

Procedure Files

This chapter describes how to work with procedure files.

- What is a Procedure File?
 - Executing a Procedure File
 - External Parameters
 - Addressing a Host Session from a Procedure File
 - Aborting or Terminating a Procedure File
 - Creating a Procedure File Using Learn Mode
 - Debugging a Procedure File
 - Conventions
 - Command Categories
 - Variables in Procedure Files
-

What is a Procedure File?

Procedure files (extension *nep*) initiate operations which are executed on the host, the PC, or both. You can create procedure files using a PC editor, a host editor or in learn mode.

For example, you can create a procedure file which automates the process of logging on to a host system. All required logon commands are contained in the procedure file and are executed when the procedure file is invoked.

Example - logging on to a VM mainframe system:

```
:LOGON
  TYPE CR
  IF *SCREEN NE 'CP READ' GOTO ERROR
  TYPE 'LOGON' *USERID1 CR
  TYPE *PASSWORD1 CR
  SET #CNT1 0
:VM
  IF *SCREEN EQ 'VM/SP RELEASE 4' GOTO NAT
  INCR #CNT1
  IF #CNT1 GT 10 GOTO ERROR
  PAUSE 200
  GOTO VM
:NAT
  TYPE 'NAT' CR
  GOTO FINI
:ERROR
  MSG 'CANNOT LOGON - TRY LATER'
:FINI
  EXIT
```

Sample procedure files are copied to your hard disk during installation (if specified).

Executing a Procedure File

You can execute a procedure file in the following ways:

- Select the procedure file from the task list.
- Enter the name of the procedure file and, if applicable, all required parameters in the command line.
- Execute a procedure file which invokes another procedure file, see below.
- Press the key or key combination that is defined for the procedure file.

To invoke a procedure file from another procedure file

1. Open the procedure file from which you want to invoke another procedure file.
2. Insert the EXECUTE command at the appropriate location:

```
EXECUTE filename
```

3. Save your modifications.
4. Execute the modified procedure file.

External Parameters

You can pass up to 9 external parameters to a procedure file. The external parameters can be checked within the procedure file, using the appropriate parameter variables.

Example:

The following command executes the procedure file MYPROG with the parameters 1234, abc and Abc:

```
MYPROG 1234 abc 'Abc'
```

See also: *Task Parameters* in the *Overview of Object Properties*.

Addressing a Host Session from a Procedure File

A host session can be addressed from a procedure file in the following ways:

- **Explicitly connect to a specific host session**
Use the CONNECT command:

```
CONNECT sessionname
```
- **Implicitly connect to the active host session**
If you execute a procedure file while a host sessions is open.

- **Implicitly connect to the default host session**

If you execute a procedure file while no host session is open, the default session is automatically opened when the first statement addressing a host session is encountered. The default session is specified in the user properties. Commands that require open sessions are, for example, `TYPE` and `IF *SCREEN`.

Aborting or Terminating a Procedure File

The execution of a procedure files is aborted or terminated in the following cases:

- When you choose the **Cancel** button in a dialog box that was invoked by the procedure file.
- When a non-recoverable error occurs during procedure file processing (for example, a syntax or logic error).
- When the maximum host response time is exceeded.
- When the `EXIT` or `CANCEL` command is issued from within the procedure file.
- When a normal exit from the procedure file occurs (i.e. when the last statement is executed).

You can also abort the currently active procedure file as described below.

▶ **To abort the currently active procedure file**

- From the **Utilities** menu, choose **Cancel Procedure**.

Or:

Choose the following toolbar button:



Creating a Procedure File Using Learn Mode

In learn mode, all keyboard input entered in a host session is stored in a procedure file. The commands in the procedure file are automatically executed when you execute the procedure file.

For example, you can use learn mode to record all keyboard input required to logon to the host.

Caution:

Though learn mode creates procedure files that can be executed successfully, they are not robust as long as you do not add error and exception handling.

