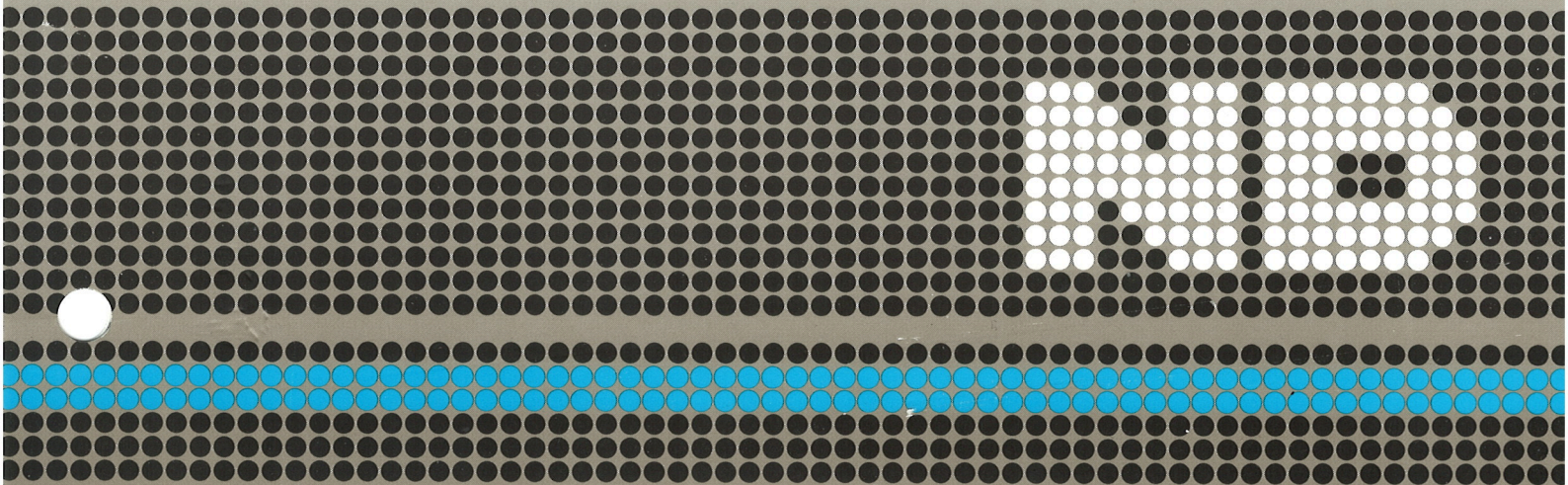


# **User Environment Library Routines**

**ND-60.261.1 EN**





# **User Environment Library Routines**

**ND-60.261.1 EN**

The information in this manual is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this manual. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S. Copyright ©1986 by Norsk Data A.S.

PRINTING RECORD	
PRINTING	NOTES
06/86	Version 1 EN

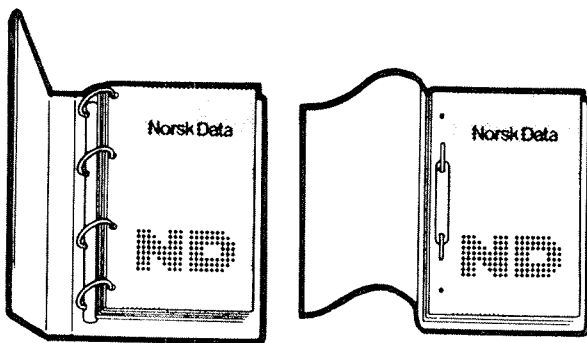
User Environment Library Routines  
 Publ.No. ND-60.261.1 EN

#### UPDATING

Manuals can be updated in two ways, new versions and revisions. New versions consist of a completely new manual which replaces the old one, and incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Customer Support Information and can be ordered from the address below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and give an evaluation of the manual. Both detailed and general comments are welcome.



#### RING BINDER OR PLASTIC COVER

The manual can be placed in a ring binder for greater protection and convenience of use. Ring binders may be ordered at a price of Nkr. 45.- per binder.

The manual may also be placed in a plastic cover. This cover is more suitable for manuals of less than 100 pages than for larger manuals.

Please send your order, as well as all types of inquiries and requests for documentation to the local ND office, or (in Norway) to:

Norsk Data A.S  
 Graphic Center  
 P.O.Box 25 BOGERUD  
 N-0621 OSLO 6 - Norway



I would like to order

..... Ring Binders, 40 mm, at NOK 45.- per binder

..... Plastic Covers, at NOK 10.- per cover

Name: .....

Company: .....

Address: .....

Preface:

## THE PRODUCT

This manual describes the library routines for the product:

USER ENVIRONMENT ND-210518, version C

The manual replaces the previous chapter 10 in the User Environment Reference Manual (ND-60.179.2 EN).

User Environment is run under the operating system SINTRAN III.

## THE READER

The manual is intended for programmers who want to use User Environment library routines.

## PREREQUISITE KNOWLEDGE

User Environment programmers should attend a User Environment operation course, and be familiar with the operating system SINTRAN III.

## THE MANUAL

The manual contains

- an overview of the prerequisites for using the library routines
- a description of the data fields that are used in the user and terminal profiles
- a list of error messages
- detailed descriptions of the available library routines, with examples of use from FORTRAN, COBOL and PLANC.

## CHANGES SINCE THE PREVIOUS VERSION

This manual has in general the same contents as chapter 10 of the User Environment Reference Manual, ND-60.179.2 EN.

The following changes have been made compared to the old chapter 10:

- changes in the storage format of the database in the C version of User Environment have been included
- errors have been corrected
- the routines are presented in a somewhat changed sequence



## RELATED MANUALS

User Environment Reference Manual  
ND-60.194.3  
SINTRAN III Introduction  
ND-60.125  
SINTRAN III Timesharing/Batch Guide  
ND-60.132  
SINTRAN III System Supervisor  
ND-30.003  
SINTRAN III Reference Manual  
ND-60.128

# TABLE OF CONTENTS

Section	Page
1 LIBRARY ROUTINES USER ENVIRONMENT . . . . .	3
2 THE USER PROFILE . . . . .	11
3 THE TERMINAL PROFILE . . . . .	17
4 THE LIBRARY ROUTINES . . . . .	23
UEIACTN /UEACTN . . . . .	23
UEICWAR / UECWAREA . . . . .	25
UEIERRT /UEERRT . . . . .	27
UEIERRO /UEERROR . . . . .	27
UEIGSIP /UEGSIP . . . . .	29
UEIPSIP / UEPSIP . . . . .	32
UEIGTP /UEGTP . . . . .	34
UEIPTP /UEPTP . . . . .	37
UEIGUP /UEGUP . . . . .	41
UEIPUR /UEPUP . . . . .	44
UEIGVER / UEGVERSION . . . . .	48
UEILGIN / UELGIN . . . . .	49
UEILGOU / UELGOU . . . . .	52
UEIPRLK / UEPLRK . . . . .	53
UEIPLCK / UEPLCK . . . . .	53
5 ERROR MESSAGES . . . . .	57

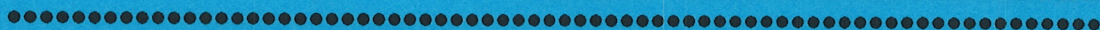






## CHAPTER 1

## LIBRARY ROUTINES IN USER ENVIRONMENT









---

## 1 LIBRARY ROUTINES USER ENVIRONMENT

---

### LIBRARY ROUTINES

Library routines are used to make available to user-written programs information that the User Environment system has about its users and terminals.

### USER-WRITTEN UE PROGRAMS

In addition to using the UE menus, where you can store or change information in the user and terminal profiles, you can write your own application programs to carry out similar functions.

These can for example be programs which store the user or terminal profiles in another format than the standard one, or which set the next task to something else than that defined in the standard menu.

