988          2138  2     then return false;
;  1989          2139  2
```

```
;  1990          2140  2 !
;  1991          2141  2 ! UIC was successfully parsed.  Return correct data, plus true indicaton.
;  1992          2142  2 !
;  1993          2143  2
;  1994          2144  2 (.retgrp)<0,16> = .converted_uic<16,16>;
;  1995          2145  2 (.retmem)<0,16> = .converted_uic<0,16>;
;  1996          2146  2 return true;
;  1997          2147  1 end;

                                    0004 00000          .ENTRY  PARSE_UIC, Save R2                      ; 2084
                          52 00000000'  00  9E 00002    MOVAB   CONVERTED_UIC, R2                       ;
                       E4 A2 00000000G  00  3C 00009    MOVZWL  TOKENLEN, TPARSE_BLOCK+8                ; 2126
                       E8 A2 00000000G  00  D0 00011    MOVL    TOKENPTR, TPARSE_BLOCK+12               ; 2127
                          00000000'  00  9F 00019    PUSHAB  UIC_KEYS                                ; 2129
                          00000000'  00  9F 0001F    PUSHAB  UIC_STATES
                                DC  A2  9F 00025    PUSHAB  TPARSE_BLOCK
                00000000G  00         03  FB 00028    CALLS   #3, LIB$TPARSE
                          29          50  E9 0002F    BLBC    R0, 2$
                          50    02    A2  3C 00032    MOVZWL  CONVERTED_UIC+2, R0                     ; 2132
                        3FFF  8F        50  B1 00036    CMPW    R0, #16383
                                        1E  1A 0003B    BGTRU   2$
                          0E    0C    AC  E8 0003D    BLBS    ALLOW_WILD, 1$                          ; 2134
                        FFFF  8F        62  B1 00041    CMPW    CONVERTED_UIC, #65535                   ; 2136
                                        13  13 00046    BEQL    2$
                        3FFF  8F        50  B1 00048    CMPW    R0, #16383                              ; 2137
                                        0C  13 0004D    BEQL    2$
                          04  BC        50  B0 0004F 1$:    MOVW    R0, @RETGRP                             ; 2144
                          08  BC        62  B0 00053    MOVW    CONVERTED_UIC, @RETMEM                  ; 2145
                          50            01  D0 00057    MOVL    #1, R0                                  ; 2146
                                            04 0005A    RET
                                        50  D4 0005B 2$:    CLRL    R0                                      ; 2147
                                            04 0005D    RET

; Routine Size:  94 bytes,    Routine Base:  $CODE$ + 0FF9
```

```
;1999     2148  1 %sbttl 'parse_wild - parse wildcarded user specification'
;2000     2149  1 global routine parse_wild (desc_addr,nulldefault) =
;2001     2150  2 begin
;2002     2151  2
;2003     2152  2 !++
;2004     2153  2 !
;2005     2154  2 ! FUNCTIONAL DESCRIPTION:
;2006     2155  2 !
;2007     2156  2 !       Parse one of six methods by which User Authorization File
;2008     2157  2 !       records may be specified.  This routine defines boolean
;2009     2158  2 !       variables for input to the WILD_USER routine.
;2010     2159  2 !
;2011     2160  2 ! INPUTS:
;2012     2161  2 !
;2013     2162  2 !       DESCADDR        Address of the token string descriptor
;2014     2163  2 !       NULLDEFAULT     TRUE => A null user specification defaults to *
;2015     2164  2 !
;2016     2165  2 ! IMPLICIT INPUTS:
;2017     2166  2 !
;2018     2167  2 !       The parsing variables NEXTTOKEN, TOKENLEN, and TOKENPTR are modified.
;2019     2168  2 !
;2020     2169  2 ! OUTPUTS:
;2021     2170  2 !
;2022     2171  2 !       none
;2023     2172  2 !
;2024     2173  2 ! IMPLICIT OUTPUTS:
;2025     2174  2 !
;2026     2175  2 !       The following boolean variables are decided:
;2027     2176  2 !
;2028     2177  2 !       UIC_FLAG - UIC form (instead of username)
;2029     2178  2 !       GRP_WILD - Group wild card (implies UIC_FLAG)
;2030     2179  2 !       MEM_WILD - Member wild card (implies UIC_FLAG)
;2031     2180  2 !       STR_WILD - all users alphabetically (implies NOT UIC_FLAG)
;2032     2181  2 !
;2033     2182  2 !       RECBUF - The appropriate keys are initialized
;2034     2183  2 !
;2035     2184  2 ! ROUTINE VALUE:
;2036     2185  2 !
;2037     2186  2 !       If the syntax does not follow one of the six methods the routine
;2038     2187  2 !       returns FALSE.  The outputs are meaningful only when the value
;2039     2188  2 !       returned is TRUE.
;2040     2189  2 !
;2041     2190  2 !--
;2042     2191  2
;2043     2192  2
;2044     2193  2 ! Identify the next token in the command buffer.
;2045     2194  2 !
;2046     2195  2
;2047     2196  2 if not cli$present (.desc_addr) or
;2048     2197  2    not cli$get_value (.desc_addr, tokendsc) or
;2049     2198  2    .tokenlen eq. 0
;2050     2199  2 then
;2051     2200  2     if not .nulldefault
;2052     2201  2     then
;2053     2202  2         return LIB$SIGNAL(UAF$_NOUSERSPEC)
;2054     2203  2     else
;2055     2204  3         begin
```

```
; 2056          2205  3          uic_flag = false;
; 2057          2206  3          grp_wild = false;
; 2058          2207  3          mem_wild = false;
; 2059          2208  3          str_wild = true;
; 2060          2209  3          match_token = '*';
; 2061          2210  3          match_tokenlen = 1;
; 2062          2211  3          return true;
; 2063          2212  3          end
; 2064          2213  2      else
; 2065          2214  3          begin
; 2066          2215  3          tokenlen = .tokenlen;
; 2067          2216  3          tokenptr = .tokenptr;
; 2068          2217  3          end;
; 2069          2218  2
; 2070          2219  2
; 2071          2220  2      !
; 2072          2221  2      ! Decide whether a UIC or a Username form was used.
; 2073          2222  2      !
; 2074          2223  2
; 2075          2224  2      if .(.tokenptr)<0,8> eql '['
; 2076          2225  2      then
; 2077          2226  3          begin
; 2078          2227  3
; 2079          2228  3          if not parse_wild_uic ( .tokenlen, .tokenptr )
; 2080          2229  3          then
; 2081          2230  3              return LIB$SIGNAL(UAF$_INVUSERSPEC, 2, .tokenlen, .tokenptr);
; 2082          2231  3
; 2083          2232  3          end
; 2084          2233  2      else
; 2085          2234  3          begin
; 2086          2235  3          uic_flag = false;
; 2087          2236  3          grp_wild = false;
; 2088          2237  3          mem_wild = false;
; 2089          2238  3
; 2090          2239  3          !
; 2091          2240  3          ! Decide whether or not the Username is wildcarded.
; 2092          2241  3          !
; 2093          2242  3
; 2094          2243  3          if not ch$fail (ch$find_ch (.tokenlen, .tokenptr, '*'))
; 2095          2244  3                or
; 2096          2245  3             not ch$fail (ch$find_ch (.tokenlen, .tokenptr, '%'))
; 2097          2246  3 !              OR
; 2098          2247  3 !           .by_account
; 2099          2248  3          then
; 2100          2249  4              begin
; 2101          2250  4              match_tokenlen = .tokenlen;
; 2102          2251  4              ch$move (.tokenlen, .tokenptr, match_token);
; 2103          2252  4              str_wild = true;
; 2104          2253  4              end
; 2105          2254  3          else
; 2106          2255  4              begin
; 2107          2256  4              str_wild = false;
; 2108          2257  4              ch$copy (.tokenlen, .tokenptr, %char (' '),
; 2109          2258  4                               uaf$s_username, recbuf[uaf$t_username]);
; 2110          2259  3              end;
; 2111          2260  2          end;
; 2112          2261  2
```

```
; 2113          2262  2 true
; 2114          2263  1 end;


                                          0FFC 00000          .ENTRY  PARSE_WILD, Save R2,R3,R4,R5,R6,R7,R8,R9,-   ; 2149
                                                                      R10,R11                                          ;
                    5B 00000000G   00  9E 00002               MOVAB   UIC_FLAG, R11                                    ;
                    5A 00000000G   00  9E 00009               MOVAB   LIB$SIGNAL, R10                                  ;
                    59 00000000G   00  9E 00010               MOVAB   STR_WILD, R9                                     ;
                    58 00000000G   00  9E 00017               MOVAB   TOKENLEN, R8                                     ;
                            04     AC  DD 0001E               PUSHL   DESC_ADDR                                        ; 2196
         00000000G  00             01  FB 00021               CALLS   #1, CLI$PRESENT                                  ;
                    17             50  E9 00028               BLBC    R0, 1$                                           ;
                       00000000G   00  9F 0002B               PUSHAB  TOKENDSC                                         ; 2197
                            04     AC  DD 00031               PUSHL   DESC_ADDR                                        ;
         00000000G  00             02  FB 00034               CALLS   #2, CLI$GET_VALUE                                ;
                    04             50  E9 0003B               BLBC    R0, 1$                                           ;
                                   68  B5 0003E               TSTW    TOKENLEN                                         ; 2198
                                   2F  12 00040               BNEQ    3$                                               ;
                    0A      08     AC  E8 00042 1$:           BLBS    NULLDEFAULT, 2$                                  ; 2200
                       00000000G   8F  DD 00046               PUSHL   #UAF$_NOUSERSPEC                                 ; 2202
                    6A             01  FB 0004C               CALLS   #1, LIB$SIGNAL                                   ;
                                       04 0004F               RET                                                      ; 2204
                                   6B  D4 00050 2$:           CLRL    UIC_FLAG                                         ; 2205
                       00000000G   00  D4 00052               CLRL    GRP_WILD                                         ; 2206
                       00000000G   00  D4 00058               CLRL    MEM_WILD                                         ; 2207
                    69             01  D0 0005E               MOVL    #1, STR_WILD                                     ; 2208
         00000000G  00             2A  D0 00061               MOVL    #42, MATCH_TOKEN                                 ; 2209
         00000000G  00             01  D0 00068               MOVL    #1, MATCH_TOKENLEN                               ; 2210
                    79                 11 0006F               BRB     9$                                               ; 2211
                    57             68  3C 00071 3$:           MOVZWL  TOKENLEN, R7                                     ; 2228
                    56 00000000G   00  D0 00074               MOVL    TOKENPTR, R6                                     ; 2224
            5B      8F             66  91 0007B               CMPB    (R6), #91                                        ;
                                   23  12 0007F               BNEQ    4$                                               ;
                                   56  DD 00081               PUSHL   R6                                               ; 2228
                                   57  DD 00083               PUSHL   R7                                               ;
         00000000V  00             02  FB 00085               CALLS   #2, PARSE_WILD_UIC                               ;
                    5B             50  E8 0008C               BLBS    R0, 9$                                           ;
                       00000000G   00  DD 0008F               PUSHL   TOKENPTR                                         ; 2230
                    7E             68  3C 00095               MOVZWL  TOKENLEN, -(SP)                                  ;
                                   02  DD 00098               PUSHL   #2                                               ;
                       00000000G   8F  DD 0009A               PUSHL   #UAF$_INVUSERSPEC                                ;
                    6A             04  FB 000A0               CALLS   #4, LIB$SIGNAL                                   ;
                                       04 000A3               RET                                                      ;
                                   6B  D4 000A4 4$:           CLRL    UIC_FLAG                                         ; 2235
                       00000000G   00  D4 000A6               CLRL    GRP_WILD                                         ; 2236
                       00000000G   00  D4 000AC               CLRL    MEM_WILD                                         ; 2237
     66             57             2A  3A 000B2               LOCC    #42, R7, (R6)                                    ; 2243
                                   02  12 000B6               BNEQ    5$                                               ;
                                   51  D4 000B8               CLRL    R1                                               ;
                                   51  D5 000BA 5$:           TSTL    R1                                               ;
                                   0C  12 000BC               BNEQ    7$                                               ;
     66             57             25  3A 000BE               LOCC    #37, R7, (R6)                                    ; 2245
                                   02  12 000C2               BNEQ    6$                                               ;
```

