Entire Connection

Commands

Version 9.3.3

October 2025

ADABAS & NATURAL

This document applies to Entire Connection Version 9.3.3 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: PCC-COMMANDS-933-20251001

Table of Contents

Preface

The following commands are provided with Entire Connection:

BEEP	EXECUTE	PERFORM	RSPMONITOR
CANCEL	EXIT	POPDIR	SCHEDTOP/SCHEDULE
CD	GOTO	PUSHDIR	SET
CHDRIVE	IF/IFNOT	QA	SHIFT
CHMOD	INCR	QUIT	SLEEP
CLOSE	INPUT	RD	TOGGLE
CONNECT	LEARN	READ	TYPE
DECR	LOG	REC_BUFF	WAIT
DISCONNECT	MD	REC_SCR	WAITFOR
DOS	MONITOR	REC_XFER	WAITM
DOSDIR	MSG	RECALL	WAITUNTIL
ELAPSETIME	OPEN-I	RENAME	WRITE
EMULATE	OPEN-O	RESET	
ERASE	OS	RETURN	
EXECTASK	PAUSE	REVEAL	

Conventions

The following information is provided below:

- Command Format
- Syntax Conventions
- Issuing Commands

See also: *Sending Commands to the PC* in the section *Terminal Emulation*.

Command Format

Most commands have one or more operands. The operands are separated by blanks.

```
command operand1 operand2 ... operandn
```

There are mandatory and optional operands. Some operands have optional parameters. Operands can consists of character strings or integer values.

Syntax Conventions

In command descriptions, the following conventions are used:

Convention	Description
UPPERCASE	Items in uppercase are commands, variables etc. You must specify these items as indicated in the syntax.
lowercase italics	Items in lowercase italics are placeholders for information that you must provide. To signal case-sensitivity or embedded blanks, you must enclose the item in double or single quotation marks (" " or ' ').
[ ]	Square brackets indicate that the enclosed item is optional.
{ }	Curly braces indicate that one of the enclosed items is required.
|	Vertical lines are used as separators between alternative parameter values.
...	Ellipsis are used when the item preceding the ellipsis can be repeated.

Issuing Commands

Each command description provides the following information, indicating where a command can be used. There are the following possiblities:

Usage	Description
Procedure File	The command can be used in a procedure file.
Command Line	The command can be issued in the command line.
Key	The command can be assigned to a key or key combination.
API	The command can be used in a program which uses the application programming interface (API).

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 BEEP

Description

Sound the PC alarm.

The system variable `BEEP` must be switched on.

Syntax

```
BEEP
```

Procedure File Example

Copyscr.ncp

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

3 CANCEL

Description

Abort processing of a procedure file.

If this command is used within a nested procedure file, all procedure files are aborted. This command has no effect on scheduled tasks and procedure files.

Syntax

```
CANCEL
```

Procedure File Example

Parms.ncp

Variables Returned

None

Related Commands

EXIT

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

4 CD

Description

Change the current directory on the current drive.

Syntax

```
CD {[drive:]\directory\...}
```

Examples

- Change to the directory *SAG* on the current drive:

```
CD \SAG
```

- Change to the directory which is determined by the contents of the variables #FILEDRIVE and #FILEPATH:

```
CD #FILEDRIVE ':' #FILEPATH
```

- Change to the directory *SAG\PCC* on the current drive:

```
CD \SAG\PCC
```

Variables Returned

#RC (SUCCESS if CD was successful. FAILURE if CD was not successful.)

Related Commands

CHDRIVE, MD, RD

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	No
API:	No

5 CHDRIVE

Description

Change the current drive.

Syntax

```
CHDRIVE drive
```

Examples

- Change to drive A:

```
CHDRIVE A
```

- Change to the drive which is determined by the contents of the variable #FILEDRIVE:

```
CHDRIVE #FILEDRIVE
```

Variables Returned

#RC (SUCCESS if CHDRIVE was successful. FAILURE if CHDRIVE was not successful.)

Related Commands

CD, MD, RD

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	No
API:	No

6 CHMOD

Description

Change the file attributes.

Syntax

```
CHMOD path [options]
```

path is as follows:

{[[*drive*:]*directory*\...\]*filename*[.*extension*]}

options includes the following attributes:

A	archive
H	hidden
R	read-only
S	system

If you do not specify any options, all attributes are turned off.

Examples

- Switch off all attributes:

```
CHMOD Test.ncp
```

- Set the “archive” attribute:

```
CHMOD Test.ncp A
```

- Set the attributes “hidden” and “read-only” for the file defined in the #FILESPEC variable:

```
CHMOD #FILESPEC HR
```

Variables Returned

#RC (SUCCESS if CHMOD was successful. FAILURE if CHMOD was not successful.)

Related Commands

DOSDIR

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

7 CLOSE

Description

Close a file.

If the CLOSE command is omitted, any input file that is open is automatically closed when processing of a procedure file (including all nested procedure files) terminates. If an output file is open, an end-of-file marker is generated.

Syntax

```
CLOSE filenumber
```

filenumber can be a number between 1 and 4.

Example

- Close file 1:

```
CLOSE 1
```

Procedure File Example

Copyscr.ncp

Variables Returned

#RC (SUCCESS if CLOSE was successful. FAILURE if CLOSE was not successful.)

Related Commands

OPEN-I, OPEN-O

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

8 CONNECT

Description

Open a host session.

When the session is opened, the `*COMMTYPE` variable contains the session name.

The session name is displayed in the title bar of the terminal application window.

Syntax

```
CONNECT sessionname
```

sessionname is the name of a session that was defined in the session properties.

Example

- To open the host session named MYIBM:

```
CONNECT MYIBM
```

Variables Returned

None

Related Commands

`DISCONNECT`

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	No
API:	Yes

9 DECR

Description

Subtract 1 from the global and local counter variables or screen position variables.

Syntax

```
DECR {counter|screen-position-variable}...
```

counter can be one of the following variables:

#CNT0 through #CNT9 (local)
+CNT0 through +CNT9 (global)

Valid numbers for the counter variables are between 0 and 32767.

screen-position-variable represents one of the following variables:

#ROW, #COL, #LENGTH (local)
+ROW, +COL, +LENGTH (global)

Examples

- Subtract 1 from the local counter #CNT1:

```
DECR #CNT1
```

- Subtract 1 from the global counter +CNT1:

```
DECR +CNT1
```

- Subtract 1 from the local counters #CNT1 and #CNT2:

```
DECR #CNT1 #CNT2
```

- Subtract 1 from the local counter #CNT1 and from the global counter +CNT1:

```
DECR #CNT1 +CNT1
```

- Subtract 2 from the local counter #CNT1:

```
DECR #CNT1 #CNT1
```

Procedure File Example

Findfile.ncp

Variables Returned

None

Related Commands

INCR, RESET, SET

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

10 DISCONNECT

Description

Close a host session.