To ease the interaction between applications written for ND's Transaction Processing System (TPS) and User Environment, it is possible to go from TPS to UE without having to go via manual login.

The individual user may, via programs, store information which can be retrieved by an application program. For example, a user may set default values for an application program, which the program may retrieve without having to prompt the user for them.

### Must be logged in to UE

To get access to the information in the UE user and terminal profiles from user written programs, the user MUST be logged in to UE, or the program must execute a call which carries out the login.



Who can change information in user and terminal profiles?

A user with the authorization code PUBLIC can only read/store information from his/her own profile record, and only that information which in the profile description is marked RW (read/write). Information like user and terminal number, password, etc, is protected.

The System Supervisor can read/write information about all users and terminals.

## UE LIBRARY

The library contains standard calls for reading and storage of information in the user and terminal profiles, and certain functions for determining the next task, logging in and out of UE, etc.

In this chapter, the individual calls are described with the parameters they require, what values they may assume, and the values returned after the call has been executed. There are also examples for the languages COBOL, FORTRAN and PLANC.

Some of the calls, those which read or write information in the user or terminal profiles, have a particular layout for the names of the attributes used. The layout of such calls is described in a special section.

## Programming hints

Sometimes it is useful or necessary to let variables of various types have the same address in the memory when you write programs that use UE.

This is the case for the REDEFINES clause in COBOL, the EQUIVALENCE statement in FORTRAN and the symbol = in the declaration part of PLANC programs.

This means that you can store an integer in a 16-bit integer with the same address in the memory as a string of bytes. This is particularly relevant in connection with routines that read from the attributes EXPPERM (which contains allowed users) and EXPALTA (which contains allowed alternative user areas).

## LIBRARY FILES

After the user written program has been compiled, the UE LIBRARY must be read in @NRL or the @BRF-LINKER on the ND-100, or @LINKAGE-LOADER on the ND-500, together with the object file and the standard library for the relevant programming language. On the ND-100, it is important to know whether the program is compiled as one-bank or two-bank.

The following UE LIBRARY FILES may be used:

(U-E)UE-PLIB-1B:BRF for ND-100 one-bank,  
(U-E)UE-PLIB-2B:BRF for ND-100 two-bank,  
(U-E)UE-PLIB:NRF for ND-500.

For PLANC programs, there are two files which may be included in the programs to obtain the necessary IMPORT declarations and error codes:

(U-E)UE-PLIB-B00:IMPT  
(U-E)UE-PLIB-B00:DEFS

For the UE-PLIB calls, the maximum length (number of bytes) that can be transferred in one call is approximately 950. In the C version of UE, this limitation may be bypassed by storing elements on one or more pages in UE. This means that you can have as many alternative user areas as you like (in the extended attribute EXPALTA) in the user profile, or as many allowed users as you like (in the extended attribute EXPPERM) in the terminal profile. You call the next page if the desired choice is not among those you have available already.



## Functions:

COBOL FORTRAN / PLANC	Function	Page
UEIACTN / UEACTION	Set/cancel next task name . . . . .	23
UEICWAR / UECWAREA	Change user area . . . . .	25
UEIERRO / UEERROR	Convert all error codes to text . . .	27
UEIERRT / UEERRT	Convert error code to text for UE . .	27
UEIGSIP / UEGSIP	Read special information for user . .	29
UEIGTP / UEGTP	Read information for terminal . . . .	34
UEIGUP / UEGUP	Read information for user . . . . .	41
UEIGVER / UEGVERSION	Get UE version identification . . . .	48
	changes . . . . .	53
UEIPSIP / UEPSIP	Store special information for user. .	32
UEIPTP / UEPTP	Store information for terminal . . .	37
UEIPUP / UEPUP	Store information for user . . . . .	44
UEIRLCK / UERLCK	Release a user/terminal profile . . .	53

## Routine names:

You must use the function names in the left hand column (UEIxxx) when calling from COBOL, FORTRAN or BASIC programs, and the names in the right hand column when calling from PLANC programs.

## User/terminal profiles:

The user and terminal profile calls read or store information about one or several data elements (attributes). Each element has a symbolic name.

The programmer may choose which attributes s/he wants to use in the program, and in which order.

Name, data type and length are described in the next section.

The information about these functions, you read from or store on a data area you reserve in the program. The data area must have data types, lengths and sequences as specified for the individual attributes.

Status code: All functions return a status code which says whether the call has been correctly executed, or if errors have occurred. The status code contains 0 (zero) when the call has been correctly executed. All other values are error codes.

Error code/message: Check the status code after each call. In case of an error code, you must read a text message by means of the UEERRT call, and display this to the user. All error texts are listed under the call UEIERRT/UEERRT. See page 57.





## CHAPTER 2

### THE USER PROFILE

---







## 2 THE USER PROFILE

### USER PROFILE

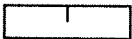




Contains information about a user.





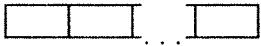

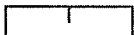
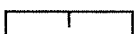


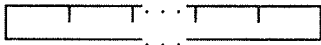
Who can change attributes? The column Read/Write indicates whether the field can be read or changed by users with authorization PUBLIC.

If the column is marked:

R the field may be read  
RW the field may be read and stored  
W the field may only be stored

The System Supervisor can read and store all fields.

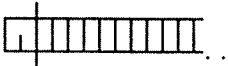
Symbolic names:	Data type and length:	Read/Write:	Use:
USERNO	 2 x integer (integer 4)	R	User number, as specified in UE. Cannot be changed.
USERNAME	 30 characters	R	User name, may be abbreviated. Can only be changed by the supervisor.
PASSWORD	 30 characters	W	Password for the UE profile.  The field is protected against read. Returns blanks only.
AUTHORIZ	 integer	R	Authorization code: 0=PUBLIC, 1=SYSTEM SUPERVISOR.
SIN-COMM	 integer	R	SINTRAN command allowed 0=no, 1=yes.

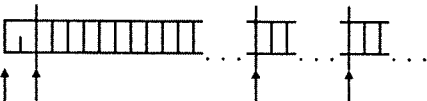
MENU-SYS	 50 characters	R	Name of the menu system file. Language code and file type are added by UE, eg., "-EN:MENU" or "-NO:MENU".
LOGINTAS	 30 characters	RW	Standard task name.
LANGUAGE	 integer	RW	Language code: 0=English, 1=Norw., 2=Swedish, 3=Danish, 4=French, 5=German, 6=Dutch, 7=Finnish 8=Icelandic 9=Italian 10=Spanish, 11=Portugese.
EXPERTLE	 integer	RW	Expert level: 0=beginner, 1=advanced.
LOGINDAT	 6 x integer	R	Date of last login {yr,month,day,hour,min,sec}
LOGINCOU	 integer	R	Login count (accumulated)
FROMTIME	 2 x integer	R	Earliest time of login/ logout (hour(0:23),minute(0:59)).
TOTIME	 2 x integer	R	Latest time of login/ logout (hour(0:23),minute(0:59)).
WEEKDAYS	 integer	R	Weekdays when login is allowed Bit 0 = Monday, 1 = Tuesday and so on. 6 = Sunday. If bit is set, login is allowed.
like this:	 ↑ Bit 0 = Monday		
MAILINFO	 10 x integer	R	First integer gives the number of unread messages from NOTIS-ID.

MAINAREA ... RW Name of main user area  
50 characters (SINTRAN user).

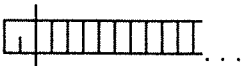
EXPALTA You will find more details about the use of this field on page 41.

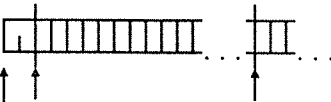
When reading: R

Before  
the call:   
↑  
The page you want to read.

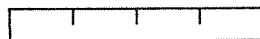
After  
the call:   
↑    ↑                                    ↑    ↑  
50 characters for names of alternative user areas.  
integer = total number of alternative user areas.  
(If you have more than 17 users, you will need more than one page to store the alternative user areas.)