```
                                                 51  D4 000C4          CLRL    R1                                                      ;
                                                 51  D5 000C6 6$:      TSTL    R1                                                      ;
                                                 14  13 000C8          BEQL    8$                                                      ;
                                 00000000G   00  57  D0 000CA 7$:      MOVL    R7, MATCH_TOKENLEN                                      ; 2250
                        00000000G   00           66  57  28 000D1      MOVC3   R7, (R6), MATCH_TOKEN                                   ; 2251
                                                 69  01  D0 000D9      MOVL    #1, STR_WILD                                            ; 2252
                                                     0C  11 000DC      BRB     9$                                                      ; 2243
                                                 69  D4 000DE 8$:      CLRL    STR_WILD                                                ; 2256
                20              20               66  57  2C 000E0      MOVC5   R7, (R6), #32, #32, RECBUF+4                            ; 2258
                                                         00  000E5
                                         00000000G   50  01  D0 000EA 9$:      MOVL    #1, R0                                          ; 2263
                                                         04  000ED      RET                                                             ;

; Routine Size:  238 bytes,    Routine Base:  $CODE$ + 1057
```

```
; 2116          2264  1 %sbttl 'parse_wild_uic - parse wildcarded UIC'
; 2117          2265  1 global routine parse_wild_uic ( strlen, strptr ) =
; 2118          2266  2 begin
; 2119          2267  2
; 2120          2268  2 !++
; 2121          2269  2 !
; 2122          2270  2 ! FUNCTIONAL DESCRIPTION:
; 2123          2271  2 !
; 2124          2272  2 !       Parse the uic methods by which User Authorization File
; 2125          2273  2 !       records may be specified.  This routine defines boolean
; 2126          2274  2 !       variables for input to the WILD_USER routine.
; 2127          2275  2 !
; 2128          2276  2 ! INPUTS:
; 2129          2277  2 !
; 2130          2278  2 !       STRLEN  - Length of UIC string
; 2131          2279  2 !       STRPTR  - Address of UIC string
; 2132          2280  2 !
; 2133          2281  2 ! IMPLICIT INPUTS:
; 2134          2282  2 !
; 2135          2283  2 !
; 2136          2284  2 ! OUTPUTS:
; 2137          2285  2 !
; 2138          2286  2 !       True  - success
; 2139          2287  2 !       False - failure
; 2140          2288  2 !
; 2141          2289  2 ! IMPLICIT OUTPUTS:
; 2142          2290  2 !
; 2143          2291  2 !       The following boolean variables are decided:
; 2144          2292  2 !
; 2145          2293  2 !       UIC_FLAG - UIC form (instead of username)
; 2146          2294  2 !       GRP_WILD - Group wild card (implies UIC_FLAG)
; 2147          2295  2 !       MEM_WILD - Member wild card (implies UIC_FLAG)
; 2148          2296  2 !       STR_WILD - all users alphabetically (implies NOT UIC_FLAG)
; 2149          2297  2 !
; 2150          2298  2 !       RECBUF - The appropriate keys are initialized
; 2151          2299  2 !
; 2152          2300  2 ! ROUTINE VALUE:
; 2153          2301  2 !
; 2154          2302  2 !       If the syntax does not follow one of the six methods the routine
; 2155          2303  2 !       returns FALSE.  The outputs are meaningful only when the value
; 2156          2304  2 !       returned is TRUE.
; 2157          2305  2 !
; 2158          2306  2 !--
; 2159          2307  2
; 2160          2308  2 uic_flag = true;
; 2161          2309  2 str_wild = false;
; 2162          2310  2
; 2163          2311  2 !
; 2164          2312  2 ! Crank UIC through tparse
; 2165          2313  2 !
; 2166          2314  2 tparse_block [tpa$l_stringcnt] = .strlen;
; 2167          2315  2 tparse_block [tpa$l_stringptr] = .strptr;
; 2168          2316  2
; 2169          2317  2 if not lib$tparse (tparse_block, uic_states, uic_keys)
; 2170          2318  2 then
; 2171          2319  2     return false ;
; 2172          2320  2
```

```
;  2173          2321  2 !
;  2174          2322  2 ! Decide whether or not the Group subfield is wildcarded.
;  2175          2323  2 !
;  2176          2324  2 if .converted_uic<16,16> eql uic$k_wild_group
;  2177          2325  2 then
;  2178          2326  3     begin
;  2179          2327  3     grp_wild = true;
;  2180          2328  3     recbuf[uaf$w_grp] = 0;
;  2181          2329  3     end
;  2182          2330  3
;  2183          2331  2 else
;  2184          2332  3     begin
;  2185          2333  3     grp_wild = false;
;  2186          2334  3     recbuf[uaf$w_grp] = .converted_uic<16,16>;
;  2187          2335  2     end;
;  2188          2336  2
;  2189          2337  2 !
;  2190          2338  2 ! Decide whether or not the Member subfield is wildcarded.
;  2191          2339  2 !
;  2192          2340  2
;  2193          2341  2 if .converted_uic<0,16> eql uic$k_wild_member
;  2194          2342  2 then
;  2195          2343  3     begin
;  2196          2344  3     mem_wild = true;
;  2197          2345  3
;  2198          2346  3     !
;  2199          2347  3     ! Whether it is wildcarded or not this subfield is initialized
;  2200          2348  3     ! for the wildcard processor.
;  2201          2349  3     !
;  2202          2350  3
;  2203          2351  3     recbuf[uaf$w_mem] = 0;
;  2204          2352  3     end
;  2205          2353  2 else
;  2206          2354  3     begin
;  2207          2355  3     mem_wild = false;
;  2208          2356  3     recbuf[uaf$w_mem] = .converted_uic<0,16>;
;  2209          2357  2     end;
;  2210          2358  2
;  2211          2359  2 return true ;
;  2212          2360  1 end ;

                                                003C 00000          .ENTRY  PARSE_WILD_UIC, Save R2,R3,R4,R5                ; 2265
                                    55 00000000G   00   9E 00002    MOVAB   MEM_WILD, R5
                                    54 00000000G   00   9E 00009    MOVAB   GRP_WILD, R4
                                    53 00000000G   00   9E 00010    MOVAB   RECBUF+38, R3
                                    52 00000000'   00   9E 00017    MOVAB   CONVERTED_UIC, R2
                00000000G   00                     01   D0 0001E    MOVL    #1, UIC_FLAG                                    ; 2308
                                00000000G          00   D4 00025    CLRL    STR_WILD                                        ; 2309
                      E4    A2          04         AC   7D 0002B    MOVQ    STRLEN, TPARSE_BLOCK+8                          ; 2314
                                00000000'          00   9F 00030    PUSHAB  UIC_KEYS                                        ; 2317
                                00000000'          00   9F 00036    PUSHAB  UIC_STATES
                                        DC         A2   9F 0003C    PUSHAB  TPARSE_BLOCK
                00000000G   00                     03   FB 0003F    CALLS   #3, LIB$TPARSE
```

```
              33                50 E9 00046            BLBC    R0, 5$                     ;
              50          02    A2 3C 00049            MOVZWL  CONVERTED_UIC+2, R0        ;  2324
        3FFF  8F                50 B1 0004D            CMPW    R0, #16383                 ;
                                07 12 00052            BNEQ    1$                         ;
              64                01 D0 00054            MOVL    #1, GRP_WILD               ;  2327
                                63 B4 00057            CLRW    RECBUF+38                  ;  2328
                                05 11 00059            BRB     2$                         ;  2324
                                64 D4 0005B 1$:        CLRL    GRP_WILD                   ;  2333
              63                50 B0 0005D            MOVW    R0, RECBUF+38              ;  2334
              50                62 3C 00060 2$:        MOVZWL  CONVERTED_UIC, R0          ;  2341
        FFFF  8F                50 B1 00063            CMPW    R0, #65535                 ;
                                08 12 00068            BNEQ    3$                         ;
              65                01 D0 0006A            MOVL    #1, MEM_WILD               ;  2344
                          FE    A3 B4 0006D            CLRW    RECBUF+36                  ;  2351
                                06 11 00070            BRB     4$                         ;  2341
                                65 D4 00072 3$:        CLRL    MEM_WILD                   ;  2355
          FE  A3                50 B0 00074            MOVW    R0, RECBUF+36              ;  2356
              50                01 D0 00078 4$:        MOVL    #1, R0                     ;  2359
                                   04 0007B            RET                                ;
                                50 D4 0007C 5$:        CLRL    R0                         ;  2360
                                   04 0007E            RET                                ;