When this command is issued from the command line or a procedure file, the current session is closed.

When this command is issued via the API, the session addressed by the API is closed.

Important: This command must not be included in a procedure file that executes an automatic logon to or logoff from the host.

Syntax

```
DISCONNECT
```

Variables Returned

None

Related Commands

CONNECT

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	No
API:	Yes

11 DOS

Description

Execute a DOS command.

This command opens a DOS window under Windows (Command Prompt) in which the commands that have been specified with the command DOS are executed.

The commands DOS and OS are identical.

Syntax

```
DOS doscommand
```

Examples

- Display all files with the extension *exe*:

```
DOS DIR *.exe
```

- Execute the batch file *Test.bat*:

```
DOS Test
```

Variables Returned

None

Related Commands

OS

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	Yes
API:	No

12 DOSDIR

Description

Display DOS directory information.

The contents of the directory are returned in the variables listed below.

The `DOSDIR` command shows the contents of a directory in the same manner as when the `DIR` command is issued in DOS.

Syntax

```
DOSDIR path [options]
```

path is as follows:

{[[*drive*:]*directory*\...\]*filename*[.*extension*]}

For *filename* and *extension*, you can use the wildcard characters asterisk (*) and question mark (?).

If you do not specify *path*, Entire Connection assumes that the `DOSDIR` command was issued with wildcard characters and searches for the next file which matches the pattern that was last used.

If the file is not stored in the current DOS path, you must specify the full path for this file.

options includes the following:

A	files with the “archive” attribute only
D	directories only
H	hidden files only
R	read-only files only
S	system files only

If you do not specify any *options*, only normal files are listed. Directories, read-only, system and hidden files are not shown in this case.

Examples

- List all *ncp* files in the current directory:

```
DOSDIR *.ncp /* list first "ncp" file
DOSDIR       /* list subsequent "ncp" files
```

- List all files containing the “archive” attribute:

```
DOSDIR *.* A /* list first file containing the "archive" attribute
DOSDIR       /* list subsequent files containing the "archive" attribute
```

- List all files which match the contents of the variable #FILESPEC:

```
DOSDIR #FILESPEC /* list first file which matches #FILESPEC
DOSDIR           /* list subsequent files which match #FILESPEC
```

Procedure File Example

Findfile.ncp

Variables Returned

#RC (SUCCESS, FILE NOT FOUND, NO MORE FILES or INVALID PATH)
#FILEDRIVE
#FILEPATH
#FILENAME
#FILEEXT
#FILESIZE
#FILEDATE
#FILEYEAR
#FILEMONTH
#FILEDAY
#FILETIME
#FILEHOUR
#FILEMINUTE

```
#FILESECOND
#FILEMODE
#FILETYPE
#FILESPEC
#FILEINFO
```

See also: *Local Variables*

Related Commands

CD, CHDRIVE, CHMOD, PUSHDIR

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

13 ELAPSETIME

Description

Calculate the difference, in seconds, between two date and time stamps.

Syntax

```
ELAPSETIME date time date time
```

date has the format YYYY/MM/DD. Starting with the year 2000, you must specify the year in the format YYYY. Up to 1999, you can specify the year in the format YY or YYYY.

time has the format HH:MM:SS.

Example

- If the dynamic variables *DATE and *TIME contain the values 1998/03/21 and 12:00:00, the value 900 is returned in the local variable #ELAPSETIME:

```
ELAPSETIME *DATE *TIME 1998/03/21 12:15:00
```

Procedure File Example

Vars.ncp

Variables Returned

#ELAPSETIME - a positive number representing the difference in seconds.

Related Commands

PAUSE, SLEEP, WAIT, WAITFOR, WAITM, WAITUNTIL

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

14 EMULATE

Description

Switch to terminal emulation mode.

This command has two functions:

- If the `EMULATE` command is used in a procedure file, it interrupts processing and switches to terminal emulation mode. This then allows for user input.
- Using the terminal emulation key which has been assigned to the `EMULATE` command, control is returned to the procedure file. Processing will then continue with the next command.

Syntax

```
EMULATE
```

Procedure File Example

Makete.ncp

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	No

15 ERASE

Description

Erase a file.

Syntax

```
ERASE path
```

path is as follows:

{[[*drive*:]*directory*\...\]*filename*[.*extension*]}

Examples

- Erase the file *Test.ncp*:

  ```
  ERASE Test.ncp
  ```

- Erase the file in the *SAG* directory which is determined by the contents of the variables #FILENAME and #FILEEXT:

  ```
  ERASE C:\SAG\ #FILENAME #FILEEXT
  ```

- Erase the file *\SAG\Test.ncp* on the current drive:

```
ERASE \SAG\Test.ncp
```

Variables Returned

#RC (SUCCESS if ERASE was successful. FAILURE if ERASE was not successful.)

Related Commands

DOSDIR, RENAME

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

16 EXECTASK

Description

Execute an Entire Connection task.

Syntax

```
EXECTASK taskname [taskparameter] ...
```

taskname is a task defined to Entire Connection.

taskparameter is a parameter that corresponds to an input parameter that is required by the specified task. You can also specify several task parameters, separated by blanks.

Examples

- Execute the task EDITOR, which invokes a local editor, without parameters:

```
EXECTASK EDITOR
```

- Execute the task EDITOR, which invokes a local editor, with parameters (in this case, the name of the file to be edited):

```
EXECTASK EDITOR Myfile.abc
```

- Execute the task MYTASK using the value contained in the local variable #PARM1:

```
EXECTASK MYTASK #PARM1
```

- Execute the task which is defined in the local variable #PARM1:

```
EXECTASK #PARM1
```

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	Yes
API:	No

17 EXECUTE

Description

Execute a procedure file.

You can nest up to 7 procedure files.

Syntax

```
EXECUTE path [procedurefileparameter] ...
```

path is as follows:

{[[*drive*:]*directory*\...\]*filename*[.*extension*]}

If you do not specify an *extension*, the extension "ncp" is automatically appended to the procedure file name.