When storing:

Deletion/  
creation:   
↑  
50 characters for names of alternative user areas.  
" D" in the two first bytes means deletion of user area,  
" C" means creation. (D = Delete, C = Create).

Replace  
one with  
another:   
↑    ↑                                    ↑  
50 characters for names of alternative user areas.  
" R" in the two first bytes means that you want to replace  
the first with the second (R = Rename).

USER-ID



5 x integer

H Full user identification.

This field contains a system-oriented user identification, aimed at persons who are specifically responsible for maintaining a UE system.

The structure of the fields:

	field	length	description	value
	a	2 bit	- format version	{00}
	b	1 bit	- not used	
	c	1 bit	- user/session ident.	{1}
	d	4 bit	- control number	{^}
	e	8 bit	- system expansion	{0}
f	f	16 bit	- system CPU no.	{^}
g	g	16 bit	- user number	{^}
h	h	32 bit	- local user number	{^}

{^} varies for each field

Field c: User/Session identification (UID/SID) is set to 1 for UID, or 0 for SID.

Field d: Control number is calculated from whole USER-ID field.

Field e: For future use, expansion of system number

Field f: System CPU number, created at system generation.

Field g: User number.

Field h: Local user number, generated by the UE system for each user profile which is created.

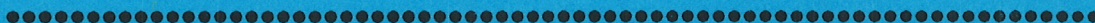
## NOTE!

Profile attributes with data type integer are 16 bits long (2 bytes) both for ND-100 and ND-500, ie.:  
for COBOL PICTURE 9(4) COMP, and  
for FORTRAN INTEGER\*2.  
All text fields must be left justified.



## CHAPTER 3

## THE TERMINAL PROFILE









### 3 THE TERMINAL PROFILE

#### THE TERMINAL PROFILE





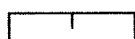


Each terminal connected to the User-Environment system has its own terminal profile.

Who can change attributes? The column Read/Write indicates whether the field can be read or changed by users with authorization PUBLIC.


If the column is marked:

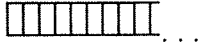
R the field may be read  
RW the field may be read and stored  
W the field may only be stored

The System Supervisor can read and write all fields.

Symbolic names:	Data type and length:	Read/Write:	Use:
TERMINNO	 integer	R	Terminal number (logical device no.) This field is created by UE-PROFILE MANAGER, and cannot be changed.
LOGINTAS	 30 characters	RW	Standard task name for the terminal.
LOGINDAT	 6 x integer	R	Date of last login (yr,mnth,day,hour,min,sec).
LOGINCOU	 integer	R	Login count (accumulated).
FROMTIME	 2 x integer	R	Earliest time of login (hour(0:23),minute(0:59)).
TOTIME	 2 x integer	R	Latest time of login (hour(0:23),minute(0:59)).
WEEKDAYS	 integer	R	Weekdays when login is allowed Bit 0 = Monday, 1 = Tuesday ..... 6 = Sunday. If bit is set, login is allowed.

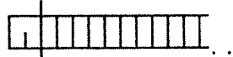


DACCOUNT ... R SINTRAN project password  
28 characters if "DIRECTUS" is filled in.

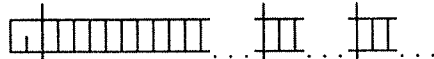
DIRECTUS ... R Direct login with this user  
30 characters name.

EXPPERM Details on the use of this field are found on page 34.

When reading: R

Before  
the call: 

↑  
The page you want to read.

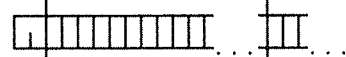
After  
the call: 

↑ 30 characters for name of allowed users.  
integer = total number of allowed users.  
(If you have more than 30 users, you will need more  
than one page to hold the allowed users.)

When storing:

Deletion/  
creation: 

↑ 30 characters for names of allowed users.  
" D" in the two first bytes means deletion of user,  
" C" means creation. (D = Delete, C = Create).

Change  
one with  
another: 

↑ 30 characters for names of allowed users.  
" R" in the two first bytes means that the first user  
is to be replaced by the other (R = Rename).

NOTE!

Profile attributes with data type integer  
are 16 bits long (2 bytes) both for  
ND-100 and ND-500, ie.:  
for COBOL PICTURE 9(4) COMP, and  
for FORTRAN INTEGER\*2.

You set up the symbolic name in the third argument of the function call (in the examples this is called ELMNAME, but you may choose other names which are more suitable), in a table (record or array), where each name occupies 8 characters. In the second argument (called ELMNO in the examples), you specify the number of element names.

no. of names in the  
table

Number of characters in  
the data area:

To protect data areas against overwrite,  
the size of the fields reserved in the  
program must be specified with number of  
characters in the field LENGTH.

NOTE!

The number of character to be received by  
the program must be put into LENGTH before  
the UE call is executed.

After the call has been executed, LENGTH  
will contain the number of transferred  
characters.

Text fields in COBOL:

In COBOL, data elements containing text  
(characters) must be defined as PICTURE  
X(n), where n = no. of characters in the  
field.

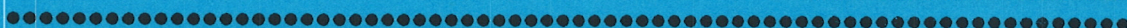
Text fields in FORTRAN:

Text fields in FORTRAN must be declared  
with INTEGER\*2, two characters per  
integer element, both for ND-100 and  
for ND-500.



## CHAPTER 4

## THE LIBRARY ROUTINES









UEIACTN / UEACTION - Set / cancel next task name

---

## 4 THE LIBRARY ROUTINES

---

### UEIACTN / UEACTION

Set/cancel a task name, menu, or command which is activated when the current task is terminated, ie., instead of returning to the ordinary menu.

The task to be started can be the name of another menu or submenu, or (if the first character is @) a program name or SINTRAN command.

---

#### PARAMETERS:

##### function

☐ integer

If 0 (FALSE): DELETES the contents in the next task; the "old" text is transferred to the next parameter: TASKNAME.

IF 1 (TRUE): SETS the contents of parameter TASKNAME as next task.

To distinguish between menus and other tasks, all SINTRAN commands, application program or standard subsystem names must be preceded by the @ character.

##### taskname

...

20 characters

Name of the next menu task, SINTRAN command, application program or subsystem to be started. If the function code is 0 (delete next task), the parameter will be transferred from the "old" text which is already there.

##### length

integer

Number of significant characters in TASKNAME.

##### status

integer

Return status,  
= 0 : OK, function is executed,  
>< 0 : error, not executed.



---

```
COBOL:  WORKING-STORAGE SECTION.
        77  FUNCTION          COMP.
        77  TASKNAME          PICTURE X(20) VALUE IS "@SINTRAN-COMMAND"
        77  LENGTH            COMP.
        77  STATUS            COMP.
        88  OK                VALUE IS 0.

        MOVE 1 TO FUNCTION.
        MOVE 17 TO LENGTH.
        CALL "UEIACTN" USING FUNCTION TASKNAME LENGTH STATUS.
```

---

```
FORTRAN: INTEGER*2 TASKNAME(10)
          INTEGER  FUNCTION,LENGTH,STATUS
          DATA TASKNAME/"@SINTRAN-COMMAND"/,LENGTH/17/

          FUNCTION=1
          CALL UEIACTN(FUNCTION,TASKNAME,LENGTH,STATUS)

or:
          CALL UEIACTN(1,"@SINTRAN-COMMAND",17,STATUS)
```

---

```
PLANC:  IMPORT (routine void,integer(boolean,bytes read write)&
: UEACTION)
integer: status
boolean: function
bytes:   taskname(0:19)

false=: function           % Cancel next task
 '@SINTRAN-COMMAND'=:taskname
UEACTION(function,taskname(0:16))=:status
```

**NOTE!**

TASKNAME can be a menu name or a SINTRAN command if the first character is @. The validity of the task name is checked after the current task has been terminated, not in the call itself. Error messages to the user may therefore come from UE or SINTRAN.