; Routine Size:  127 bytes,    Routine Base:  $CODE$ + 1145
```

```
;  2214          2361  1 %sbttl 'getstring - get string from input'
;  2215          2362  1 global routine getstring (addr, maxsize, type) =
;  2216          2363  2 begin
;  2217          2364  2
;  2218          2365  2 !++
;  2219          2366  2 !
;  2220          2367  2 ! FUNCTIONAL DESCRIPTION:
;  2221          2368  2 !
;  2222          2369  2 !       This routine reads a string variable and stores it as
;  2223          2370  2 !       a blank filled or counted string in the field supplied.
;  2224          2371  2 !       The first character is checked to see if it is a quote
;  2225          2372  2 !       or double quote and if so, a different terminator
;  2226          2373  2 !       set is used so that embedded blanks may be contained
;  2227          2374  2 !       in the string.
;  2228          2375  2 !
;  2229          2376  2 ! INPUTS:
;  2230          2377  2 !
;  2231          2378  2 !       ADDR - address to store string
;  2232          2379  2 !       MAXSIZE - size of field in which string will be stored.
;  2233          2380  2 !                 If this is a counted string, the maximum input
;  2234          2381  2 !                 acceptable is MAXSIZE - 1
;  2235          2382  2 !       TYPE - string type, either COUNTED_STRING or FILLED_STRING
;  2236          2383  2 !
;  2237          2384  2 ! IMPLICIT INPUTS:
;  2238          2385  2 !
;  2239          2386  2 !       none
;  2240          2387  2 !
;  2241          2388  2 ! OUTPUTS:
;  2242          2389  2 !
;  2243          2390  2 !       none
;  2244          2391  2 !
;  2245          2392  2 ! IMPLICIT OUTPUTS:
;  2246          2393  2 !
;  2247          2394  2 !       none
;  2248          2395  2 !
;  2249          2396  2 ! ROUTINE VALUE:
;  2250          2397  2 !
;  2251          2398  2 !       true -> string inserted successfully
;  2252          2399  2 !       false -> string too long
;  2253          2400  2 !
;  2254          2401  2 ! SIDE EFFECTS:
;  2255          2402  2 !
;  2256          2403  2 !       none
;  2257          2404  2 !--
;  2258          2405  2
;  2259          2406  2
;  2260          2407  2 if .type eql counted_string
;  2261          2408  2 then
;  2262          2409  2     maxsize = .maxsize - 1;
;  2263          2410  2
;  2264          2411  2 if .tokenlen gtr .maxsize
;  2265          2412  2 then
;  2266          2413  2     return
;  2267          2414  2         LIB$SIGNAL(UAF$_INVSTR, 2, .tokenlen, .tokenptr);
;  2268          2415  2
;  2269          2416  2 if .type eql counted_string
;  2270          2417  2 then
```

```
;  2271          2418  3      begin
;  2272          2419  3      (.addr)<0,8> = .tokenlen;
;  2273          2420  3      ch$copy (.tokenlen, .tokenptr, %char (blank),
;  2274          2421  3                  .maxsize-1, .addr+1);
;  2275          2422  3      end
;  2276          2423  2  else
;  2277          2424  2      ch$copy (.tokenlen, .tokenptr, %char (blank),
;  2278          2425  2                  .maxsize, .addr);
;  2279          2426  2  return true;
;  2280          2427  1  end;


                                                00FC 00000          .ENTRY  GETSTRING, Save R2,R3,R4,R5,R6,R7      ;  2362
                                    57 00000000G  00   9E C0002     MOVAB   TOKENLEN, R7                           ;
                                    56 00000000G  00   9E 00009     MOVAB   TOKENPTR, R6                           ;
                                                  54   D4 00010     CLRL    R4                                     ;  2407
                                    01        0C  AC   D1 00012     CMPL    TYPE, #1                               ;
                                                  05   12 00016     BNEQ    1$                                     ;
                                                  54   D6 00018     INCL    R4                                     ;
                                              08  AC   D7 0001A     DECL    MAXSIZE                                ;  2409
                                    50            67   3C 0001D 1$: MOVZWL  TOKENLEN, R0                          ;  2411
                                08  AC            50   D1 00020     CMPL    R0, MAXSIZE                            ;
                                                  14   15 00024     BLEQ    2$                                     ;
                                                  66   DD 00026     PUSHL   TOKENPTR                               ;  2414
                                                  50   DD 00028     PUSHL   R0                                     ;
                                                  02   DD 0002A     PUSHL   #2                                     ;
                                       00000000G  8F   DD 0002C     PUSHL   #UAF$_INVSTR                           ;
                                00000000G  00         04   FB 00032     CALLS   #4, LIB$SIGNAL                     ;
                                                           04 00039     RET                                        ;
                                    50        04  AC   D0 0003A 2$: MOVL    ADDR, R0                               ;  2419
                                    52            67   3C 0003E     MOVZWL  TOKENLEN, R2                           ;
                                    53            66   D0 00041     MOVL    TOKENPTR, R3                           ;  2420
                                    11            54   E9 00044     BLBC    R4, 3$                                 ;
                                    60            52   90 00047     MOVB    R2, (R0)                               ;  2419
                              51        08  AC   01   C3 0004A     SUBL3   #1, MAXSIZE, R1                        ;  2421
                        51    20            63   52   2C 0004F     MOVC5   R2, (R3), #32, R1, 1(R0)               ;
                                                01   A0 00054                                                    ;
                                                  07   11 00056     BRB     4$                                     ;  2416
                        08  AC        20        63   52   2C 00058 3$: MOVC5   R2, (R3), #32, MAXSIZE, (R0)      ;  2425
                                                  60        0005E                                                  ;
                                    50            01   D0 0005F 4$: MOVL    #1, R0                                 ;  2426
                                                           04 00062     RET                                        ;  2427

; Routine Size:  99 bytes,    Routine Base:  $CODE$ + 11C4
```

```
;  2282          2428  1 %sbttl 'getval - get numeric value from user'
;  2283          2429  1 global routine getval (addr, size) =
;  2284          2430  2 begin
;  2285          2431  2
;  2286          2432  2 !++
;  2287          2433  2 !
;  2288          2434  2 ! FUNCTIONAL DESCRIPTION:
;  2289          2435  2 !
;  2290          2436  2 !       Routine to return binary value for next ASCII decimal
;  2291          2437  2 !       number in the input stream.
;  2292          2438  2 !
;  2293          2439  2 ! INPUTS:
;  2294          2440  2 !
;  2295          2441  2 !       ADDR - address to return converted value
;  2296          2442  2 !       SIZE - size in bits of field in which to return value
;  2297          2443  2 !
;  2298          2444  2 ! IMPLICIT INPUTS:
;  2299          2445  2 !
;  2300          2446  2 !       NEXTTOKEN - contains address of delimiter preceeding the number
;  2301          2447  2 !
;  2302          2448  2 ! OUTPUTS:
;  2303          2449  2 !
;  2304          2450  2 !       field described by ADDR and SIZE receives binary value
;  2305          2451  2 !
;  2306          2452  2 ! IMPLICIT OUTPUTS:
;  2307          2453  2 !
;  2308          2454  2 !       none
;  2309          2455  2 !
;  2310          2456  2 ! ROUTINE VALUE:
;  2311          2457  2 !
;  2312          2458  2 !       true -> value converted successfully
;  2313          2459  2 !       false  > non-numeric character encounted before next
;  2314          2460  2 !              delimiter found
;  2315          2461  2 !
;  2316          2462  2 ! SIDE EFFECTS:
;  2317          2463  2 !
;  2318          2464  2 !       If error is encountered, an error message will be printed.
;  2319          2465  2 !
;  2320          2466  2 !--
;  2321          2467  2
;  2322          2468  2 local
;  2323          2469  2         value;                          ! for value returned
;  2324          2470  2                                         ! by CVTNUM routine
;  2325          2471  2
;  2326          2472  2 !
;  2327          2473  2 ! Check for no value supplied
;  2328          2474  2 !
;  2329          2475  2
;  2330          2476  2 if .tokenlen eql 0
;  2331          2477  2 then
;  2332          2478  2     return LIB$SIGNAL(UAF$_NOARG, 2, .tokenlen, .tokenptr);
;  2333          2479  2
;  2334          2480  2 !
;  2335          2481  2 ! Convert to decimal. Report error if any non-numerics encountered.
;  2336          2482  2 !
;  2337          2483  2
;  2338          2484  2 if not cvtnum (.tokenlen, .tokenptr, 10, value)
```

```
;  2339          2485  2 then
;  2340          2486  2     return LIB$SIGNAL(UAF$_BADVALUE, 2, .tokenlen, .tckenptr);
;  2341          2487  2
;  2342          2488  2 !
;  2343          2489  2 ! Check that the value supplied is within the size of
;  2344          2490  2 ! the field to be stored.
;  2345          2491  2 !
;  2346          2492  2
;  2347          2493  2 if .value<.size - 1, 32 - .size + 1> neq 0
;  2348          2494  2 then
;  2349          2495  2     return LIB$SIGNAL(UAF$_VALTOOBIG, 2, .tokenlen, .tokenptr);
;  2350          2496  2
;  2351          2497  2 !
;  2352          2498  2 ! Finally, store the value in field specified .
;  2353          2499  2 !
;  2354          2500  2
;  2355          2501  2 (.addr)<0, .size> = .value;
;  2356          2502  2
;  2357          2503  2 return true;
;  2358          2504  1 end;


                                                001C 00000          .ENTRY  GETVAL, Save R2,R3,R4                   ; 2429
                                    54 0000[0000G   00   9E 00002   MOVAB   TOKENLEN, R4                            ;
                                    53 00000000G    00   9E 00009   MOVAB   TOKENPTR, R3                            ;
                                    5E              04   C2 00010   SUBL2   #4, SP                                 ;
                                    50              64   3C 00013   MOVZWL  TOKENLEN, R0                           ; 2476
                                                    0E   12 00016   BNEQ    1$                                      ;
                                                    63   DD 00018   PUSHL   TOKENPTR                               ; 2478
                                                    50   DD 0001A   PUSHL   R0                                      ;
                                                    02   DD 0001C   PUSHL   #2                                      ;
                                        00000000G   8F   DD 0001E   PUSHL   #UAF$_NOARG                            ;
                                                    40   11 00024   BRB     3$                                      ;
                                                    5E   DD 00026 1$: PUSHL   SP                                   ; 2484
                                                    0A   DD 00028   PUSHL   #10                                     ;
                                                    63   DD 0002A   PUSHL   TOKENPTR                               ;
                                              7E        64   3C 0002C   MOVZWL  TOKENLEN, -(SP)                     ;
                                    00000000V 00        04   FB 0002F   CALLS   #4, CVTNUM                         ;
                                              0F        50   E8 00036   BLBS    R0, 2$                              ;
                                                    63   DD 00039   PUSHL   TOKENPTR                               ; 2486
                                              7E        64   3C 0003B   MOVZWL  TOKENLEN, -(SP)                     ;
                                                    02   DD 0003E   PUSHL   #2                                      ;
                                        00000000G   8F   DD 00040   PUSHL   #UAF$_BADVALUE                         ;
                                                    1E   11 00046   BRB     3$                                      ;
                                52        08      AC        01   C3 00048 2$: SUBL3   #1, SIZE, R2                ; 2493
                                50                21        AC   C3 0004D   SUBL3   SIZE, #33, R0                  ;
                        51      6E                50        EF 00052   EXTZV   R2, R0, VALUE, R1                   ;
                                                    15   13 00057   BEQL    4$                                      ;
                                                    63   DD 00059   PUSHL   TOKENPTR                               ; 2495
                                              7E        64   3C 0005B   MOVZWL  TOKENLEN, -(SP)                     ;
                                                    02   DD 0005E   PUSHL   #2                                      ;
                                        00000000G   8F   DD 00060   PUSHL   #UAF$_VALTOOBIG                        ;
                                    00000000G   00      04   FB 00066 3$: CALLS   #4, LIB$SIGNAL                   ;
                                                        04 0006D   RET
```