If you do not specify a *drive* and/or *directory*, Entire Connection tries to locate the file in the current drive and directory. If the file cannot be found, Entire Connection then tries to locate the file in the procedure directory which has been defined in the user properties.

You can specify up to 9 *procedurefileparameters* (separated by blanks) that correspond the input parameters of the specified procedure file. These are stored in the local variables #PARM1 through #PARM9. The variable #PARM0 contains the full path of the executed procedure file. The variable #PARMNO contains the number of passed parameters passed (00 to 09).

Examples

- Execute the procedure file *Test1.ncp* without parameters:

```
EXECUTE Test1
```

- Execute the procedure *Myproc* and using the parameter value contained in the local variable #PARM1:

```
EXECUTE Myproc #PARM1
```

- Execute the procedure file *Test2.abc* with two parameters:

```
EXECUTE Test2.abc one two
```

- Execute the procedure file *\MYDIR\Test3.ncp* with three parameters:

```
EXECUTE \MYDIR\Test3 one two three
```

- Execute the procedure file which is defined by the local variable #PARM1 without parameters:

```
EXECUTE #PARM1
```

Procedure File Examples

Ncpnest.ncp, *Findfile.ncp*

Variables Returned

#RC (SUCCESS if there was no runtime error. FAILURE if there was a runtime error.)

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	Yes
API:	Yes

18 EXIT

Description

Leave a procedure file and return to the previous procedure file or to Entire Connection.

You can also use this command to pass information back to the parent procedure file.

The `EXIT` command is always executed at the end of a procedure file, even if the last command is not explicitly `EXIT`.

Syntax

```
EXIT [{string|variable}...]
```

Examples

- Leave the procedure file:

  ```
  EXIT
  ```

- Return the current time to the parent procedure file:

  ```
  EXIT 'The current time is ' *TIME
  ```

- Return a message that the file defined in the local variable `#FILESPEC` could not be found:

```
EXIT 'The file ' #FILESPEC ' was not found'
```

- Return a message that the procedure file completed successfully:

```
EXIT 'Success'
```

Procedure File Example

Copyscr.ncp

Variables Returned

`+RC` - contains all values passed with `EXIT` (global variables are always available).

Related Commands

`CANCEL`

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

19 GOTO

Description

Branch to another location in the procedure file.

The tag can be defined before or after the `GOTO` command.

Syntax

```
GOTO tag
```

Example

- Branch to the tag labeled `CONTINUE`:

```
GOTO CONTINUE
```

Procedure File Examples

Findfile.ncp, Copyscr.ncp

Variables Returned

None

Related Commands

`PERFORM`

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

20 IF/IFNOT

Description

`IF` checks a condition. `IFNOT` is short for `IF *SCREEN NE`.

If the condition is true, the statement is executed. If the condition is false, the next statement is executed.

Text enclosed in single or double quotation marks is case-sensitive.

Syntax

```
IF [variable] [operator] variable command
```

```
IFNOT variable command
```

For *variable*, you can use any of the variables provided by Entire Connection (e.g. `#PARM1` or `#CNT1`). When dealing with mainframe applications, the `*SCREEN` variable (syntax see below) is useful to determine which screen is currently being transmitted without having to display it. You can then react as required.

operator is one of the following:

`EQ`	equal
`NE`	not equal
`GE`	greater than or equal to
`LE`	less than or equal to
`GT`	greater than
`LT`	less than

For *command*, you can use any Entire Connection command except the following:

IF
IFNOT
WAITFOR

If you omit the optional operands *variable* and *operator* in an IF statement, the variable *SCREEN (whole screen) and the operator EQ are used.

Syntax for *SCREEN

The variable *SCREEN can only be used once per IF/IFNOT.

```
*SCREEN [row column[length]]
```

row is a value between 1 and the maximum number of lines +1.

column is a value between 1 and the maximum line size.

length is a value between 1 and the screen size.

For example:

*SCREEN means the whole screen.

*SCREEN 2 1 means from row 2, column 1 to the end of the screen.

*SCREEN 2 1 80 means from row 2, column 1, the next 80 positions.

Examples

- Search the whole screen for NEXT. If NEXT is found, branch to the NEXT tag:

```
IF 'NEXT' GOTO NEXT
```

- Search the whole screen for NEXT. If NEXT is not found, branch to the CONTINUE tag:

```
IFNOT 'NEXT' GOTO CONTINUE
```

- Search the next 4 positions for NEXT, starting in row 2 and column 1:

```
IF *SCREEN 2 1 4 EQ 'NEXT' GOTO CONTINUE
```

- Search the next 80 positions for NEXT, starting in row 2 and column 1:

```
IF *SCREEN 2 1 80 EQ 'NEXT' GOTO CONTINUE
```

- Search for the contents of the local variable #PARM1, starting in the row defined by the +ROW variable and the column defined by the +COL variable; the number of positions to be searched is defined by the value of the #CNT3 variable:

```
IF *SCREEN +ROW +COL #CNT3 EQ #PARM1
GOTO CONTINUE
```

- Send the defined keyboard input to the host, if the content of the local variable #PARM1 is not blank:

```
IF #PARM1 NE ' ' TYPE '*NAT' CR
```

Procedure File Examples

Findfile.ncp, Copyscr.ncp

Variables Returned

If the condition for the command IF *SCREEN EQ is true, the screen position of the string is returned in the following local variables:

#ROW (valid values are between 1 and the maximum number of lines +1)

#COL (valid values are between 1 and the maximum line size)

Related Commands

WAITFOR

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	Yes

21 INCR

Description

Add 1 to the global and local variable counters or screen position variables.

Syntax

```
INCR {counter|screen-position-variable}...
```

counter can be one of the following variables:

#CNT0 through #CNT9 (local)
+CNT0 through +CNT9 (global)

Valid numbers for the counter variables are between 0 and 32767.

screen-position-variable represents one of the following variables:

#ROW, #COL, #LENGTH (local)
+ROW, +COL, +LENGTH (global)

Examples

- Add 1 to the global counter +CNT1:

```
INCR +CNT1
```

- Add 1 to the local counters #CNT1 and #CNT2:

```
INCR #CNT1 #CNT2
```

- Add 1 to the local counter `#CNT1` and the global counter `+CNT1`:

```
INCR #CNT1 +CNT1
```

- Add 2 to the local counter `#CNT1`:

```
INCR #CNT1 #CNT1
```

Procedure File Examples

Findfile.ncp, Copyscr.ncp

Variables Returned

None

Related Commands

`DECR`, `RESET`, `SET`

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

22 INPUT

Description

Prompt the user for input.