---

## UEICWAR / UECWAREA


Change user area.

This call may be used to change to another user area, defined as main or alternative user area in the profile for the current user in UE.

---

### PARAMETERS:


#### **areaname**

...

50 characters

The name of the new user area. This user area must be defined in the profile as main or alternative user area.

#### **length**

 integer

No. of characters in AREANAME.

NOTE! For COBOL and FORTRAN:

Number of characters reserved for AREANAME must be put in to LENGTH, before the call is executed.


#### **project**

...

30 characters

Project password for SINTRAN III if the accounting system is used.

#### **plength**


 integer

Number of characters in PROJECT.

NOTE! For COBOL and FORTRAN:

Number of characters reserved for PROJECT must be put into PLENGTH, before the call is executed.

#### **status**

 integer

Return status

= 0 : OK, function is executed,  
>< 0 : error, not executed.

---

```
COBOL:  WORKING-STORAGE SECTION.  
        77  STATUS      COMP.  
        77  AREANAME    PIC X(50).  
        77  LENGTH      COMP.  
        77  PROJECT     PIC X(30).  
        77  PLENGTH     COMP.  
        MOVE "SINTRAN-USERNAME" TO AREANAME.  
        MOVE 16 TO LENGDE.  
        MOVE "PROJECT-PASSWORD" TO PROJECT.  
        MOVE 16 TO PLENGTH.  
        CALL "UEICWAR" USING AREANAME LENGTH PROJECT PLENGTH STATUS.
```

---

```
FORTRAN: INTEGER STATUS, LENGTH, PLENGTH  
          INTEGER*2 AREANAME(25), PROJECT(15)  
          DATA AREANAME /"SINTRAN-NAME          "/  
          DATA PROJECT /"PROJECT-PASSWORD      "/  
  
          LENGTH=12  
          PLENGTH=16  
          CALL UEICWAR(AREANAME,LENGTH,PROJECT,PLENGTH,STATUS)
```

---

```
PLANC:  IMPORT (routine void,integer (bytes read, bytes read) &  
             : UECWAREA)  
        integer: status  
        bytes: areaname(0:49):='sintran-name';  
        bytes: project(0:29):='project-password';  
        UECWAREA(areaname,project) =: status
```



---

**UEIERRT / UEERRT**  
**UEIERRO / UEERROR**

Convert UE error status code to text.

Error codes in UE consist of a Standard System Indicator (SSI) and an error code.

These routines convert the error code to the text which is displayed for the user.

You yourself have to make sure that the message is written out. The routine only converts the text to the language set in the user's user profile.

The routine calls UEIERRT and UEERRT convert the error code to text, as shown on page 57.

Error codes from other system routines will here be "translated" to a UE message.

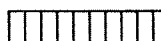
The calls UEIERRO / UEERROR can be used when you want to retrieve text for error codes which come from underlying routines, for example SINTRAN III, the file system, or XMSG.

---

**PARAMETERS:****error code**

integer

Error codes returned from the previous UE call where the error was discovered.

**uemessage**

80 characters

Text for the given error code. If there are several error message files in the system, the language code in the user profile will be used to find the correct language.

**length**

heltall

Length in number of characters reserved to receive the error message text.

After the call has been executed, the parameter contains the number of characters transferred.

<p><b>NOTE!</b> For COBOL and FORTRAN Number of characters in the receiving field must be specified before the call is executed.</p>
--

---

```
COBOL:  WORKING-STORAGE SECTION.  
        77  STATUS      COMP.  
          88  OK          VALUE 0.  
        77  LENGTH      COMP.  
        01  UEMESSAGE  PICTURE X(80).  
  
        CALL "UEIxxxx" USING ....  
        IF NOT OK THEN  
            MOVE 80 TO LENGTH.  
            CALL "UEIERRT" USING STATUS UEMESSAGE LENGTH.  
            DISPLAY ( 24 , 1 ) UEMESSAGE, WITH INVERSE-VIDEO,  
                                AUTO-ERASE.  
            DISPLAY ( 25 , 1 ) BEEP.  
            STOP "Type CR/ENTER to continue".
```

---

```
FORTRAN: INTEGER STATUS,LENGTH  
          INTEGER*2 UEMELDING(40)  
  
          CALL UEIxxxx (....  
          IF (STATUS .NE. 0) THEN  
              LENGTH=80  
              CALL UEIERRT(STATUS,UEMESSAGE,LENGTH)  
              WRITE (1) UEMESSAGE  
              PAUSE "Type CR/ENTER to continue"
```

---

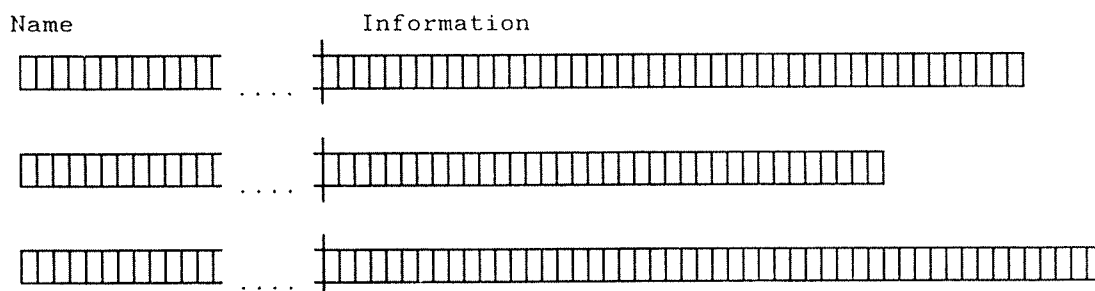
```
PLANC:  IMPORT (routine void,void(integer,bytes write,&  
              integer read write)&  
              : UEERRT)  
integer: status, length  
bytes   : uemessage(0:79)  
  
80 =: length  
if (UExxxx[.....] =: status) >< 0 then  
    UEERRT(status,uemessage,length)  
    output(1,'A',uemessage(0:length-1)  
    output(1,'A','Type CR/ENTER to continue')
```

**UEIGSIP / UEGSIP**

Get special information stored in connection with the user profile of a user.

These two calls are intended for application programs that want to store information in the user profile, which later can be read, using a name of your own choice. Formats, datatypes, etc. depend entirely on the application program. However, the maximum length is set to 950 characters. UEIGSIP/UEIPSIP reads/writes only one record at the time and transfers the contents between the user profile and the calling program. An error message will occur if you try to operate with too much data.

The library routines in UE keep track of the information you write/read by means of the names you give your information strings, as shown in this illustration:



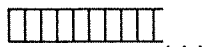
26 char. name parts, max. ca. 900 char. can be read/written per call

The System Supervisor can retrieve information about other users.

Normally, information can only be retrieved for the user name which executes the call. However, the System Supervisor can also retrieve information for any user by specifying the user name in parentheses (USER) in front of "subname".



## PARAMETERS:

**subname**


26 characters

The name given to the data area SUBINFO connected to the user profile, max. 26 characters. This name is later used to retrieve the information to the application program.

If SUBINFO is to be set for another user, the user name must be entered in front of the subname, in parentheses.

NOTE! For COBOL and FORTRAN:


If the field "subname" has less than 26 characters, the field must be terminated by an apostrophe (').

**subinfo**

Variable length,  
max. 900 characters


Data area where the special information is to be returned.

The organization of this field is handled by the application program.

**subleng** integer

No. of characters in SUBINFO, max. 950 characters.

Is used to protect the field against overwrite. The return of the call gives this parameter the number of characters transferred.

**status** integer

Return status,

= 0 : OK, function is executed,

>< 0 : error, not executed.

If SUBNAME does not exist, the error code 16003B/7171D is returned.