```
      04   BC      08   AC            00          6E  F0 0006E 4$:     INSV    VALUE, #0, SIZE, @ADDR          ; 2501
                                      50          01  D0 00075         MOVL    #1, R0                          ; 2503
                                                  04 00078             RET                                     ; 2504

; Routine Size:  121 bytes,    Routine Base:  $CODE$ + 1227
```

```
;2360        2505  1 %sbttl 'cvtnum - convert decimal ascii to binary'
;2361        2506  1 routine cvtnum (size, adr, radix, valadr) =
;2362        2507  2 begin
;2363        2508  2
;2364        2509  2 !++
;2365        2510  2 !
;2366        2511  2 ! FUNCTIONAL DESCIRPTION:
;2367        2512  2 !
;2368        2513  2 !       Routine to convert ascii digits to binary.
;2369        2514  2 !
;2370        2515  2 ! INPUTS:
;2371        2516  2 !
;2372        2517  2 !       SIZE - number of digits in input buffer
;2373        2518  2 !       ADR - address of the buffer containing ascii digits
;2374        2519  2 !       RADIX - radix of number to be converted
;2375        2520  2 !       VALADR - address of longword to receive converted value
;2376        2521  2 !
;2377        2522  2 ! IMPLICIT INPUTS:
;2378        2523  2 !
;2379        2524  2 !       none
;2380        2525  2 !
;2381        2526  2 ! OUTPUTS:
;2382        2527  2 !
;2383        2528  2 !       longword pointed to by VALADR receives converted value
;2384        2529  2 !
;2385        2530  2 ! IMPLICIT OUTPUTS:
;2386        2531  2 !
;2387        2532  2 !       none
;2388        2533  2 !
;2389        2534  2 ! ROUTINE VALUE:
;2390        2535  2 !
;2391        2536  2 !       true -> value successfully converted
;2392        2537  2 !       false -> non-numeric encountered before end of string
;2393        2538  2 !
;2394        2539  2 ! SIDE EFFECTS:
;2395        2540  2 !
;2396        2541  2 !       none
;2397        2542  2 !--
;2398        2543  2
;2399        2544  2 bind
;2400        2545  2         sum = .valadr;                  ! address ref scaler
;2401        2546  2
;2402        2547  2
;2403        2548  2 sum = 0;
;2404        2549  2
;2405        2550  2 !
;2406        2551  2 ! Loop thorugh buffer.  Stop on end of string or non-decimal digit.
;2407        2552  2 !
;2408        2553  2
;2409        2554  3 incr i from .adr to  (.adr + .size - 1)
;2410        2555  2 do
;2411        2556  3     begin
;2412        2557  3     local
;2413        2558  3         digit,
;2414        2559  3         pointer;
;2415        2560  3
;2416        2561  3     digit = .(.i)<0,8>;
```

```
;  2417          2562  3
;  2418          2563  3      !
;  2419          2564  3      ! Check validity of digit depending on radix
;  2420          2565  3      !
;  2421          2566  3
;  2422          2567  3
;  2423          2568  3      if ch$fail (pointer = ch$find_ch (.radix, numbers, .digit))
;  2424          2569  3      then
;  2425          2570  3          return false
;  2426          2571  3      else
;  2427          2572  4          sum = (.sum*.radix) +  (.pointer - numbers)
;  2428          2573  2      end;
;  2429          2574  2
;  2430          2575  2  return true;
;  2431          2576  1  end;
```

```
                                                 003C 00000 CVTNUM:  .WORD   Save R2,R3,R4,R5                    ;  2506
                                      55 00000000'   00  9E 00002          MOVAB   NUMBERS, R5
                                                 10   BC  D4 00009          CLRL    @VALADR                     ;  2548
                                54        08   AC       04   AC  C1 0000C          ADDL3   SIZE, ADR, R4       ;  2554
                                53        08   AC            01  C3 00012          SUBL3   #1, ADR, I          ;  2568
                                                            21  11 00017          BRB     3$
                                                      50        63  9A 00019 1$:      MOVZBL  (I), DIGIT          ;  2561
                                65        0C   AC             50  3A 0001C          LOCC    DIGIT, RADIX, NUMBERS   ;  2568
                                                            02  12 00021          BNEQ    2$
                                                            51  D4 00023          CLRL    R1
                                                            51  D5 00025 2$:      TSTL    POINTER
                                                            19  13 00027          BEQL    4$
                                     52        10   BC       0C   AC  C5 00029          MULL3   RADIX, @VALADR, R2  ;  2572
                                                      50        65  9E 0002F          MOVAB   NUMBERS, R0
                                                      51        50  C2 00032          SUBL2   R0, R1
                                10   BC                   52        51  C1 00035          ADDL3   R1, R2, @VALADR
                                     DB                   53        54  F2 0003A 3$:      AOBLSS  R4, I, 1$       ;  2568
                                                      50        01  D0 0003E          MOVL    #1, R0              ;  2575
                                                                04 00041          RET
                                                            50  D4 00042 4$:      CLRL    R0                  ;  2576
                                                                04 00044          RET

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 12A0
```

```
; 2433        2577  1 %sbttl 'PASSWORD MACROS -- for N-ary passwords'
; 2434        2578  1
; 2435        2579  1 !++
; 2436        2580  1 !
; 2437        2581  1 ! FUNCTIONAL DESCRIPTION:
; 2438        2582  1 !
; 2439        2583  1 !       These macros and the routines that follow allow for
; 2440        2584  1 !       the use of any number of passwords. At the time of
; 2441        2585  1 !       this writing, only two (primary and secondary) pass-
; 2442        2586  1 !       words were in effect. When a change is made to the
; 2443        2587  1 !       number of passwords, it will be necessary, only, to
; 2444        2588  1 !       update the argument list macros below.
; 2445        2589  1 !
; 2446        2590  1 !--
; 2447        2591  1
; 2448        2592  1 macro
; 2449        2593  1
; 2450        2594  1     $NARY_LIST    = 'PRIMARY', '', 'SECONDARY', '2' %,
; 2451        2595  1     $N_LIST       =            '',             '2' %;
; 2452        2596  1
; 2453        2597  1 !
; 2454        2598  1 !       The remainder of the text will need no modification for N-ary passwords
; 2455        2599  1 !
; 2456        2600  1
; 2457        2601  1 macro $PASSWORD(n) = %name('PASSWORD_',n) %;
; 2458        2602  1 macro $PRESENT(n) = %name('PRESENT_',n) %;
; 2459        2603  1
; 2460      M 2604  1 macro OWN_PASSWORD[n] =
; 2461      M 2605  1     own $PASSWORD(n): block[DSC$K_D_BLN, byte]
; 2462        2606  1         preset([DSC$B_CLASS] = DSC$K_CLASS_D); %;
; 2463        2607  1
; 2464        2608  1 OWN_PASSWORD($N_LIST);
```

```
;  2466          2609  1 %sbttl 'UAF$GENERATE -- Generate random passwords'
;  2467          2610  1 routine UAF$GENERATE =
;  2468          2611  2 begin
;  2469          2612  2
;  2470          2613  2 !++
;  2471          2614  2 !
;  2472          2615  2 ! FUNCTIONAL DESCRIPTION:
;  2473          2616  2 !
;  2474          2617  2 !       Decide which passwords to generate
;  2475          2618  2 !
;  2476          2619  2 ! INPUTS:
;  2477          2620  2 !
;  2478          2621  2 !       none
;  2479          2622  2 !
;  2480          2623  2 ! IMPLICIT INPUTS:
;  2481          2624  2 !
;  2482          2625  2 !       CLI$GET_VALUE(%ascid'GENERATE_PASSWORD')
;  2483          2626  2 !
;  2484          2627  2 ! OUTPUTS:
;  2485          2628  2 !
;  2486          2629  2 !       none
;  2487          2630  2 !
;  2488          2631  2 ! IMPLICIT OUTPUTS:
;  2489          2632  2 !
;  2490          2633  2 !       Places new values in $PASSWORD(n)'s
;  2491          2634  2 !
;  2492          2635  2 ! ROUTINE VALUE:
;  2493          2636  2 !
;  2494          2637  2 !       none
;  2495          2638  2 !
;  2496          2639  2 ! SIDE EFFECTS:
;  2497          2640  2 !
;  2498          2641  2 !       none
;  2499          2642  2 !
;  2500          2643  2 !--
;  2501          2644  2
;  2502          2645  2     own WHICH: block[DSC$K_D_BLN, byte] preset([DSC$B_CLASS] = DSC$K_CLASS_D);
;  2503          2646  2
;  2504        M 2647  2     macro ACTIVE_PASSWORD(n) =
;  2505          2648  2         ch$neg(%name('UAF$S_PWD',n), RECBUF[%name('UAF$Q_PWD',n)], 0, 0, 0) %;
;  2506        M 2649  2     macro WHICH_EQL[THIS] =
;  2507        M 2650  2         ch$eql( .WHICH[DSC$W_LENGTH], .WHICH[DSC$A_POINTER],
;  2508          2651  2                 %charcount(THIS), uplit(%ascii %string(THIS)) ) %;
;  2509        M 2652  2     macro GENERATE_NEW_PASSWORD[THIS, n] =
;  2510        M 2653  2         if (ACTIVE_PASSWORD(n) and (FALSE or WHICH_EQL('CURRENT'));
;  2511        M 2654  2         or WHICH_EQL(THIS, 'BOTH', 'ALL') then
;  2512        M 2655  2             AUTHORIZE_GENERATE($PASSWORD(n), uplit(%ascii THIS), % charcount(THIS))
;  2513        M 2656  2         else
;  2514          2657  2             $PASSWORD(n)[DSC$W_LENGTH] = 0; %;
;  2515          2658  2
;  2516          2659  2     CLI$GET_VALUE(%ascid'GENERATE_PASSWORD', WHICH);
;  2517          2660  2     GENERATE_NEW_PASSWORD($NARY_LIST);
;  2518          2661  2
;  2519          2662  2     return TRUE;
;  2520          2663  2
;  2521          2664  1 end;
```

```
                                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