Syntax

```
INPUT parametervariable length message [message]
```

parametervariable can be one of the following variables:

+PARM1 through +PARM9 (global)
#PARM1 through #PARM9 (local)

For a global *parametervariable*, each character entered at the command prompt is displayed as an asterisk (*). This is important when entering passwords or other sensitive data. If a local *parametervariable* already contains a value, this value is displayed.

length is a value between 1 and 72.

message can be a character string or a variable. One or two lines of message text prompt the user to enter a value. This value is then stored in the *parametervariable*.

Examples

- Prompt for input of an 8 byte long user ID:

```
INPUT #PARM1 8 'Enter your user ID'
```

- Prompt for input of an 8 byte long password which will not be displayed:

```
INPUT +PARM1 8 'Enter your password'
```

- Prompt for input of a value. The length of this value is determined by the the local variable #CNT1. The prompt message is contained in the local variable #PARM2:

```
INPUT #PARM1 #CNT1 #PARM2
```

Procedure File Example

Copyscr.ncp

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

23 LEARN

Description

Create a procedure file in learn mode.

To switch learn mode on and off, you must press the key combination assigned to learn mode.

The default key combination depends on the type of session and the corresponding default key scheme.

Type of Session / Default Key Scheme	Default Key Combination
TN3270 / SAGKEYS1	CTRL+L
BS2000 / BS2000KEYS	ALT+T
VTxxx / VT220PC	ALT+L

After the `LEARN` command has been issued, you are asked for a file name for the procedure file. All keyboard input is then written to the specified procedure file.

During terminal emulation, the character L is shown in column 76 of the status line when learn mode is switched on.

Syntax

```
LEARN
```

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	No

24 LOG

Description

Write a message to the log file.

The name of the log file is *<username>.log*. It is stored in the log/trace directory. If the file does not yet exist, it is created. If the file does exist, the message is appended to it.

Syntax

```
LOG {string|variable}...
```

When the message is written to the log file, a space is not inserted between the operands.

Examples

- Write the current time to the log file:

```
LOG 'The time is ' *TIME
```

- Write a message to the log file, indicating that file name contained in the #FILESPEC variable was not found:

```
LOG 'File ' #FILESPEC ' was not found'
```

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	Yes

25 MD

Description

Make a directory.

Syntax

```
MD {[drive:]\directory\...}
```

Examples

- Make a directory called *NCP* in the current directory:

  ```
  MD NCP
  ```

- Make a directory called *NCP* in the root directory of drive C:

  ```
  MD C:\NCP
  ```

- Make a directory called *NCP* in the *SAG* directory:

  ```
  MD \SAG\NCP
  ```

- Make a directory, using the contents of the variables #FILEDRIVE and #FILEPATH:

```
MD #FILEDRIVE ':' #FILEPATH
```

Variables Returned

#RC (SUCCESS if MD was successful. FAILURE if MD was not successful.)

Related Commands

CD, CHDRIVE, RD

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

26 MONITOR

Description

Write communication-specific data to disk.

This corresponds to the communication trace function on the **Test** property page of the session properties.

Data are written to the file *Mon<nn>.trc* in the log/trace directory, where *nn* is a number starting with 00 and being incremented by one. If the file does not yet exist, the file *Mon00.trc* is created. If the file does exist, the new data are appended to it.

Data are recorded until you issue the `MONITOR` command again. Using a key combination, you can switch this mode on and off.

The default key combination depends on the type of session and the corresponding default key scheme.

Type of Session / Default Key Scheme	Default Key Combination
TN3270 / SAGKEYS1	CTRL+M
BS2000 / BS2000KEYS	CTRL+M
VTxxx / VT220PC	ALT+M

During terminal emulation, the character M is shown in column 5 of the status line when data are recorded.

Important: This command is provided for Entire Connection problem resolution and should only be used with the assistance and direction of your technical support.

Syntax

```
MONITOR
```

Variables Returned

None

Usage

Procedure File:	No
Command Line:	No
Key:	Yes
API:	No

27 MSG

Description

Display a message.

The message is displayed in the terminal application's output window.

Syntax

```
MSG {string|variable}...
```

When the message is displayed, there is no space between the operands.

Examples

- Display a message with the current time:

```
MSG 'The time is ' *TIME
```

- Display a message, indicating that the file name contained in the #FILESPEC variable was not found:

```
MSG 'File ' #FILESPEC ' was not found'
```

Procedure File Example

Findfile.ncp

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

28 OPEN-I

Description

Open a file from which data are read (input file).

The file is available across all nested procedure files. You can, for example, issue the `READ` command from more than one procedure file.

Syntax

```
OPEN-I filenumber path
```

filenumber is a number between 1 and 4.

path is as follows:

{[[*drive*:]*directory*\...\]*filename*[.*extension*]}

Examples

- Open *Test.ncp* as input file 1:

```
OPEN-I 1 Test.ncp
```

- Open the file defined by the variable `#FILESPEC` as input file 2:

```
OPEN-I 2 #FILESPEC
```

Procedure File Example

Copyscr.ncp

Variables Returned

#RC (SUCCESS if OPEN-I was successful. FAILURE if OPEN-I was not successful.)

Related Commands

READ, CLOSE

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

29 OPEN-O

Description

Open a file into which data are written (output file).

The file is available across all nested procedure files. You can, for example, issue the `WRITE` command from more than one procedure file.

If you write to an exisiting file, the new data are appended to this file.

Syntax

```
OPEN-O filenumber path
```

filenumber is a number between 1 and 4.

path is as follows:

{[[*drive*:]*directory*\...\]*filename*[.*extension*]}

Examples

- Open *Test.ncp* as output file 1:

```
OPEN-O 1 Test.ncp
```

- Open the file defined by the variable `#FILESPEC` as output file 2:

```
OPEN-O 2 #FILESPEC
```

Procedure File Example

Copyscr.ncp

Variables Returned

#RC (SUCCESS if OPEN-O was successful. APPEND if the file already exists. FAILURE if OPEN-O was not successful.)

Related Commands

WRITE, CLOSE

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

30 OS

Description

The commands OS and DOS are identical. For compatibility reasons, the OS command is still supported.

See the description of the DOS command.

31 PAUSE

Description

Suspend processing for a specific period of time; at the most for the entire period, or at least until a response from the host is received.