---

```
COBOL:  WORKING-STORAGE SECTION.
        77  SUBNAME          PIC X(26) VALUE "APP-007' ".
        77  SUBINFO          PIC X(950).
        77  SUBLENG          COMP.
        77  STATUS           COMP.
        MOVE 950 TO SUBLENG.
        CALL "UEIGSIP" USING SUBNAME,SUBINFO,SUBLENG,STATUS.
```

---

```
FORTRAN: INTEGER*2 SUBNAME(4),SUBINFO(50)
          INTEGER SUBLENG,STATUS
          DATA SUBNAME/"APP-007'"/

          SUBLENG = 100
          CALL UEGSIP(SUBNAME,SUBINFO,SUBLENG,STATUS)
```

---

```
PLANC:  IMPORT (routine void,integer(bytes,bytes write,&
            integer write): UEGSIP)
        bytes:  subinfo(0:999)
        integer: subleng

        UEGSIP('APP-007',subinfo,subleng)=:status
```

---

**UEIPSIP / UEPSIP**

Store special information (data) in connection with the user profile.

Any number of information records may be stored under a user profile. Each record must have a unique name under the user where it is stored, but may be repeated for other users.

The data that is stored can be retrieved from an application program with the calls UEIGSIP / UEGSIP, but the use of the field is handled by the program.

The System Supervisor can store information for another user.

Normally, information can only be stored for the user name where the call is executed. However, the System Supervisor can put information into other users' profiles.

---

**PARAMETERS:****subname**...

58 characters

Name of the data area SUBINFO, maximum length is 26 characters. The rest of the field is for extra user names.

If SUBINFO is to be set for another user, the user name must be entered in front of the subname, in parentheses.

If you want to store the same information for all users on the system - which you have to be System Supervisor to do - type (), an empty parenthesis, first in the name of the information string.

**NOTE!** For COBOL and FORTRAN:

If the field "subname" has less than 26 characters, it must be terminated by an apostrophe (').

UEIPSIP / UEPSIP - Store special information

**subinfo**


Variable length,  
max 950 characters

The contents of SUBNAME, ie., the special data connected to the user profile. If SUBNAME already exists, old information will be overwritten.

**subleng**


integer

Length of SUBINFO, no. of characters, max length is 950 characters.

Only the number of characters specified will be transferred.

**status**


integer

Return status

= 0 : OK, function is executed,

&gt;&lt; 0 : error, not executed

---

```

COBOL:  WORKING-STORAGE SECTION.
        77  SUBNAME    PIC X(26) VALUE "SPECIAL' ".
        77  SUBINFO    PIC X(40).
        77  SUBLENG    COMP.
        77  STATUS     COMP.

        MOVE "this info to be stored" TO SUBINFO.
        MOVE 36 TO SUBLENG.
        CALL "UEIPSIP" USING SUBNAME,SUBINFO,SUBLENG,STATUS.

```

---

```

FORTRAN: INTEGER*2 SUBINFO(11)
        INTEGER  SUBLENG,STATUS
        DATA     SUBINFO/"THIS INFO TO BE STORED"/

        SUBLENG = 22
        CALL UEIPSIP("APP-007'",SUBINFO,SUBLENG,STATUS)

```

---

```

PLANC:  IMPORT (routine void,integer(bytes, bytes,integer)&
        : UEPSIP)
        bytes  : subinfo(0:21)
        integer: subleng

        'this info to be stored' =: subinfo
        22 =: subleng
        UEPSIP('APP-007',subinfo,subleng) =: status

```



---

**UEIGTP / UEGTP**

Get terminal profile for a particular terminal.

In the same way as for user profiles, the terminal profile consists of a number of symbolic names.

Names, data types and lengths are described on page 17.

The attribute names must be entered in ELMNAME, the number of names in ELMNO.

The data is returned in a buffer area declared in accordance with the individual attribute.

The attribute EXPPERM can have more than one page of information, since the limit on the number of UE users who can use the same terminal is very high, more than 300. (For an explanation of the page concept, see page 5.)

In this case, the page number is to be stored before UEIGTP/UEGTP is called as an integer of two bytes in the beginning of the byte string that UE returns the answer to.

More practical information about this is given on page 4.

PARAMETERS:

**termno**



integer

Terminal number (logical device number).  
 If 0 (zero), information is fetched from  
 the terminal where the user is logged  
 in.

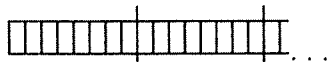
**elmno**



integer

Number of names specified in ELMNAME.

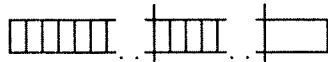
**elmname**



↑1.name    ↑2.name    ↑next name  
 8 char    8 char    8 char, osv

Table of symbolic names to be retrieved.  
 Each name consists of 8 characters, and  
 is entered as a text string.

**uedata**



↑1.field    ↑2.field    ↑etc

The profile elements are put into this  
 data area. You have to adapt data type  
 and length to the description of the ter-  
 minal profile on page 17.

**length**



integer

Number of characters reserved in the data  
 area. Used to control that the area  
 is not exceeded.

NOTE!

The total number of characters in the  
 attributes to be retrieved must be  
 put into LENGTH, before the UE call  
 is executed.

**status**



integer

Return status

= 0 : OK, function is executed,  
 > 0 : error, not executed.

---

```
COBOL:  WORKING-STORAGE SECTION.
        77  TERMNO      COMP.
        77  ELMNO       COMP.
        77  LENGTH      COMP.
        01  ELMNAME.
            02 SYMNAME   PICTURE X(8) OCCURS 8 TIMES.
        01  UEDATA.
            02          DIRECTUS PICTURE X(30)
            02          LOGONTIM PICTURE 9(4) COMP OCCURS 6 TIMES.

        MOVE "DIRECTUS" TO SYMNAME(1).
        MOVE "LOGINDAT" TO SYMNAME(2).
        MOVE 2 TO ELMNO.
        MOVE 42 TO LENGTH.
        CALL "UEIGTP" USING TERMNO ELMNO ELMNAME LENGTH STATUS.
```

---

```
FORTRAN: INTEGER*2 UEDATA(15)
          CHARACTER DIRUSER*30
          INTEGER   LENGTH,STATUS
          EQUIVALENCE (UEDATA,DIRUSER)

          LENGTH=30
          CALL UEIGTP(0,1,"DIRECTUS",UEDATA,LENGTH,STATUS)
```

---

```
PLANC:  IMPORT (routine void,integer(integer,integer,bytes,&
              bytes write,integer write)&
              : UEGTP)
          integer:  length, status
          bytes:    uedata(0:29)

          UEGTP(0,1,'DIRECTUS',uedata,length)=:status
```



---

**UEIPTP / UEPTP**

Store terminal attributes for a terminal.

In the same way as for UEIGTP/UEGTP, you must specify the number of elements and their symbolic names in the sequence the data is stored in the field to be transferred.

If the terminal profile is reserved with the calls UEPLCK or UEIPLCK, it will automatically be released after the call has been executed. To avoid uncontrolled changes, we recommend that a profile record is reserved before it is updated.

---

**PARAMETERS:****termno**

integer

Terminal number (logical device number).

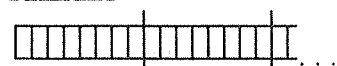
If 0 (zero), information is stored for the terminal where the call is executed.

**elmno**

integer

No. of names specified in ELMNAME.

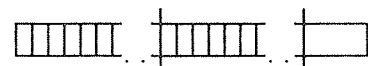
elname



↑1.name    ↑2.name    ↑next name  
8 char    8 char    8 char, etc

Table of symbolic names to be stored.  
Each name has 8 characters, and is  
entered as a text string.

## uedata



↑                    ↑                    ↑                    ↑  
1.field    2.ffield    etc.

The profile elements to be stored are put into this data area before the call is executed. Data types and lengths must fit the description of the terminal profile on page 17.

You can change the list of allowed users of a terminal by changing the EXPPERM attribute. Specify the change in the two first bytes of the field, before you give the user name.