4F 57 53 53 41 50 5F 45 54 41 52 45 4E 45 47  004E0 P.AFI:  .ASCII  \GENERATE_PASSWORD\<0><0><0>                    ;
                                              00 00 00 44 52  004EF                                                 ;
                                                 010E0011     004F4 P.AFH:  .LONG   17694737                        ;
                                                 00000000'    004F8         .ADDRESS P.AFI                          ;
                                    00 54 4E 45 52 52 55 43   004FC P.AFJ:  .ASCII  \CURRENT\<0>                   ;
                                    00 59 52 41 4D 49 52 50   00504 P.AFK:  .ASCII  \PRIMARY\<0>                   ;
                                                 48 54 4F 42  00508 P.AFL:  .ASCII  \BOTH\                        ;
                                                 00 4C 4C 41  00510 P.AFM:  .ASCII  \ALL\<0>                      ;
                                    00 59 52 41 4D 49 52 50   00514 P.AFN:  .ASCII  \PRIMARY\<0>                   ;
                                    00 54 4E 45 52 52 55 43   0051C P.AFO:  .ASCII  \CURRENT\<0>                   ;
                        00 00 00 59 52 41 44 4E 4F 43 45 53   00524 P.AFP:  .ASCII  \SECONDARY\<0><0><0>           ;
                                                 48 54 4F 42  00530 P.AFQ:  .ASCII  \BOTH\                        ;
                                                 00 4C 4C 41  00534 P.AFR:  .ASCII  \ALL\<0>                      ;
                        00 00 00 59 52 41 44 4E 4F 43 45 53   00538 P.AFS:  .ASCII  \SECONDARY\<0><0><0>           ;

                                                                        .PSECT  $OWN$,NOEXE,2

                                                 00#  00114 PASSWORD_:
                                                                        .BYTE   0[3]                             ;
                                                 02   00117             .BYTE   2                                ;
                                                      00118             .BLKB   4
                                                 00#  0011C PASSWORD_2:
                                                                        .BYTE   0[3]                             ;
                                                 02   0011F             .BYTE   2                                ;
                                                      00120             .BLKB   4
                                                 00#  00124 WHICH:      .BYTE   0[3]                             ;
                                                 02   00127             .BYTE   2                                ;
                                                      00128             .BLKB   4

                                                                        .PSECT  $CODE$,NOWRT,2

                                                      007C 00000 UAF$GENERATE:
                                                                        .WORD   Save R2,R3,R4,R5,R6              ; 2610
                                    56 00000000V   00  9E 00002         MOVAB   AUTHORIZE_GENERATE, R6           ;
                                    55 00000000'   00  9E 00009         MOVAB   P.AFH, R5                        ;
                                    54 00000000'   00  9E 00010         MOVAB   WHICH, R4                        ;
                                                   54  DD 00017         PUSHL   R4                               ; 2659
                                                   55  DD 00019         PUSHL   R5                               ;
                          00000000G   00           02  FB 0001B         CALLS   #2, CLI$GET_VALUE                ;
00                  00 00000000G   00           08  2D 00022         CMPC5   #8, RECBUF+340, #0, #0, a#^X00000000 ; 2660
                                00000000       9F      0002B
                                               0D  13 00030         BEQL    1$                                  ;
                                    50   04    A4  D0 00032         MOVL    WHICH+4, R0                         ;
07                  00              60         64  2D 00036         CMPC5   WHICH, (R0), #0, #7, P.AFJ          ;
                                         08    A5      0003B
                                               27  13 0003D         BEQL    2$                                  ;
                                    50   04    A4  D0 0003F 1$:     MOVL    WHICH+4, R0                         ;
07                  00              60         64  2D 00043         CMPC5   WHICH, (R0), #0, #7, P.AFK          ;
                                         10    A5      00048
                                               1A  13 0004A         BEQL    2$                                  ;
                                    50   04    A4  D0 0004C         MOVL    WHICH+4, R0                         ;
```

```
                04          00          60                  64 2D 00050          CMPC5   WHICH, (R0), #0, #4, P.AFL
                                                        18  A5    00055
                                                            0D 13 00057          BEQL    2$
                                                50      04  A4 D0 00059          MOVL    WHICH+4, R0
                03          00              60              64 2D 0005D          CMPC5   WHICH, (R0), #0, #3, P.AFM
                                                        1C  A5    00062
                                                            0D 12 00064          BNEQ    3$
                                                            07 DD 00066 2$:      PUSHL   #7
                                                        20  A5 9F 00068          PUSHAB  P.AFN
                                                        F0  A4 9F 0006B          PUSHAB  PASSWORD
                                                66              03 FB 0006E      CALLS   #3, AUTHORIZE_GENERATE
                                                                03 11 00071      BRB     4$
                                                        F0  A4 B4 00073 3$:      CLRW    PASSWORD
                00          00  00000000G   00              08 2D 00076 4$:      CMPC5   #8, RECBUF+348, #0, #0, a#^X00000000
                                                00000000    9F    0007F
                                                            0D 13 00084          BEQL    5$
                                                50      04  A4 D0 00086          MOVL    WHICH+4, R0
                07          00              60              64 2D 0008A          CMPC5   WHICH, (R0), #0, #7, P.AFO
                                                        28  A5    0008F
                                                            27 13 00091          BEQL    6$
                                                50      04  A4 D0 00093 5$:      MOVL    WHICH+4, R0
                09          00              60              64 2D 00097          CMPC5   WHICH, (R0), #0, #9, P.AFP
                                                        30  A5    0009C
                                                            1A 13 0009E          BEQL    6$
                                                50      04  A4 D0 000A0          MOVL    WHICH+4  R0
                04          00              60              64 2D 000A4          CMPC5   WHICH, (R0), #0, #4, P.AFQ
                                                        3C  A5    00049
                                                            0D 13 000AB          BEQL    6$
                                                50      04  A4 D0 000AD          MOVL    WHICH+4, R0
                03          00              60              64 2D 000B1          CMPC5   WHICH, (R0), #0, #3, P.AFR
                                                        40  A5    000B6
                                                            0D 12 000B8          BNEQ    7$
                                                            09 DD 000BA 6$:      PUSHL   #9
                                                        44  A5 9F 000BC          PUSHAB  P.AFS
                                                        F8  A4 9F 000BF          PUSHAB  PASSWORD_2
                                                66              03 FB 000C2      CALLS   #3, AUTHORIZE_GENERATE
                                                                03 11 000C5      BRB     8$
                                                        F8  A4 B4 000C7 7$:      CLRW    PASSWORD_2
                                                50              01 D0 000CA 8$:  MOVL    #1, R0                          ; 2662
                                                                04 000CD         RET                                     ; 2664

; Routine Size:  206 bytes,    Routine Base:  $CODE$ + 12E5
```

```
;  2523          2665  1 %sbttl 'GETPASSWORD - get user password'
;  2524          2666  1 routine GETPASSWORD =
;  2525          2667  2 begin
;  2526          2668  2
;  2527          2669  2 !++
;  2528          2670  2 !
;  2529          2671  2 ! FUNCTIONAL DESCRIPTION:
;  2530          2672  2 !
;  2531          2673  2 !       Action routine to process password.
;  2532          2674  2 !
;  2533          2675  2 ! INPUTS:
;  2534          2676  2 !
;  2535          2677  2 !       none
;  2536          2678  2 !
;  2537          2679  2 ! IMPLICIT INPUTS:
;  2538          2680  2 !
;  2539          2681  2 !       none
;  2540          2682  2 !
;  2541          2683  2 ! OUTPUTS:
;  2542          2684  2 !
;  2543          2685  2 !       none
;  2544          2686  2 !
;  2545          2687  2 ! IMPLICIT OUTPUTS:
;  2546          2688  2 !
;  2547          2689  2 !       sets hashed password field in recbuf
;  2548          2690  2 !
;  2549          2691  2 ! ROUTINE VALUE:
;  2550          2692  2 !
;  2551          2693  2 !       true -> success
;  2552          2694  2 !       false -> failure
;  2553          2695  2 !
;  2554          2696  2 ! SIDE EFFECTS:
;  2555          2697  2 !
;  2556          2698  2 !       none
;  2557          2699  2 !--
;  2558          2700  2
;  2559          2701  2     own
;  2560          2702  2
;  2561          2703  2         USER_DSC: $bblock[8],
;  2562          2704  2         REC_ENCRYPT_DSC: $bblock[8];
;  2563          2705  2
;  2564        M 2706  2     macro LOCAL_PRESENT[n] =
;  2565          2707  2         local $PRESENT(n): initial(TRUE); %;
;  2566          2708  2
;  2567          2709  2     LOCAL_PRESENT($N_LIST);
;  2568          2710  2
;  2569          2711  2 !
;  2570          2712  2 !       What are the passwords ???
;  2571          2713  2 !
;  2572          2714  2
;  2573          2715  2     if CLI$PRESENT(SD_PASSWORD) eql CLI$_NEGATED then
;  2574          2716  3         begin
;  2575          2717  3             macro BLANK_PASSWORD[n] = $PRESENT(n) = FALSE; %;
;  2576          2718  3             BLANK_PASSWORD($N_LIST);
;  2577          2719  3         end
;  2578          2720  2     else if CLI$PRESENT(SD_PASSWORD) then
;  2579          2721  3         begin
```

```
; 2580        M 2722  3          macro GET_PASSWORD[n] =
; 2581        M 2723  3             $PASSWORD(n)[DSC$W_LENGTH] = 0;
; 2582        M 2724  3             $PASSWORD(n)[DSC$A_POINTER] - 0;
; 2583          2725  3             $PRESENT(n) = CLI$GET_VALUE(SD_PASSWORD, $PASSWORD(n)); %;
; 2584          2726  3          GET_PASSWORD($N_LIST);
; 2585          2727  2       end;
; 2586          2728  2
; 2587          2729  2 !
; 2588          2730  2 !       When is it being changed ???
; 2589          2731  2 !
; 2590          2732  2
; 2591          2733  2    $GETTIM(timadr=TIME_BUF);
; 2592          2734  2
; 2593          2735  2 !
; 2594          2736  2 !       If this is a new password ...
; 2595          2737  2 !
; 2596          2738  2
; 2597          2739  2    if .RECBUF[UAF$W_SALT] eql 0 then
; 2598          2740  2       RECBUF[UAF$W_SALT] = .TIME_BUF<3*8,16>;
; 2599          2741  2
; 2600          2742  2 !
; 2601          2743  2 !       Find the length of the username
; 2602          2744  2 !
; 2603          2745  2
; 2604          2746  3    begin
; 2605          2747  3
; 2606          2748  3       builtin LOCC;
; 2607          2749  3       builtin R0;
; 2608          2750  3       LOCC(%ref(%char(BLANK)), %ref(UAF$S_USERNAME), RECBUF[UAF$T_USERNAME]);
; 2609          2751  3       USER_DSC[DSC$W_LENGTH] = UAF$S_USERNAME - .R0;
; 2610          2752  3       USER_DSC[DSC$A_POINTER] = RECBUF[UAF$T_USERNAME];
; 2611          2753  3
; 2612          2754  2    end;
; 2613          2755  2
; 2614          2756  2 !
; 2615          2757  2 !       Check the new passwords for syntax
; 2616          2758  2 !
; 2617          2759  2
; 2618          2760  3    begin
; 2619          2761  3
; 2620        M 2762  3       macro CHECK_PASSWORD[n] =
; 2621        M 2763  3
; 2622        M 2764  3          if .$PASSWORD(n)[DSC$W_LENGTH] gtru 31 then
; 2623        M 2765  3             return LIB$SIGNAL(UAF$_PWDSYNTAX);
; 2624        M 2766  3          incr COUNTER to .$PASSWORD(n)[DSC$W_LENGTH]-1 do
; 2625        M 2767  3             if ch$fail( ch$find_ch(.SYMBOL_STR<0,8>, SYMBOL_STR+1,
; 2626        M 2768  3                         .vector[.$PASSWORD(n)[DSC$A_POINTER],
; 2627        M 2769  3                         .COUNTER;, byte]) )
; 2628        M 2770  3                then
; 2629          2771  3                   return LIB$SIGNAL(UAF$_PWDSYNTAX); %;
; 2630          2772  3
; 2631          2773  3       CHECK_PASSWORD($N_LIST);
; 2632          2774  3
; 2633          2775  2    end;
; 2634          2776  2
; 2635          2777  2 !
; 2636          2778  2 !       Modify the passwords
```