During the pause, it is checked whether a response from the host has been received. If the response is received before the end of the defined pause, processing of the procedure file is continued.

Syntax

```
PAUSE milliseconds
```

milliseconds can be in the range from 0 through 32000 (i.e. 0 thousands of a second through 32 seconds). The value 1000 is equal to 1 second.

Examples

- Suspend processing for 1 second (1000 milliseconds):

```
PAUSE 1000
```

- Suspend processing for the number of seconds that is defined by the variable #CNT1:

```
PAUSE #CNT1
```

Procedure File Example

Vars.ncp

Variables Returned

None

Related Commands

ELAPSETIME, WAIT, WAITFOR, WAITM, WAITUNTIL

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	Yes

32 PERFORM

Description

Branch to another location in the procedure file and execute the statements defined at this location.

The RETURN command continues processing with the statements located directly after the PERFORM command. The tag can be defined before or after the PERFORM command.

PERFORM commands can be nested up to seven levels.

Syntax

```
PERFORM tag
```

Example

- Branch to the ROUTINE1 tag and execute the statements defined at this location:

```
PERFORM ROUTINE1
```

Variables Returned

None

Related Commands

RETURN

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

33 POPDIR

Description

Return to the position in the directory hierarchy that was saved using the `PUSHDIR` command.

You can only use this command, if you have previously issued the `PUSHDIR` command.

Syntax

```
POPDIR
```

Procedure File Example

Findfile.ncp

Variables Returned

None

Related Commands

`PUSHDIR`

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

34 PUSHDIR

Description

Save the current position in the directory hierarchy.

You can only use this command, if you have previously issued the `DOSDIR` command.

This command is useful, for example, when you need to search various directories for specific files.

Syntax

```
PUSHDIR
```

Procedure File Example

Findfile.ncp

Variables Returned

None

Related Commands

`DOSDIR`, `POPDIR`

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

35 QA

Description

Record sessions, terminal emulation screens and user input in Entire Test Client format to disk. These data are used by our support and development for the reproduction of problems.

Using the key combination CTRL+Q, you can switch this mode on and off.

When you issue the `QA` command to switch this mode on, a dialog box appears in which you have to specify a folder and file name for the data to be recorded. The file extension is always *qau*.

Data are recorded until you issue the `QA` command again. Another dialog box appears in which you can enter a short description (up to 39 characters). If you choose the **Cancel** button, a description is not added to the data file.

During terminal emulation, the character Q is shown in column 61 of the status line when `QA` is active.

Syntax

```
QA
```

Variables Returned

None

Related Commands

`REC_BUFF`, `REC_SCR`, `REC_XFER`

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	No

36 QUIT

Description

Close an Entire Connection terminal.

If a logoff procedure has been defined, it is executed before the terminal is closed.

Syntax

```
QUIT
```

Variables Returned

None

Related Commands

CANCEL, EXIT

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	Yes
API:	No

37 RD

Description

Remove a directory.

Syntax

```
RD {[drive:]\directory\...}
```

Examples

- Remove the directory called *NCP* from the current directory:

```
RD NCP
```

- Remove the directory called *NCP* from the root directory of drive C:

```
RD C:\NCP
```

- Remove the directory called *NCP* from the *SAG* directory:

```
RD \SAG\NCP
```

- Remove the directory which is defined by the contents of the variables #FILEDRIVE and #FILEPATH:

```
RD #FILEDRIVE ':' #FILEPATH
```

Variables Returned

#RC (SUCCESS if RD was successful. FAILURE if RD was not successful.)

Related Commands

CD, CHDRIVE, MD

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

38 READ

Description

Read data from an input file.

The file is available across all nested procedure files. You can, for example, issue the `READ` command from more than one procedure file.

Syntax

```
READ filenumber parametervariable ...
```

`filenumber` is a number between 1 and 4.

`parametervariable` represents one or up to 9 of the following variables:

`+PARM1` through `+PARM9` (global)
`#PARM1` through `#PARM9` (local)

The result of a `READ` operation is stored in these variables.

If you specify only one *`parametervariable`*, the entire record is stored in that variable.

If you specify more than one *`parametervariable`*, the record is first split into fields that are separated by blanks. The first field in then placed in the first variable, and so on.

If you specify more variables than fields, the unused variables are reset to null. If there are more fields than variables, the last variable contains the rest of the record.

The maximum ASCII record length supported by the `READ` command is 255 bytes. Therefore, the maximum amount of data that can be stored in one *`parametervariable`* is also 255 bytes.

Examples

- Read input file 1 and store the entire record in a variable:

```
READ 1 #PARM1
```

- Read input file 2 and store the record in four variables:

```
READ 2 #PARM1 #PARM2 #PARM3 #PARM4
```

Procedure File Example

Copyscr.ncp

Variables Returned

#RC (SUCCESS if a record was read successfully. EOF if the end-of-file marker is reached. FAILURE if READ was not successful.)

Related Commands

OPEN-I, CLOSE

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

39 REC_BUFF

Description

Record data untranslated from the terminal emulation buffer to disk.

The buffers are written to the file *Buffer.trc* in the log/trace directory. If the file does not yet exist, it is created. If the file does exist, the new buffers are appended to it.

The buffer is only written to disk when a carriage return (CR) is sent to the mainframe. Buffers are recorded until you issue the `REC_BUFF` command again.

The default key combination depends on the type of session and the corresponding default key scheme.

Type of Session / Default Key Scheme	Default Key Combination
TN3270 / SAGKEYS1	CTRL+B
BS2000 / BS2000KEYS	CTRL+B
VTxxx / VT220PC	ALT+B

Important: This command is provided for Entire Connection problem resolution and should only be used with the assistance and direction of your technical support.

During terminal emulation, the character B is shown in column 75 of the status line when `REC_BUFF` is active.

Syntax

```
REC_BUFF
```

Variables Returned

None

Related Commands

QA, REC_SCR, REC_XFER

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	No

40 REC_SCR

Description

Record terminal emulation screens to disk.

The screens are written to the file *Screen.trc* in the log/trace directory. If the file does not yet exist, it is created. If the file does exist, the new screens are appended to it.

Screens are recorded until you issue the `REC_SCR` command again. Using a key combination, you can switch this mode on and off.

The default key combination depends on the type of session and the corresponding default key scheme.