You can:

- Delete an allowed user (use D for Delete)
- Add a new allowed user (use C for Create)
- Replace an allowed user for another (use R for Rename)

The data field for creation of new/removal of old user is to have 32 bytes:



```

↑ ↑
| |
Name of allowed user (30 bytes)

```

First a space, then C to create a new user or D to delete an old one.

The datafield for replacing one allowed user with another is to have 62 bytes:

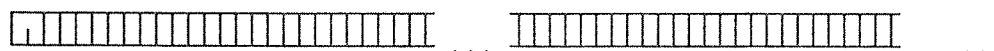


Diagram illustrating memory allocation for two users:

- Old user (30 bytes)
- New user (30 bytes)

First a space, then R for Rename

[illegible]

integer

Number of characters reserved in the data area. Must be the same as the number of characters in the profile attributes to be stored.

NOTE!

The total number of characters in the attributes to be stored must be put into LENGTH, before the call is executed.

Only the specified number of characters  
will be transferred.

integer

### Return status

```
= 0 : OK,    function is executed,  
> 0 : error, not executed.
```



---

```

COBOL:  WORKING-STORAGE SECTION.
        77  TERMNO                COMP.
        77  ELMNO                COMP.
        77  LENGTH              COMP.
        77  STATUS              COMP.
        01  ELMNAME.
            02  SYMNAME            PIC X(8) OCCURS 8 TIMES.
        01  UEDATA.
            02  UETERM            PICTURE 9(4) COMP.
            02  UETASK            PICTURE X(30).

        MOVE "TERMNO  " TO SYMNAME(1).
        MOVE "LOGINTAS" TO SYMNAME(2).
        MOVE ZERO TO TERMNO.
        MOVE 2 TO ELMNO, MOVE 32 TO LENGTH.
        CALL "UEIPTP" USING TERMNO ELMNO ELMNAME UEDATA LENGTH STATUS

```

---

```

FORTRAN: INTEGER*2 OPGNAME(15),ELMNAME(8),UEDATA(16)
        INTEGER  TERMNO,ELMNO,LENGTH,STATUS
        EQUIVALENCE (UEDATA(1),TERMNO),(UEDATA(2),OPGNAME(1))
        DATA      ELMNAME/"TERMNO  LOGINTAS"/,ELMNO/2/

        TERMNO = 0
        LENGTH = 32
        CALL UEIPTP(TERMNO,ELMNO,ELMNAME,LENGTH,STATUS)

```

---

```

PLANC:  IMPORT (routine void,integer(integer,integer,bytes,&
              bytes write,integer write)&
        : UEPTP)
        integer: termno,elmno,length,status
        bytes   : elmname(0:15),uedata(0:59)

        0=:termno;2=:elmno; 'DIRECTUSLOGINTAS'=:elmname
        'USERNAME-FOR-DIRECT      '=:uedata(0:29);
        'MENU-TASK-NAME          '=:uedata(30:59);
        60=:length
        UEPTP(termno,elmno,elmname,uedata,length)=:status

```

---

## UEIGUP / UEGUP

Get profile attribute for a particular user.

The call retrieves a single attribute, or a set of attributes, specified with symbolic names in the third argument of the call: ELMNAME.

The symbolic names for the user profile, with type and length of each data element, are explained on page 11.

The attribute EXPALTA can have more than one page of information, since a user may have as many as 200 alternative user areas. In this case, the page number is to be stored before UEIGUP/UEGUP is called as an integer of two bytes in the beginning of the string which UE returns the answer to. (For an explanation of the page concept, see page 5.)

Practical information about how to do this is found on page 4.

The number of symbolic names in the call must be specified in the second argument: ELMNO.

For those attributes (data elements) which are to be retrieved, space must be reserved in a table (RECORD or ARRAY), with each element in the same sequence as in ELMNAME.

After the call has been executed, each field will contain the information found in the user profile.

---

### PARAMETERS:

**user**

□□□□□□□□...

30 characters

The name of the UE user whose profile attribute you want. If the field has less than 30 characters (inc. blanks), it must be terminated by an apostrophe (').

If a blank field is specified, attributes for the user currently logged into UE will be retrieved.

Each UE user has a number in addition to the name. The number may not be changed and will therefore always identify the UE user.

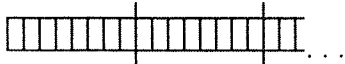
You can read this number: You can start the parameter with two bytes, where all bits are equal to 1, and continue it with four bytes which contain the UE user's number.

A simple way of filling two bytes with one-bits is to set them to the same address as an integer of two bytes, and then set this integer to minus 1, which is represented as a sequence of ones in the computer. See also the hints on page 4.

**elmno**

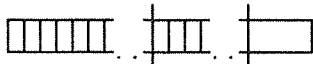
integer

Number of attribute names set in ELMNAME.

**elmname**

↑1.name    ↑2.name    ↑next name  
8 char    8 char    8 char, etc

Table of symbolic names of the attributes to be read. Each name consists of 8 characters, and is entered as a text string.

**uedata**

↑1.field    ↑2.field etc

Data area which receives the attributes retrieved from the user profile. Data type and length must correspond with the sequence of names in ELMNAME, and as described for the profile.

**length**

integer

Number of characters reserved for the UEDATA field to receive the data. Used for checking that no more data is transferred than is reserved in the program.

NOTE! For COBOL and FORTRAN:

The number of characters in the attributes to be read must be specified in LENGTH before the UE call is executed.

**status**

integer

Return status

= 0 : OK, function is executed,  
> 0 : error, not executed.

---

```
COBOL:  WORKING-STORAGE SECTION.
        77  USER      PICTURE X(30) VALUE IS "UE-USER-NAME' ".
        77  ELMNO      COMP VALUE IS 3.
        77  LENGTH     COMP.
        77  STATUS     COMP.
        01  ELMNAME.
            02 SYMNAME  PICTURE  X(8)  OCCURS  3 TIMES.
        01  UEDATA.
            02 LOGDATE.
                03 LTIME  PICTURE 9(4) COMP OCCURS 6 TIMES.
            02 LOGCOUNT PICTURE 9(4) COMP.
            02 LANG      PICTURE 9(4) COMP.

        MOVE "LOGINDAT" TO SYMNAME(1).
        MOVE "LOGINCOU" TO SYMNAME(2).
        MOVE "LANGUAGE" TO SYMNAME(3).
        MOVE 3 TO ELMNO.
        MOVE 16 TO LENGTH.
        CALL "UEIGUP" USING USER ELMNO ELMNAME UEDATA LENGTH STATUS
```

---

```
FORTRAN:  INTEGER*2 USER(15),ELMNAME(12),UEDATA(25)
           INTEGER*2 LOGTIME(2)
           INTEGER  ELMNO,LENGTH,STATUS, LANGCOD,LOGCOUNT

           DATA USER/"UE-USER-NAME' "/,
-           ELMNAME/"LOGINDATLOGINCOULANGUAGE"/,
-           ELMNR/3/
           EQUIVALENCE (UEDATA(1),LOGTIME),(UEDATA(7),LOGCOUNT)
-           (UEDATA(8),LANGCOD)

           LENGTH = 16
           CALL UEIGUP(USER,ELMNO,ELMNAME,UEDATA,LENGTH,STATUS)
```

---

```
PLANC:  IMPORT (routine void,integer(bytes,integer,bytes,&
              bytes write,integer write)&
: UEGUP)
integer: length, status
bytes  : uedata(0:15)

UEGUP('UE-USER-NAME',3,'logindatlogincoulanguage',&
      uedata,length)=:status
```



**UEIPUP / UEPUP**

Store user profile attribute for a specified user.

In the same way as for UEIGUP/UEGUP, the data elements to be retrieved must be specified with number of elements and their symbolic names, and in the same sequence as in the fields where the data is stored.

A description of the symbolic names, and data type and length for the individual elements, is found on page 11.