```
;  2637          2779  2 !
;  2638          2780  2
;  2639          2781  2     begin
;  2640          2782  2
;  2641        M 2783  2        macro MODIFY_PASSWORD[n] =
;  2642        M 2784  2
;  2643        M 2785  2           if not .$PRESENT(n) then
;  2644        M 2786  2              begin ! Blank out the current values
;  2645        M 2787  2
;  2646        M 2788  2                 ch$fill(0, %name('UAF$S_PWD',n), RECBUF[%name('UAF$Q_PWD',n)]);
;  2647        M 2789  2                 ch$fill(0, %name('UAF$S_PWD',n,'_DATE'),
;  2648        M 2790  2                           RECBUF[%name('UAF$Q_PWD',n,'_DATE')]);
;  2649        M 2791  2
;  2650        M 2792  2                 RECBUF[%name('UAF$V_PWD',n,'_EXPIRED')] = FALSE;
;  2651        M 2793  2                 PWD_FLAG = FALSE;
;  2652        M 2794  2
;  2653        M 2795  2              end
;  2654        M 2796  2           else if .$PASSWORD(n)[DSC$W_LENGTH] neq 0 then
;  2655        M 2797  2              begin ! Encrypt the new values
;  2656        M 2798  2
;  2657        M 2799  2                 RECBUF[%name('UAF$B_ENCRYPT',n)] = ENCRYPT;
;  2658        M 2800  2                 REC_ENCRYPT_DSC[DSC$W_LENGTH] = %name('UAF$S_PWD',n);
;  2659        M 2801  2                 REC_ENCRYPT_DSC[DSC$A_POINTER] = RECBUF[%name('UAF$Q_PWD',n)];
;  2660        M 2802  2                 PWD_FLAG = FALSE;
;  2661        M 2803  2
;  2662        M 2804  2                 LGI$HPWD(REC_ENCRYPT_DSC, $PASSWORD(n),
;  2663        M 2805  2                    .RECBUF[%name('UAF$B_ENCRYPT',n)], .RECBUF[UAF$W_SALT],
;  2664        M 2806  2                    user_dsc);
;  2665        M 2807  2
;  2666        M 2808  3                 ch$move( %name('UAF$S_PWD',n,'_DATE'), TIME_BUF,
;  2667        M 2809  3                           RECBUF[%name('UAF$Q_PWD',n,'_DATE')]);
;  2668        M 2810  3
;  2669          2811  3              end; %;
;  2670          2812  3
;  2671          2813  3        MODIFY_PASSWORD($N_LIST);
;  2672          2814  3
;  2673          2815  2     end;
;  2674          2816  2
;  2675          2817  2 !
;  2676          2818  2 !       That's all folks ...
;  2677          2819  2 !
;  2678          2820  2
;  2679          2821  2     return TRUE;
;  2680          2822  2
;  2681          2823  1 end;


                                                .PSECT  $OWN$,NOEXE,2

                                        0012C USER_DSC:
                                                .BLKB   8
                                        00134 REC_ENCRYPT_DSC:
                                                .BLKB   8

                                                .EXTRN  SYS$GETTIM
```

```
                                                          .PSECT  $CODE$,NOWRT,2

                                  0FFC 00000 GETPASSWORD:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11    : 2666
                5B 00000000G  00    9E 00002              MOVAB   PWD_FLAG, R11                            :
                5A 00000000G  00    9E 00009              MOVAB   SYMBOL_STR, R10                          :
                59 00000000G  00    9E 00010              MOVAB   TIME_BUF, R9                             :
                58 00000000G  00    9E 00017              MOVAB   RECBUF+358, R8                           :
                57 00000000'  00    9E 0001E              MOVAB   PASSWORD_, R7                            :
                54            01    D0 00025              MOVL    #1, PRESENT_                             : 2667
                56            01    D0 00028              MOVL    #1, PRESENT_2                            :
                   00000000'  00    9F 0002B              PUSHAB  SD_PASSWORD                              : 2715
       00000000G  00          01    FB 00031              CALLS   #1, CLI$PRESENT                          :
       00000000G  8F          50    D1 00038              CMPL    R0, #CLI$_NEGATED                        :
                              06    12 0003F              BNEQ    1$                                       :
                              54    D4 00041              CLRL    PRESENT_                                 : 2718
                              56    D4 00043              CLRL    PRESENT_2                                :
                              40    11 00045              BRB     2$                                       : 2715
                   00000000'  00    9F 00047 1$:          PUSHAB  SD_PASSWORD                              : 2720
       00000000G  00          01    FB 0004D              CALLS   #1, CLI$PRESENT                          :
                  30          50    E9 00054              BLBC    R0, 2$                                   :
                              67    B4 00057              CLRW    PASSWORD_                                : 2726
                        04    A7    D4 00059              CLRL    PASSWORD_+4                              :
                              57    DD 0005C              PUSHL   R7                                       :
                   00000000'  00    9F 0005E              PUSHAB  SD_PASSWORD                              :
       00000000G  00          02    FB 00064              CALLS   #2, CLI$GET_VALUE                        :
                  54          50    D0 0006B              MOVL    R0, PRESENT_                             :
                        08    A7    B4 0006E              CLRW    PASSWORD_2                               :
                        0C    A7    D4 00071              CLRL    PASSWORD_2+4                             :
                        08    A7    9F 00074              PUSHAB  PASSWORD_2                               :
                   00000000'  00    9F 00077              PUSHAB  SD_PASSWORD                              :
       00000000G  00          02    FB 0007D              CALLS   #2, CLI$GET_VALUE                        :
                  56          50    D0 00084              MOVL    R0, PRESENT_2                            :
                              59    DD 00087 2$:          PUSHL   R9                                       : 2733
       00000000G  00          01    FB 00089              CALLS   #1, SYS$GETTIM                           :
                              68    B5 00090              TSTW    RECBUF+358                               : 2739
                              04    12 00092              BNEQ    3$                                       :
                  68    03    A9    B0 00094              MOVW    TIME_BUF+3, RECBUF+358                   : 2740
FE9E     C8       20          20    3A 00098 3$:          LOCC    #32, #32, RECBUF+4                       : 2750
  18     A7       20          50    A3 0009E              SUBW3   R0, #32, USER_DSC                        : 2751
           1C     A7    FE9E  C8    9E 000A3              MOVAB   RECBUF+4, USER_DSC+4                     : 2752
                  1F          67    B1 000A9              CMPW    PASSWORD_, #31                           : 2773
                              45    1A 000AC              BGTRU   9$                                       :
                  53          67    3C 000AE              MOVZWL  PASSWORD_, R3                            :
                  52          01    CE 000B1              MNEGL   #1, COUNTER                              :
                              15    11 000B4              BRB     6$                                       :
                  51          6A    9A 000B6 4$:          MOVZBL  SYMBOL_STR, R1                           :
                  50    04    A7    D0 000B9              MOVL    PASSWORD_+4, R0                          :
  01     AA       51        6240    3A 000BD              LOCC    @COUNTER[R0], R1, SYMBOL_STR+1           :
                              02    12 000C3              BNEQ    5$                                       :
                              51    D4 000C5              CLRL    R1                                       :
                              51    D5 000C7 5$:          TSTL    R1                                       :
                              28    13 000C9              BEQL    9$                                       :
         E7       52          53    F2 000CB 6$:          AOBLSS  R3, COUNTER, 4$                          :
                  1F    08    A7    B1 000CF              CMPW    PASSWORD_2, #31                          :
                              1E    1A 000D3              BGTRU   9$                                       :
                  53    08    A7    3C 000D5              MOVZWL  PASSWORD_2, R3                           :
```

```
                                52              01 CE 000D9          MNEGL   #1, COUNTER
                                                23 11 000DC          BRB     10$
                                51              6A 9A 000DE 7$:      MOVZBL  SYMBOL_STR, R1
                                50      0C      A7 D0 000E1          MOVL    PASSWORD_2+4, R0
                01      AA      51          6240 3A 000E5          LOCC    (COUNTER)[R0], R1, SYMBOL_STR+1
                                                02 12 000EB          BNEQ    8$
                                                51 D4 000ED          CLRL    R1
                                                51 D5 000EF 8$:      TSTL    R1
                                                0E 12 000F1          BNEQ    10$
                                    00000000G   8F DD 000F3 9$:      PUSHL   #UAF$_PWDSYNTAX
                        00000000G   00          01 FB 000F9          CALLS   #1, LIB$SIGNAL
                                                   04 00100          RET
                        D9              52      53 F2 00101 10$:     AOBLSS  R3, COUNTER, 7$
                                        16      54 E8 00105          BLBS    PRESENT, 11$                        2813
        08              00              6E      00 2C 00108          MOVC5   #0, (SP), #0, #8, RECBUF+340
                                            EE  A8    0010D
        08              00              6E      00 2C 0010F          MOVC5   #0, (SP), #0, #8, RECBUF+380
                                            16  A8    00114
                            6F      A8          02 8A 00116          BICB2   #2, RECBUF+469
                                                6B D4 0011A          CLRL    PWD_FLAG
                                                2F 11 0011C          BRB     12$
                                                67 B5 0011E 11$:     TSTW    PASSWORD_
                                                2B 13 00120          BEQL    12$
                            02      A8      00G 8F 90 00122          MOVB    #ENCRYPT, RECBUF+360
                            20      A7          08 B0 00127          MOVW    #8, REC_ENCRYPT_DSC
                            24      A7      EE  A8 9E 0012B          MOVAB   RECBUF+340, REC_ENCRYPT_DSC+4
                                                6B D4 00130          CLRL    PWD_FLAG
                                            18  A7 9F 00132          PUSHAB  USER_DSC
                                        7E      68 3C 00135          MOVZWL  RECBUF+358, -(SP)
                                        7E      02 A8 9A 00138       MOVZBL  RECBUF+360, -(SP)
                                                57 DD 0013C          PUSHL   R7
                                            20  A7 9F 0013E          PUSHAB  REC_ENCRYPT_DSC
                                00000000G   00      05 FB 00141      CALLS   #5, LGI$HPWD
                16      A8              69          08 28 00148      MOVC3   #8, TIME_BUF, RECBUF+380
                                        16          56 E8 0014D 12$: BLBS    PRESENT_2, 13$
        08              00              6E      00 2C 00150          MOVC5   #0, (SP), #0, #8, RECBUF+348
                                            F6  A8    00155
        08              00              6E      00 2C 00157          MOVC5   #0, (SP), #0, #8, RECBUF+388
                                            1E  A8    0015C
                            6F      A8          04 8A 0015E          BICB2   #4, RECBUF+469
                                                6B D4 00162          CLRL    PWD_FLAG
                                                31 11 00164          BRB     14$
                                            08  A7 B5 00166 13$:     TSTW    PASSWORD_2
                                                2C 13 00169          BEQL    14$
                            03      A8      00G 8F 90 0016B          MOVB    #ENCRYPT, RECBUF+361
                            20      A7          08 B0 00170          MOVW    #8, REC_ENCRYPT_DSC
                            24      A7      F6  A8 9E 00174          MOVAB   RECBUF+348, REC_ENCRYPT_DSC+4
                                                6B D4 00179          CLRL    PWD_FLAG
                                            18  A7 9F 0017B          PUSHAB  USER_DSC
                                        7E      68 3C 0017E          MOVZWL  RECBUF+358, -(SP)
                                        7E      03 A8 9A 00181       MOVZBL  RECBUF+361, -(SP)
                                                08 A7 9F 00185       PUSHAB  PASSWORD_2
                                            20  A7 9F 00188          PUSHAB  REC_ENCRYPT_DSC
                                00000000G   00      05 FB 00188      CALLS   #5, LGI$HPWD
                1E      A8              69          08 28 00192      MOVC3   #8, TIME_BUF, RECBUF+388
                                        50          01 D0 00197 14$: MOVL    #1, R0                              2821
                                                       04 0019A      RET                                         2823
```

; Routine Size: 411 bytes,   Routine Base: $CODE$ + 13B3