Type of Session / Default Key Scheme	Default Key Combination
TN3270 / SAGKEYS1	CTRL+S
BS2000 / BS2000KEYS	CTRL+S
VTxxx / VT220PC	ALT+S

Screens are only recorded in terminal emulation mode, or when a procedure file or API program is executed and the system variable `DISPLAY` is set to `ON`.

During terminal emulation, the character S is shown in column 79 of the status line when `REC_SCR` is active.

Syntax

```
REC_SCR
```

Procedure File Example

Recscr.ncp

Variables Returned

None

Related Commands

QA, REC_BUFF, REC_XFER

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	Yes

41 REC_XFER

Description

Record data transfer buffers to disk.

The data transfer buffers are written to the file *Xfer.trc* in the log/trace directory. If the file does not yet exist, it is created. If the file does exist, the new data transfer buffers are appended to it.

Data transfer buffers are recorded until you issue the `REC_XFER` command again. Using a key combination, you can switch this mode on and off.

The default key combination depends on the type of session and the corresponding default key scheme.

Type of Session / Default Key Scheme	Default Key Combination
TN3270 / SAGKEYS1	CTRL+F
BS2000 / BS2000KEYS	CTRL+X
VTxxx / VT220PC	ALT+F

Important: This command is provided for Entire Connection problem resolution and should only be used with the assistance and direction of your technical support.

During terminal emulation, the character X is shown in column 80 of the status line when `REC_XFER` is active.

Syntax

```
REC_XFER
```

Variables Returned

None

Related Commands

QA, REC_BUFF, REC_SCR

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	Yes

42 RECALL

Description

Display in the current input field up to 20 commands and/or character strings that have been entered by the user during terminal emulation.

To use this command, the Recall feature must be enabled in the session properties.

Syntax

```
RECALL
```

Variables Returned

None

Usage

Procedure File:	No
Command Line:	No
Key:	Yes
API:	No

43 RENAME

Description

Rename or move a file.

Syntax

```
RENAME path-1 path-2
```

path is as follows:

{[[*drive*:]*directory*\...\]*filename*[.*extension*]}

To move a file, specify a different directory for *path-2* (the drive must be the same). When you move a file, you can also rename it (optionally). To do so, specify a new file name and/or extension.

Examples

- Rename the file *Test.ncp* to *Test.xyz*:

  ```
  RENAME Test.ncp Test.xyz
  ```

- Move the file *Test.ncp* to the directory *\SAG\NCP*:

  ```
  RENAME Test.ncp \SAG\NCP\Test.ncp
  ```

- Move the file *Test.ncp* to *\SAG\NCP* and rename it to *Down.ncp*:

```
RENAME Test.ncp \SAG\NCP\Down.ncp
```

Variables Returned

#RC (SUCCESS if RENAME was successful. FAILURE if RENAME was not successful.)

Related Commands

ERASE

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

44 RESET

Description

Reset a global or local variable to zero or blank.

Syntax

```
RESET {global-variable|local-variable}...
```

or

```
RESET {GLOBALS|LOCALS}
```

Examples

- Reset the variable #PARM1:

  ```
  RESET #PARM1
  ```

- Reset all global variables:

  ```
  RESET GLOBALS
  ```

- Reset all local variables:

  ```
  RESET LOCALS
  ```

- Reset the local variables #PARM1 and #CNT1, and the global variable +PARM1:

```
RESET #PARM1 #CNT1 +PARM1
```

Procedure File Example

Findfile.ncp

Variables Returned

None

Related Commands

DECR, INCR, SET, SHIFT

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

45 RETURN

Description

Continue processing with the statement that occurs directly after the `PERFORM` command.

You can only use this command, if you have previously issued the `PERFORM` command.

Syntax

```
RETURN
```

Variables Returned

None

Related Commands

`PERFORM`

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

46 REVEAL

Description

Display the field attributes of the emulation and the value of the ASCII character at the current cursor position.

This command can only be used during terminal emulation. The following information is shown in the status line of the terminal emulation screen, starting at column 38:

```
aaa/bbb fields ccccc dd
```

aaa is the number of input fields on the screen.

bbb is the number of unprotected input fields on the screen.

ccccc indicates that the cursor is positioned on one of the following fields:

FM	field mark (attribute)
PROT	protected field
NUM	unprotected numeric field
ALPHA	unprotected alphanumeric field
NON-DISP	non-display field

dd is the ASCII value of the character (in hexidecimal notation) at the current cursor position.

Example:

```
043/001 fields PROT 6e
```

If the REVEAL command is in effect, data transfer is disabled. Data transfer buffers are shown on the screen. The information is shown until you issue the REVEAL command again. Using a key combination, you can switch this mode on and off.

The default key combination depends on the type of session and the corresponding default key scheme.

Type of Session / Default Key Scheme	Default Key Combination
TN3270 / SAGKEYS1	CTRL+R
BS2000 / BS2000KEYS	ALT+R
VTxxx / VT220PC	ALT+R

Syntax

```
REVEAL
```

Procedure File Example

Revattr.ncp

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	Yes

47 RSPMONITOR

Description

Switch the response time monitor for terminal emulation on and off.

When you switch on the response time monitor, the dynamic variables listed below are reset to zero.

During terminal emulation, the character R is shown in column 75 of the status line when the response time monitor is active.

Syntax

```
RSPMONITOR
```

Variables Returned

The following dynamic variables are always set (calculated) whenever you press a terminal emulation key (CLEAR, CR, PF1 etc.).

*RSPCOUNT	Number of transactions (keyboard input)
*RSPTIME	Current response time
*RSPAVG	Average response time
*RSPMIN	Minimum response time
*RSPMAX	Maximum response time
*RSPTOTAL	Total response time

The response time is represented in seconds and hundredths of seconds.

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	Yes
API:	No

48 SCHEDTOP/SCHEDULE

Description

Schedule the execution of further tasks or procedure files.

Processing starts after processing of the current procedure file has finished.

Using SCHEDTOP, tasks or procedure files are scheduled on a first-in-first-out (FIFO) basis. Using SCHEDULE, tasks or procedure files are scheduled on a last-in-first-out (LIFO) basis.

Caution: Do not use the commands SCHEDULE and SCHEDTOP within the same procedure file or group of nested procedure files.

Syntax

```
SCHEDTOP {task|procedure} [parameter] ...
```

```
SCHEDULE {task|procedure} [parameter] ...
```

task is a task defined to Entire Connection.

procedure is a procedure file defined to Entire Connection.

parameter is any valid input parameter for the task or procedure file.