If a user profile is reserved for updating with the calls UEPLCK or UEIPLCK, the reservation will automatically be cancelled after this call has been executed.

It is not necessary to reserve a profile to store data in it. However, to avoid unintentional overwrite, it is strongly recommended.

**PARAMETERS:****user**
...

30 characters

Name of the UE user whose profile attribute you want to store.

If the field consists of less than 30 characters (inc. blanks), it must be terminated by an apostrophe (') in COBOL and FORTRAN. A blank field sets the values for the UE user currently logged in.

UE users may change their names. To keep a unique identification of UE users, a number has been introduced in addition to the name. This number is always the same and will therefore always identify the user.

You can use this number for programming purposes - for details, see page 41.

**elmno**


integer

Number of attribute names specified in ELMNAME.

# elname

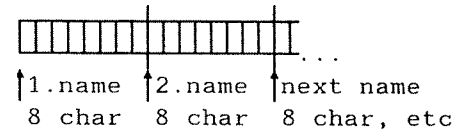
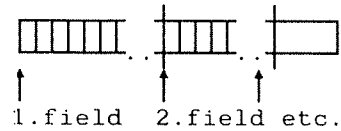


Table of symbolic names of the attributes to be stored. Each name has 8 characters, and is entered as a text string.

# uedata



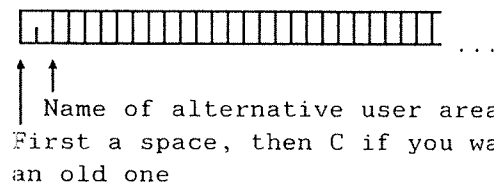
The values to be stored in the profile attributes are put into this data area before the function is called. You have to adapt data type and length to the description of the individual attributes on pages 11 and 17.

If you want to make changes in the alternative user areas of a UE user, you must specify the change in the two first bytes of the field, before you specify the alternative user area(s) you want to change.

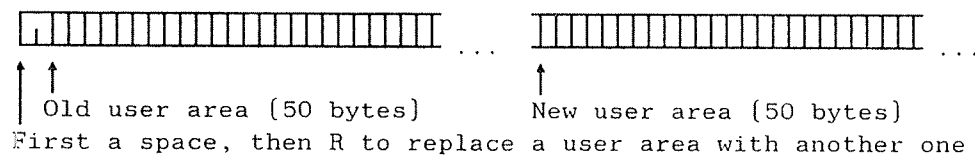
You may:

- Delete an alternative user area (use D for Delete)
- Insert a new alternative user area (use C for Create)
- Replace an alternative user area with another (use R for Rename)

The data field for creation of new/removal of old alternative user area is to have 52 bytes:



The datafield for replacing one alternative user area with another is to have 102 bytes:



An example may clarify this: If you want to delete FLOPPY-USER from the list of alternative user areas, fill in the bytes like this:

D	F	L	O	P	P	Y	-	U	S	E	R										
---	---	---	---	---	---	---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--

**length**
 integer

The length of the data to be transferred, in number of characters. This length must correspond to the profile attributes to be stored, since only the specified number of characters will be transferred.

**NOTE!**

The total number of characters to be changed must be put into the parameter LENGTH, before the UE call is executed.

**status**
 integer
**Return status**

= 0 : OK, function is executed,  
>< 0 : error, not executed

---

```

COBOL:  WORKING-STORAGE SECTION.
        77  USER          PICTURE X(30).
        77  ELMNO         COMP.
        77  LENGTH        COMP.
        77  STATUS        COMP.
        01  ELMNAME.
          02 SYMNAME      PICTURE X(8) OCCURS 3 TIMES.
        01  UEDATA.
          02 FTIME.
            03  FROMHOU    PICTURE 9(4) COMP VALUE IS 8.
            03  FROMMIN    PICTURE 9(4) COMP VALUE IS 30.
          02 TTIME.
            03  TOHOUR     PICTURE 9(4) COMP VALUE IS 16.
            03  TOMIN      PICTURE 9(4) COMP VALUE IS 0.

        MOVE "MY-USER-NAME'" TO USER.
        MOVE "FROMTIME" TO SYMNAME{1}.
        MOVE "TOTIME  " TO SYMNAME{2}.
        MOVE 2 TO ELMNO.
        MOVE 8 TO LENGTH.
        CALL "UEIPUP" USING USER ELMNO ELMNAME UEDATA LENGTH STATUS.
  
```

---

```

FORTRAN: INTEGER LENGTH, STATUS, ELMNO
          INTEGER*2 UEUSER(15),ELMNAVE(8),UEDATA(4)
          DATA UEUSER/"MY-USER-NAME'"/
          DATA ELMNAVN/"FROMTIMETOTIME  "/
          DATA IFRMHOU/8/, IFRMMIN/30/, ITOHOU/16/, ITOMIN/0/
          EQUIVALENCE (UEDATA(1), IFRMTHOU, (UEDATA(2), IFRMMIN),
- (UEDATA(3), ITOHOU), (UEDATA(4), ITOMIN)
          ELMNO=2
          LENGTH=8
          CALL UEIPUP(UEUSER,ELMNO,ELMNAME,UEDATA,LENGTH,STATUS)
  
```

---

```

PLANC:  IMPORT {routine void,integer{bytes,integer,bytes,&
           bytes write, integer}&
          : UEPUP}
          integer: elmno, length, status
          bytes   : elmname(0:15);
          bytes   : uedata(0:7);
          integer: fromhou=uedata(0:1), frommin=uedata(2:3),
                    tohour =uedata(4:5), tomin  =uedata(6:7);
          8=:fromhou;30=:frommin;16=:tohour;0=:tomin;
          'FROMTIME'=:elmname(0:7);'TOTIME  '=:elmname(8:15);
          2=:ELMNO;8=:length;
          UEPUP('MY-USER-NAME',elmno,elmname,uedata,length)=:status
  
```



---

**UEIGVER / UEGVERSION**

Get UE version identification

This call returns an identification which shows what version of UE is in use.

---

**PARAMETERS:****version**

integer

The version identification is returned as an integer containing an ASCII value:  
101B = 'A', 102B = 'B', etc.

**status**

integer

Return status

= 0 : OK, function is executed,  
>< 0 : error, not executed.

---

COBOL: WORKING-STORAGE SECTION.  
77 STATUS COMP.  
77 VERSION PICTURE X(2).  
CALL "UEIGVER" USING VERSION, STATUS.  
DISPLAY "UE-VERSION:", VERSION.

---

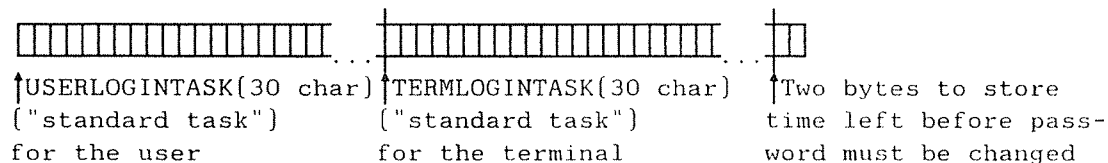
FORTTRAN: INTEGER STATUS  
INTEGER VERSION  
CALL UEIGVER(VERSION, STATUS)  
WRITE(1,'A12,A2') "UE-VERSION: ", VERSION

---

PLANC: IMPORT (routine void,integer (integer write) &  
: UECWAREA)  
integer: status, version  
UEGVERSION(version) =: status  
output(1,'A', 'UE-version:'); output(1,'A',version);

Login to UE from other applications.

After the call has been executed, this data area contains two pieces of information from the UE user profile:

**buffer****length**

integer

No. of characters reserved in BUFFER. Only the specified no. of characters are returned. If the program should not transfer these data, LENGTH is set to 0 (zero).

**status**

integer

Return status

= 0 : OK, function is executed,  
>< 0 : error, not executed.

**time restricted  
password validity**

UE can require the user's password to be changed within certain time periods.