```
1 %sbttl 'AUTHORIZE_GENERATE -- Interface to PL/I GENERATE_PASSWORDS routine'
1 routine AUTHORIZE_GENERATE( NEW_PASSWORD: ref block[DSC$K_D_BLN, byte],
1                             THIS, TLENGTH ) =
2 begin
2
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Interface to PL/I procedure GENERATE_PASSWORD, which
2 !       generates random, pronounceable passwords. This routine
2 !       must prompt the user for input with NOECHO in effect
2 !
2 ! INPUTS:
2 !
2 !       None
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       Input from user terminal.
2 !
2 ! OUTPUTS:
2 !
2 !       PASSWORD, by descriptor.
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       User prompt.
2 !
2 ! ROUTINE VALUE:
2 !
2 !       None
2 !
2 ! NOTE:
2 !
2 !       PL/I apparently uses a form of ASCIC to represent its strings,
2 !       however, PL/I uses a word for the character count, not a byte.
2 !
2 !--
2
2   macro
2
2       DSC$L_L1 = 24, 0, 32, 0 %,
2       DSC$L_U1 = 28, 0, 32, 0 %;
2
2   literal
2
2       MAX = 10,
2       MIN = 6,
2       N_WORDS = 5;
2
2   own
2
2       CHANNEL,
2       DSC: block[DSC$K_S_BLN, byte] preset([DSC$B_CLASS]=DSC$K_CLASS_S),
2       FUNCTION: initial(IO$_READPROMPT or IO$M_CVTLOW or IO$M_NOECHO),
2       PLENGTH,
```

```
;  2740          2881  2        PROMPT: vector[45, byte] preset([0]=13, [1]=10,
;  2741          2882  2           [2]='E', [3]='n', [4]='t', [5]='e', [6]='r', [7]=' ');
;  2742          2883  2
;  2743          2884  2    bind
;  2744          2885  2
;  2745          2886  2        P2 = UPLIT(' password: ');
;  2746          2887  2
;  2747          2888  2    !
;  2748          2889  2    !     The password generator requires a complex descriptor
;  2749          2890  2    !
;  2750          2891  2
;  2751        M 2892  2    macro PLI$VARYING_CHARACTER_ARRAY[NAME, LENGTH, WIDTH] =
;  2752        M 2893  2
;  2753        M 2894  2        own
;  2754        M 2895  2
;  2755        M 2896  2            %name(NAME): blockvector[LENGTH, (WIDTH+2), byte],
;  2756        M 2897  2            %name(NAME, '_DSC'): block[32, byte]
;  2757        M 2898  2               preset([DSC$W_MAXSTRLEN] = WIDTH,
;  2758        M 2899  2                      [DSC$B_DTYPE] = 37,
;  2759        M 2900  2                      [DSC$B_CLASS] = DSC$K_CLASS_VSA,
;  2760        M 2901  2                      [DSC$A_POINTER] = %name(NAME),
;  2761        M 2902  2                      [DSC$B_DIMCT] = 1,
;  2762        M 2903  2                      [DSC$A_AC] = %name(NAME) - (WIDTH+2),
;  2763        M 2904  2                      [DSC$L_S1] = (WIDTH+2),
;  2764        M 2905  2                      [DSC$L_L1] = 1,
;  2765          2906  2                      [DSC$L_U1] = LENGTH); %;
;  2766          2907  2
;  2767          2908  2    PLI$VARYING_CHARACTER_ARRAY('PWD', N_WORDS, MAX, 'HYPH', N_WORDS, 2*MAX);
;  2768          2909  2
;  2769          2910  2    !
;  2770          2911  2    !     Build the prompt string
;  2771          2912  2    !
;  2772          2913  2
;  2773          2914  2    ch$move(.TLENGTH, .THIS, PROMPT[8]);
;  2774          2915  2    ch$move(11, P2, PROMPT[8+.TLENGTH]);
;  2775          2916  2
;  2776          2917  2    PLENGTH = 8 + .TLENGTH + 11;
;  2777          2918  2
;  2778          2919  2    !
;  2779          2920  2    !     Open a channel to the user
;  2780          2921  2    !
;  2781          2922  2
;  2782          2923  2    SYS$ASSIGN(%ascid'SYS$COMMAND:', CHANNEL, 0, 0);
;  2783          2924  2    NEW_PASSWORD[DSC$W_LENGTH] = 0;
;  2784          2925  2
;  2785          2926  2    !
;  2786          2927  2    !     Generate random passwords ...
;  2787          2928  2    !
;  2788          2929  2
;  2789          2930  2    while .NEW_PASSWORD[DSC$W_LENGTH] eql 0 do
;  2790          2931  3        begin
;  2791          2932  3
;  2792          2933  3        own
;  2793          2934  3            BUFFER: vector[128, byte],
;  2794          2935  3            IOSB: vector[4, word],
;  2795          2936  3            LISTED;
;  2796          2937  3
```

```
; 2797          2938  3          LISTED = FALSE;
; 2798          2939  3
; 2799          2940  3          SYS$QIOW(0, .CHANNEL, IO$_WRITEVBLK, 0,0,0, 0, 0, 0, 32, 0,0);
; 2800          2941  3
; 2801          2942  3          GENERATE_PASSWORDS(PWD_DSC, HYPH_DSC,
; 2802          2943  3             uplit(MIN), uplit(MAX), uplit(N_WORDS));
; 2803          2944  3
; 2804          2945  3          SYS$QIOW(0, .CHANNEL, IO$_WRITEVBLK, 0,0,0, 0, 0, 0, 32, 0,0);
; 2805          2946  3
; 2806          2947  3          incr i to N_WORDS-1 do
; 2807          2948  4             begin
; 2808          2949  4                DSC[DSC$W_LENGTH] = .PWD[.i, 0, 0, 16, 0];
; 2809          2950  4                DSC[DSC$A_POINTER] = PWD[.i, 2, 0, 0, 0];
; 2810          2951  4                STR$UPCASE(DSC, DSC);
; 2811          2952  3             end;
; 2812          2953  3
; 2813          2954  3          while not .LISTED do
; 2814          2955  4             begin
; 2815          2956  4
; 2816          2957  4                SYS$QIOW(0, .CHANNEL, .FUNCTION, IOSB, 0, 0,
; 2817          2958  4                   BUFFER, 127, 0, 0, PROMPT, .PLENGTH);
; 2818          2959  4
; 2819          2960  4                if .BUFFER[.IOSB[1]] eql 26 then ! CTRL/Z
; 2820          2961  4                   return LIB$SIGNAL(UAF$_PWDNCH, 2, .TLENGTH, .THIS)
; 2821          2962  4                else if .IOSB[1] eql 0 then
; 2822          2963  4                   exitloop;
; 2823          2964  4
; 2824          2965  4                incr i to N_WORDS-1 do
; 2825          2966  4                   LISTED = .LISTED or
; 2826          2967  4                      ch$eql(.IOSB[1], BUFFER,
; 2827          2968  4                         .PWD[.i, 0, 0, 16, 0], PWD[.i, 2, 0, 0, 0]);
; 2828          2969  4
; 2829          2970  4                if .LISTED then
; 2830          2971  5                   begin
; 2831          2972  5                      NEW_PASSWORD[DSC$W_LENGTH] = .IOSB[1];
; 2832          2973  5                      LIB$GET_VM(%ref(.IOSB[1]), NEW_PASSWORD[DSC$A_POINTER]);
; 2833          2974  5                      ch$move(.IOSB[1], BUFFER, .NEW_PASSWORD[DSC$A_POINTER]);
; 2834          2975  5                   end
; 2835          2976  4                else
; 2836          2977  4                   LIB$SIGNAL(UAF$_PWDNOL);
; 2837          2978  4
; 2838          2979  3             end;
; 2839          2980  3
; 2840          2981  2          end;
; 2841          2982  2
; 2842          2983  2 !
; 2843          2984  2 !        That's all folks ...
; 2844          2985  2 !
; 2845          2986  2
; 2846          2987  2      return SYS$DASSGN(.CHANNEL);
; 2847          2988  2
; 2848          2989  1 end;


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2
```

```
                                                00544 P.AFT:  .ASCII  \ password: \<0>
                                                00550 P.AFV:  .ASCII  \SYS$COMMAND:\
                                      010E000C  0055C P.AFU:  .LONG   17694732
                                      00000000' 00560         .ADDRESS P.AFV
                                      00000006  00564 P.AFW:  .LONG   6
                                      0000000A  00568 P.AFX:  .LONG   10
                                      00000005  0056C P.AFY:  .LONG   5