Examples

- Schedule the task BUDGET, which runs a Natural program on the mainframe and downloads data to the ASCII file *Budget.ncd*:

```
SCHEDULE BUDGET Budget.ncd
```

- Schedule the task `BUDGET` as the first task to be executed after processing of the current procedure file has finished:

```
SCHEDTOP BUDGET Budget.ncd
```

- Schedule a task or procedure file whose name is defined by the variable `#PARM1`:

```
SCHEDULE #PARM1
```

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	Yes

49 SET

Description

Assign a value to a local variable, global variable or system variable. You cannot assign a value to a dynamic variable.

The values assigned with `SET` are only valid for the current Entire Connection session.

Using the command line or terminal emulation keys, you can only assign values to global variables and system variables.

Syntax

```
SET variable {string|variable}...
```

Examples

- Set the system variable `LOGON` to `YES`:

```
SET LOGON YES
```

- Set the global variable `+PARM1` to `LOGON = YES`:

```
SET +PARM1 'LOGON = YES'
```

- Set the file name for `PCFILE 5` to *Test.ncd*:

```
SET PCFILE 5 DOWN DATA Test.ncd
```

- Set the XSL stylesheet type and the name of the stylesheet for download to XML:

```
SET PCFILE 7 DOWN CONVERT text/xsl http://PCxyz/xml/employ2.xsl
```

See also: *Specifying a File Name Using the SET Command* in the *Terminal Emulation* documentation.

- Set the local variable #PARM1 to the value which occurs on the terminal emulation screen, starting at row 2, column 1 for a length of 80 characters:

```
SET #PARM1 *SCREEN 2 1 80
```

The syntax for *SCREEN is:

```
*SCREEN [row column length]
```

row is a value between 1 and 25.

column is a value between 1 and 80.

length is a value between 1 and 80.

- Set the local variable #ENVIRONMENT to the value of the DOS environment parameter PATH:

```
SET #ENVIRONMENT PATH
```

Procedure File Example

Findfile.ncp

Variables Returned

None

Rules for SET PCFILE

The file extension defined with SET PCFILE indicates the type of dynamic format conversion that is to be used.

Format	Extension
ASCII	Any other extension not included in this table (e.g. **.ncd*, **.ncs*, **.ncm*, **.ncr* etc.)
Basic	**.prn*
Binary	Any binary file with any extension. The transfer format for Natural is one record with a single binary field.
COBOL	**.ncc*
dBase III	**.dbf*

Format	Extension
Data Interchange Format	*.dif
Encryption	*.enc
Excel	*.xls or *.xlsx (depending on the Excel version)
HTML	*.htm or *.html. This is a specific HTML format that can also be opened with Excel.
Lotus	*.wks, *.wk1, *.wkl
XML	*.xml

System-Generated File Names (Download only)

A file name is automatically generated, if you specify ~~RANDOM instead of a file name in the SET PCFILE statement. The file name is then generated as follows:

```
ddhhmmss.xxx
```

dd is the current day of the month as determined by the system date.

hhmmss is the current time as determined by the system time.

xxx is one of the following file extensions:

- *ncd*, if an extension was not provided for data. Example:

```
SET PCFILE 5 DOWN DATA ~~RANDOM
```

- *ncr*, if an extension was not provided for a report. Example:

```
SET PCFILE 5 DOWN REPORT ~~RANDOM
```

- The extension that was specified with ~~RANDOM. Example:

```
SET PCFILE 5 DOWN DATA ~~RANDOM.XYZ
```

Related Commands

DECR, INCR, RESET, SHIFT

Usage

Procedure File:	Yes
Command Line:	Yes - depends on the type of variable. There are restrictions for system variables concerning whether and where the SET command can be used.
Key:	Yes
API:	Yes

50 SHIFT

Description

Shift the contents of the global or local parameter variables PARM2 through PARM9 down into PARM1 through PARM8 in order to set PARM9 to a null value.

This command is useful, for example, with interactive procedure files (or a set of nested procedure files) which pass an external value to +PARM1. If the value of +PARM1 is to be changed dynamically for each iteration, you pass the values for the variables +PARM1 through +PARM9 and execute the SHIFT command prior to each iteration.

Syntax

```
SHIFT {GLOBALS|LOCALS}
```

Examples

- Shift all global parameter values:

```
SHIFT GLOBALS
```

- Shift all local parameter values:

```
SHIFT LOCALS
```

Procedure File Example

Vars.ncp

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

51 SLEEP

Description

Suspend procedure file processing for a specific period of time.

Unlike the PAUSE command, SLEEP does not check whether data is received from the host.

Syntax

```
SLEEP milliseconds
```

milliseconds can be in the range from 0 through 32000 (i.e. 0 thousands of a second through 32 seconds). The value 1000 is equal to 1 second.

Examples

- Suspend procedure file processing for 1 second (1000 milliseconds):

  ```
  SLEEP 1000
  ```

- Suspend procedure file processing for the number of seconds that is defined by the variable #CNT1:

  ```
  SLEEP #CNT1
  ```

Variables Returned

None

Related Commands

ELAPSETIME, PAUSE, WAIT, WAITFOR, WAITM, WAITUNTIL

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

52 TOGGLE

Description

Toggle between two possible states of a system variable.

These are the system variables for which only the values ON/OFF, YES/NO or TRUE/FALSE can be set.

Syntax

```
TOGGLE systemvariable
```

Examples

- Set variable LOGON from ON to OFF or from OFF to ON:

```
TOGGLE LOGON
```

- Set variable STATUS from ON to OFF or from OFF to ON:

```
TOGGLE STATUS
```

Procedure File Example

Vars.ncp

Variables Returned

None

Related Commands

RESET, SET

Usage

Procedure File:	Yes
Command Line:	Yes
Key:	Yes
API:	Yes

53 TYPE

Description

Send simulated keyboard input to the host or PC.

For all input via terminal function keys (e.g. CLEAR or CR), Entire Connection waits for a response from the host.

Entire Connection waits until a response is received, or until the time period indicated by the system variable RESPONSE is exceeded. If a response is received within the defined period, the next statement in the procedure file is executed.

If the defined period is exceeded, processing of the procedure file is canceled. Exception: if the $TIMEOUT tag is defined in the procedure file, Entire Connection branches to this tag for further processing.