The time it takes before the UE password must be changed is defined during installation of UE. If you program for a computer with time-restricted password validity, consider the following consequences for the UELGIN call:

If two extra bytes in BUFFER have been reserved (making it 62 bytes long), the number of days until the password must be changed will be stored in the two last bytes of this field.

If you get the number -1 there, the user has such a long time before s/he must change the password, that no warnings will be given.

If the validity time for a public user's password has expired, UE will return the error message 16025B, and it will not be possible for the user to log in.

If the user is a system supervisor, the two last bytes in BUFFER will contain the number 0, and the user will be logged in.



UEILGIN / UELGIN - Login to UE from other applications

---

```

COBOL:  WORKING-STORAGE SECTION.
        77  LENGTH                COMP.
        77  STATUS                COMP.
        01  USERPASSWORD.
            02  UENAME             PIC X(30).
            02  UEPSW              PIC X(30).
            02  PROJECT            PIC X(30).
        01  BUFFER.
            02  USERTASK           PIC X(30).
            02  TERMINALTASK       PIC X(30).

        MOVE "MY-NAME"           TO UENAME.
        MOVE "MY-PASSWORD"       TO UEPSW.
        MOVE BLANKS TO PROJECT.
        MOVE 60 TO LENGTH.
        CALL "UEILGIN" USING USERPASSWORD BUFFER LENGTH STATUS.

```

---

```

FORTRAN:                                     INTEGER*2
UENAME(15),UEPSW(15),PROJECT(15),BUFFER(30),UPSW(45)
      INTEGER  LENGTH,STATUS
      EQUIVALENCE (UPSW(1),UENAME(1)),
-                (UPSW(16),UEPSW(1)),
-                (UPSW(31), PROJECT(15))
      DATA UENAME/"MY-USERNAME"              "/,
-      BMPSW /"MY-PASSWORD"                   "/,
-      PROJECT /"ADMINISTRATION-USERENV"      "/"

      LENGTH = 0
      CALL UEILGIN(USERPSW,BUFFER,LENGTH,STATUS)

```

---

```

PLANC:  IMPORT (routine void,integer(bytes,bytes write,integer write)&
      : UELGIN)

      integer: length, status
      bytes: user-password(0:89), buffer(0:59)

      'my-user-name           ' =: user-password(0:29);
      'my-password           ' =: user-password(30:59);
      'administration-userenv ' =: user-password(60:89);
      UELGIN(user-password,buffer,length) =: status

```

---

**UEILGOU / UELGOU**

Log out of UE.

This call terminates user-written programs by logging out of UE.

The user will continue to be logged in to SINTRAN, under the user area which was active when the call was executed.

UE will record the time of logout, but SINTRAN's registration of CPU and terminal use will continue.

---

**PARAMETERS:****status**

Return status



integer

= 0 : OK, function is executed,  
>< 0 : error, not executed.

---

COBOL: WORKING-STORAGE SECTION.  
77 STATUS COMP.

CALL "UEILGOU" USING STATUS.

---

FORTTRAN: INTEGER STATUS

CALL UEILGOU(STATUS)

---

PLANC: IMPORT (routine void,integer &  
: UELGOU)  
integer: status

UELGOU =: status

**UEIPRLK / UEPRLK**  
**UEIPRLK / UEPRLK**

Function call to reserve and release user and terminal profile before storage is allowed. Must be used to avoid unintentional modifications (ie., first READ and then WRITE) on the same profile from different programs at the same time.

If a profile record is reserved by another user when the PUT call (UEPUP, UEPTP) is executed, nothing will be stored, and an error status is returned. If the record is reserved by the same user who executes the PUT call, the record will automatically be released after execution.

The program itself must release a reserved profile record by calling UEIPRLK if a PUT call is not executed.

Because they have similar parameters, the two calls are described together.

**PARAMETERS**

proftype:

☐

integer

1 = the call is for a USER profile  
2 = the call is for a TERMINAL profile

ident:

2 x integer=(INTEGER4)

User number or terminal number (dependent on PROFTYPE).  
If (0), the call applies to the profile currently logged in.

status:

☐

integer

Return status

= 0 : OK, function is executed,  
>< 0 : error, not executed

UEIPRLK / UEPRLK and UEIPRLK / UEPRCK - Reserve and release profile

---

COBOL: WORKING-STORAGE SECTION.  
77 PROFTYPE COMP.  
77 IDENT PIC 9(6) COMP.  
77 STATUS COMP.  
  
MOVE 1 TO PROFTYPE.  
MOVE 0 TO IDENT.  
CALL "UEIPRLK" USING PROFTYPE,IDENT,STATUS.  
or  
CALL "UEIPLCK" USING PROFTYPE,IDENT,STATUS.

---

FORTRAN: INTEGER PROFTYPE, STATUS  
INTEGER\*4 IDENT  
PROFTYPE = 1  
IDENT = 0  
CALL UEIPRLK (PROFTYPE,IDENT,STATUS)  
or  
CALL UEIPLCK (PROFTYPE,IDENT,STATUS)

---

PLANC: IMPORT (routine void,integer(integer,integer4)&  
: UEPRLK)  
or  
IMPORT (routine void,integer(integer,integer4)&  
: UEPLCK)  
integer: status  
  
UEPRLK(1,0) =: status  
or  
UEPLCK(1,0) =: status



## CHAPTER 5

## ERROR MESSAGES









---

## 5 ERROR MESSAGES

---

### ERROR MESSAGES

The list below contains the error messages which are returned for the various error codes.

Error code: Octal/Decimal	Text:
------------------------------	-------

16000B/7168D User Environment System Module

16001B/7169D Profile not available now. It is "locked" by someone else.

16002B/7170D No profile found for this user/terminal number.

16003B/7171D No such subsystem entry for this user.

16004B/7172D Illegal parameter value in profile server request.

16005B/7173D Give more pages to user area USER-ENVIRONMENT (needed by UE).

16006B/7174D Protected function, not allowed for this user.

16010B/7176D Fatal error in profile server. Program error?

16011B/7177D Wrong password.

16012B/7178D User is active (logged in).

16013B/7179D TTY type of terminal is not allowed.

16014B/7180D Wrong UE system software version.

16015B/7181D Wrong SINTRAN III version (must be version I or later).

16016B/7182D No such profile item name.

16017B/7183D This terminal is not available to you (now).

16020B/7184D Too many attempts to log in. This terminal is locked.

16021B/7185D The User Environment system is not running.

16022B/7186D Data communication (XMSG) error in the User Environment system.

16023B/7187D No such user group name.

16024B/7188D Too much data to be transferred in a single request.

16025B/7189D Expired password. Must be changed before the user can log in.

16026B/7190D The new password must be different from the old one.





\*\*\*\*\*

## SEND US YOUR COMMENTS!!!

\*\*\*\*\*



Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.

Please let us know if you

- \* find errors
- \* cannot understand information
- \* cannot find information
- \* find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!



\*\*\*\*\*

## HELP YOURSELF BY HELPING US!!

\*\*\*\*\*

Manual name: User Environment Library Routines

Manual number: ND-60.261.1 EN

What problems do you have? (use extra pages if needed)

---

---

---

---

Do you have suggestions for improving this manual ?

---

---

---

---

---

Your name: \_\_\_\_\_ Date: \_\_\_\_\_

Company: \_\_\_\_\_ Position: \_\_\_\_\_

Address: \_\_\_\_\_

What are you using this manual for ? \_\_\_\_\_

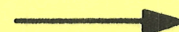
### NOTE!

This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

### Send to:

Norsk Data A.S  
Documentation Department  
P.O. Box 25, Bogerud  
0621 Oslo 6, Norway

Norsk Data's answer will be found on reverse side



Answer from Norsk Data

Date \_\_\_\_\_

Documentation Department  
P.O. Box 25, Bogerud  
0621 Oslo6, Norway