                                                              .PSECT  $OWN$,NOEXE,2

                                                0013C CHANNEL:.BLKB   4
                                           00#  00140 DSC:    .BYTE   0[3]
                                           01   00143         .BYTE   1
                                                00144         .BLKB   4
                                      00000177  00148 FUNCTION:
                                                              .LONG   375
                                                0014C PLENGTH:.BLKB   4
                          20 72 65 74 6E 45 0A 0D 00150 PROMPT: .BYTE   13, 10, 69, 110, 116, 101, 114, 32
                                                00158         .BLKB   37
                                                0017D         .BLKB   3
                                                00180 PWD:    .BLKB   60
                                         000A   001BC PWD_DSC:.WORD   10
                                       0C 25    001BE         .BYTE   37, 12
                                      00000000' 001C0         .ADDRESS PWD
                                           00#  001C4         .BYTE   0[3]
                                           01   001C7         .BYTE   1
                                           00#  001C8         .BYTE   0[4]
                                      00000000' 001CC         .ADDRESS PWD-12
                   00000005  00000001 0000000C  001D0         .LONG   12, 1, 5
                                                001DC HYPH:    .BLKB   110
                                                0024A         .BLKB   2
                                           0014 0024C HYPH_DSC:
                                                              .WORD   20
                                       0C 25    0024E         .BYTE   37, 12
                                      00000000' 00250         .ADDRESS HYPH
                                           00#  00254         .BYTE   0[3]
                                           01   00257         .BYTE   1
                                           00#  00258         .BYTE   0[4]
                                      00000000' 0025C         .ADDRESS HYPH-22
                   00000005  00000001 00000016  00260         .LONG   22, 1, 5
                                                0026C BUFFER: .BLKB   128
                                                002EC IOSB:   .BLKB   8
                                                002F4 LISTED: .BLKB   4

                                                      P2=               P.AFT

                                                              .PSECT  $CODE$,NOWRT,2

                                           03FC 00000 AUTHORIZE_GENERATE:
                                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9          ; 2825
                   59 00000000G   00   9E 00002         MOVAB   LIB$SIGNAL, R9
                   58 00000000G   00   9E 00009         MOVAB   SYS$QIOW, R8
                   57 00000000'   00   9E 00010         MOVAB   P2, R7
                   56 00000000'   00   9E 00017         MOVAB   CHANNEL, R6
                   5E                 C2 0001E         SUBL2   #4, SP
 1C    A6     08   BC       0C    AC   28 00021         MOVC3   TLENGTH, @THIS, PROMPT+8              ; 2914
```

```
                          50      1C   A6  9E 00028          MOVAB   PROMPT+8, R0                        ; 2915
                       0C BC4     67        0B  28 0002C     MOVC3   #11, P2, @TLENGTH[R0]
                       10   A     0C   AC     13  C1 00032     ADDL3   #19, TLENGTH, PLENGTH             ; 2917
                                          7E  7C 00038     CLRQ    -(SP)                               ; 2923
                                          56  DD 0003A     PUSHL   R6
                                      18  A7  9F 0003C     PUSHAB  P.AFU
                              00000000G   00      04  FB 0003F     CALLS   #4, SYS$ASSIGN
                                          04  BC  B4 00046     CLRW    @NEW_PASSWORD                       ; 2924
                                          04  BC  B5 00049 1$:  TSTW    @NEW_PASSWORD                       ; 2930
                                              03  13 0004C     BEQL    2$
                                            0111  31 0004E     BRW     11$
                                        01B8      C6  D4 00051 2$:  CLRL    LISTED                          ; 2938
                                                  7E  7C 00055     CLRQ    -(SP)                           ; 2940
                                                  20  DD 00057     PUSHL   #32
                                                  7E  7C 00059     CLRQ    -(SP)
                                                  7E  7C 0005B     CLRQ    -(SP)
                                                  7E  7C 0005D     CLRQ    -(SP)
                                                  30  DD 0005F     PUSHL   #48
                                                  66  DD 00061     PUSHL   CHANNEL
                                                  7E  D4 00063     CLRL    -(SP)
                                 68                   0C  FB 00065     CALLS   #12, SYS$QIOW
                                              28  A7  9F 00068     PUSHAB  P.AFY                           ; 2943
                                              24  A7  9F 0006B     PUSHAB  P.AFX
                                              20  A7  9F 0006E     PUSHAB  P.AFW
                                        0110      C6  9F 00071     PUSHAB  HYPH_DSC                        ; 2942
                                        0080      C6  9F 00075     PUSHAB  PWD_DSC
                              00000000G   00          05  FB 00079     CALLS   #5, GENERATE_PASSWORDS
                                                  7E  7C 00080     CLRQ    -(SP)                           ; 2945
                                                  20  DD 00082     PUSHL   #32
                                                  7E  7C 00084     CLRQ    -(SP)
                                                  7E  7C 00086     CLRQ    -(SP)
                                                  7E  7C 00088     CLRQ    -(SP)
                                                  30  DD 0008A     PUSHL   #48
                                                  66  DD 0008C     PUSHL   CHANNEL
                                                  7E  D4 0008E     CLRL    -(SP)
                                 68                   0C  FB 00090     CALLS   #12, SYS$QIOW
                                                  52  D4 00093     CLRL    I                               ; 2947
                          50      52              0C  C5 00095 3$:  MULL3   #12, I, R0                      ; 2949
                                              44  A640  9F 00099     PUSHAB  PWD[R0]
                                        04    A6        9E  B0 0009D     MOVW    @(SP)+, DSC
                                        08    A6    46  A640  9E 000A1     MOVAB   PWD+2[R0], DSC+4            ; 2950
                                              04    A6  9F 000A7     PUSHAB  DSC                             ; 2951
                                              04    A6  9F 000AA     PUSHAB  DSC
                              00000000G   00          02  FB 000AD     CALLS   #2, STR$UPCASE
                          DD      52              04  F3 000B4     AOBLEQ  #4, I, 3$                       ; 2947
                                 8C       01B8     C6  E8 000B8 4$:  BLBS    LISTED, 1$                      ; 2954
                                              10  A6  DD 000BD     PUSHL   PLENGTH                         ; 2958
                                              14  A6  9F 000C0     PUSHAB  PROMPT                          ; 2957
                                                  7E  7C 000C3     CLRQ    -(SP)
                                 7E       7F       8F  9A 000C5     MOVZBL  #127, -(SP)
                                        01B0      C6  9F 000C9     PUSHAB  BUFFER
                                                  7E  7C 000CD     CLRQ    -(SP)
                                        01B0      C6  9F 000CF     PUSHAB  IOSB
                                              0C  A6  DD 000D3     PUSHL   FUNCTION
                                                  66  DD 000D6     PUSHL   CHANNEL
                                                  7E  D4 000D8     CLRL    -(SP)
                                 68                   0C  FB 000DA     CALLS   #12, SYS$QIOW
```

```
                                50      01B2   C6   3C 000DD          MOVZWL  IOSB+2, R0                                  ; 2960
                                1A      0130 C640   91 000E2          CMPB    BUFFER[R0], #26
                                              12   12 000E8          BNEQ    5$                                          ;
                                            08 AC   DD 000EA          PUSHL   THIS                                        ; 2961
                                            0C AC   DD 000ED          PUSHL   TLENGTH                                     ;
                                            02      DD 000F0          PUSHL   #2                                          ;
                                00000000G   8F      DD 000F2          PUSHL   #UAF$_PWDNCH                                ;
                                69          04      FB 000F8          CALLS   #4, LIB$SIGNAL                              ;
                                            04      FB 000FB          RET                                                 ;
                                              50    D5 000FC 5$:      TSTL    R0                                          ; 2962
                                              03    12 000FE          BNEQ    6$                                          ;
                                            FF46    31 00100          BRW     1$                                          ;
                                              54    D4 00103 6$:      CLRL    I                                           ; 2965
                    50                        54    C5 00105 7$:      MULL3   #12, I, R0                                  ; 2968
                                              55    D4 00109          CLRL    R5                                          ;
                                          44 A640   9F 0010B          PUSHAB  PWD[R0]                                     ;
        9E          00      0130   C6      01B2   C6 2D 0010F          CMPC5   IOSB+2, BUFFER, #0, @(SP)+, PWD+2[R0]       ;
                                          46 A640      00118                                                            ;
                                              02   12 0011B          BNEQ    8$                                          ;
                                              55   D6 0011D          INCL    R5                                          ;
                            01B8   C6         55   C8 0011F 8$:      BISL2   R5, LISTED                                  ;
                    DD                        54   F3 00124          AOBLEQ  #4, I, 7$                                   ; 2966
                                              29   E9 00128          BLBC    LISTED, 9$                                  ; 2970
                                50      01B2   C6   3C 0012D          MOVZWL  IOSB+2, R0                                  ; 2972
                              04    BC          50   B0 00132          MOVW    R0, @NEW_PASSWORD                           ;
                                52      04     AC   D0 00136          MOVL    NEW_PASSWORD, R2                            ; 2973
                                        04     A2   9F 0013A          PUSHAB  4(R2)                                       ;
                              04    AE          50   D0 0013D          MOVL    R0, 4(SP)                                   ;
                                        04     AE   9F 00141          PUSHAB  4(SP)                                       ;
                            00000000G   00         02   FB 00144      CALLS   #2, LIB$GET_VM                              ;
                    04  B2      0130   C6      01B2   C6 28 0014B     MOVC3   IOSB+2, BUFFER, @4(R2)                      ; 2974
                                              09   11 00154          BRB     10$                                         ; 2970
                                00000000G     8F   DD 00156 9$:      PUSHL   #UAF$_PWDNOL                                ; 2977
                                69            01   FB 0015C          CALLS   #1, LIB$SIGNAL                              ;
                                            FF56   31 0015F 10$:     BRW     4$                                          ; 2954
                                              66   DD 00162 11$:     PUSHL   CHANNEL                                     ; 2987
                            00000000G   00         01   FB 00164      CALLS   #1, SYS$DASSGN                              ;
                                                   04   FB 0016B      RET                                                 ; 2989

; Routine Size:  364 bytes,    Routine Base:  $CODE$ + 154E

; 2849          2990  1
; 2850          2991  1 end
; 2851          2992  0 eludom



                                                                      .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;      Name                      Bytes                          Attributes
;
;  $PLIT$                         1392  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                           760  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;  $TPA_KEY0                        4  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  $TPA_STATE                     158  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  $TPA_KEY1                       19  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  $CODE$                        5818  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                          Library Statistics
;
;                                  -------- Symbols --------      Pages       Processing
;       File                       Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA18:[SYSLIB]TPAMAC.L32;1     42        30        71        14        00:00.0
;  _$255$DUA18:[SYSLIB]LIB.L32;1     18619       126         0      1000        00:01.7




;                                  COMMAND QUALIFIERS

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

; Size:         5818 code + 2333 data bytes
; Run Time:       01:16.3
; Elapsed Time:   03:04.9
; Lines/CPU Min:   2354
; Lexemes/CPU-Min: 38424
; Memory Used:  732 pages
; Compilation Complete
```

0451 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

UAFPARSE
LIS

BOOTDRUVIP
LIS

VMBUVAX.P
LIS

UVI.ROM

VMBUVAXIP
MAP

CONIO
LIS

UVMS

REMOVE
COM