Syntax

```
TYPE {string|terminal-functionkeyname|physical-functionkey}...
```

Examples

- Send the keyboard input CLEAR to the host:

```
TYPE CLEAR
```

- Send the strings *NAT and %+ to the host, and confirm each string with CR:

```
TYPE '*NAT' CR '%+' CR
```

- Send `CLEAR` and `ALT+A` to the host (`ALT+A` can be defined, for example, as `'*NAT' CR`):

```
TYPE CLEAR A-A
```

See also: *Key Schemes*

Procedure File Example

Makete.ncp

Variables Returned

None

Usage

Procedure File:	Yes
Command Line:	No
Key:	Yes
API:	Yes

54 WAIT

Description

Suspend processing of a procedure file until the user presses a command button in the dialog box.

The message (string and variable) may be up to 184 characters long.

Syntax

```
WAIT [{string|variable}...]
```

Examples

- Suspend processing without issuing a message:

```
WAIT
```

- Suspend processing and issue the message that the file name defined in the #FILESPEC variable cannot be found:

```
WAIT 'File' #FILESPEC 'was not found'
```

- Suspend processing and issue the message that procedure file completed successfully:

```
WAIT 'Procedure file completed successfully'
```

Procedure File Example

Parms.ncp

Variables Returned

None

Related Commands

PAUSE, SLEEP, WAITM, WAITFOR, WAITUNTIL, ELAPSETIME

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

55 WAITFOR

Description

Wait for a specific condition on the terminal emulation screen that will be sent by the host.

If the condition is true, the corresponding statement is executed.

If the condition is false, or if the time period defined in the system variable `RESPONSE` has exceeded without receiving a screen from the host, the next statement is executed.

Syntax

```
WAITFOR screenvariable operator string command
```

screenvariable represents the variable `*SCREEN` (syntax see below).

operator is one of the following:

`EQ`	equal
`NE`	not equal
`GE`	greater than or equal to
`LE`	less than or equal to
`GT`	greater than
`LT`	less than

For *command*, you can use any Entire Connection command except the following:

```
IF
IFNOT
WAITFOR
```

Syntax for *SCREEN

The variable *SCREEN can only be used once per WAITFOR.

```
*SCREEN [row column [length]]
```

row is a value between 1 and the maximum number of lines +1.

column is a value between 1 and the maximum line size.

length is a value between 1 and the screen size.

For example:

*SCREEN means the whole screen.

*SCREEN 2 1 means from row 2, column 1 to the end of the screen.

*SCREEN 2 1 80 means from row 2, column 1, the next 80 positions.

Examples

- Check whether the string CP READ occurs on the screen and, if the condition is true, branch to the LOGON tag:

```
WAITFOR *SCREEN EQ 'CP READ' GOTO LOGON
```

- Check whether the string NEXT occurs on the screen, starting in row 2, column 1, and, if the condition is true, send CR to the host:

```
WAITFOR *SCREEN 2 1 EQ 'NEXT' TYPE CR
```

- Check the value defined in the local variable #PARM1 and, if the condition is true, branch to the CONTINUE tag:

```
WAITFOR *SCREEN EQ #PARM1 GOTO CONTINUE
```

Procedure File Example

Waitcmds.ncp

Variables Returned

If the condition is true, the screen position of the string is returned in the following local variables:

#ROW - valid values are between 1 and the maximum number of lines +1

#COL - valid values are between 1 and the maximum line size

Related Commands

IF, PAUSE, SLEEP, WAIT, WAITM

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

56 WAITM

Description

Suspend processing of a procedure file for a specific period of time.

This command is useful with procedure files that are to be processed in UA mode. It may become necessary to inhibit the user from continuing to work for a certain period of time. In this case, a message of explanation can be issued. The message is displayed in the terminal application's output window.

While waiting, Entire Connection checks whether a response from the host has been received. If a response other than a data transfer buffer is detected, waiting is automatically terminated and the next statement in the procedure file is executed. This is useful, if you need to wait for a host process to finish, but you do not want to suspend processing of the procedure file longer than necessary.

The only other way to terminate waiting is to abort the entire procedure file.

Syntax

```
WAITM minutes [message]
```

minutes is a value between 1 and 1440, and *message* may be up to 184 characters long.

Examples

- Wait for one minute:

```
WAITM 1
```

- Wait for one minute and display a message:

```
WAITM 1 'Data not yet available - will retry in 1 minute'
```

Procedure File Example

Waitcmds.ncp

Variables Returned

None

Related Commands

PAUSE, SLEEP, WAIT, WAITFOR, WAITUNTIL, ELAPSETIME

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

57 WAITUNTIL

Description

Suspend processing of a procedure file until a specific date and time.

Syntax

```
WAITUNTIL date time
```

date has the format YYYY/MM/DD. Starting with the year 2000, you must specify the year in the format YYYY. Up to 1999, you can specify the year in the format YY or YYYY.

time has the format HH:MM:SS.

Examples

- Wait until 21. March 1999 at 23:00:

```
WAITUNTIL 1999/03/21 23:00:00
```

- Wait until the date and time defined in the dynamic variable *DATE and the local variable #PARM1:

```
WAITUNTIL *DATE #PARM1
```

Procedure File Example

Waitcmds.ncp

Variables Returned

None

Related Commands

PAUSE, SLEEP, WAIT, WAITFOR, WAITM, ELAPSETIME

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No

58 WRITE

Description

Write data to an output file. The maximum record length is 255 bytes.

The file is available across all nested procedure files. You can, for example, issue the `WRITE` command from more than one procedure file.

Syntax

```
WRITE filenumber [{string|variable}...] [;]
```

filenumber is a number between 1 and 4.

If you do not specify an operand, an empty record is written (i.e. the file only contains CR/LF).

If you specify a semicolon (;), CR/LF is not written to the file.

Examples

- Write a blank line to output file 1:

```
WRITE 1
```

- Write the contents of the local variables #PARM1 to #PARM4 to output file 2:

```
WRITE 2 #PARM1 #PARM2 #PARM3 #PARM4
```

- Write the contents of the local variables #PARM1 to #PARM4 without CR/LF to output file 2:

```
WRITE 2 #PARM1 #PARM2 #PARM3 #PARM4 ';'
```

- Write a string and the current date to output file 1:

```
WRITE 1 'Today is' *DATE
```

Procedure File Example

Vars.ncp

Variables Returned

#RC (SUCCESS if WRITE was successful. FAILURE if WRITE was not successful.)

Related Commands

OPEN-O, CLOSE

Usage

Procedure File:	Yes
Command Line:	No
Key:	No
API:	No