▶ **To invoke learn mode**

1. Open a host session.
2. Use the key combination `CTRL+L` to invoke learn mode.

`CTRL+L` is the default. You can also define another key combination.

The **Learn** dialog box appears.

3. Select the desired procedure file from the list box, or specify the name of the (new) procedure file with the extension *ncp* in the **File name** text box.

If you specify an existing procedure file, it will be overwritten by the new procedure.

4. Choose the **OK** button.

Learn mode is now active and all keyboard input is recorded.

▶ To disable learn mode

- Use the key combination CTRL+L (default) to switch off learn mode.

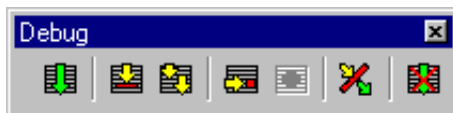
The resulting procedure file contains all of your keyboard input. You can modify this procedure file using an ASCII editor (for example, to include check routines or logical queries).

Debugging a Procedure File

When debugging a procedure file, the following additional elements are shown in the terminal application window:

- The **Debug** menu. This menu is only available as long as you debug a procedure file. It provides the same commands as the debug toolbar.
- A debug toolbar. This toolbar is only available as long as you debug a procedure file.

You can move it to the application window (e.g. below the menu bar or next to the terminal emulation screen) so that it is no longer shown in a window. To prevent docking, press CTRL while moving the window.



- A procedure window on the left. The display starts with the first executable source text line which is indicated by an arrow.

```

▶ set display off
  set status on

* Is Natural currently active ?

:start
  if *screen eq 'NEXT' goto loop1
  beep
  wait 'NATURAL not active on the mainframe - Must be on NEXT prompt'
  exit








```

- A variable window on the right, providing a list of all variables that are used in the procedure file.

Variable	Value
#parm0	D:\Program Files\Software AG\Entire I
#parmno	00

You can modify the size of the individual windows by dragging a window border with the mouse. You can also display these windows as dockable windows.

Using the following commands or buttons in the debug toolbar, you debug the procedure file.

	CTRL+SHIFT+R	Run Procedure.
	CTRL+SHIFT+S	Step Procedure.
	CTRL+SHIFT+A	Animate Procedure.
	CTRL+SHIFT+P	Toggle Breakpoint.
	CTRL+SHIFT+B	Break Procedure.
	CTRL+SHIFT+E	Set Echo Time. The Set Echo Time dialog box appears, in which you can specify the desired delay in milliseconds before execution of the next statement. See the system variable ECHO.
	CTRL+SHIFT+T	Terminate Debug.

▶ To debug a procedure file

1. From the **Utilities** menu, choose **Task List**.

The **Select Task or Procedure** dialog box appears.

2. Select the **Procedures** option button.
3. Select the procedure to be debugged.
4. Choose the **Debug** button.
5. Use the **Debug** menu or debug toolbar to debug the procedure file.

6. When debugging has been completed, terminate the debug session using the corresponding command.

Conventions

The following rules apply:

- Each line in a procedure file can consist of up to 127 characters.
- A line can contain only one command and related operands.
Exception: the commands IF, IFNOT und WAITFOR.
- Statements can be indented for readability.
Exception: tags and comment lines must start in the first column.
- Elements such as tags and commands are not case-sensitive. Character strings can be case-sensitive.
- The limit for the source code is 64 KBytes, including comments. If this size is exceeded, an error message appears and the procedure aborts.

A line can consist of the following:

- Command String
- Tag
- Comment
- Blank Line

Command String

A character string must be enclosed with quotation marks if case-sensitivity is required or if there are embedded blanks. You can either use apostrophes or double quotation marks, as long as they are paired consistently. For example:

```
'This string is a valid single quoted string'  
'This string is "also" valid'  
"This string is a valid double quoted string"  
"This string is 'also' valid"  
"This string is invalid because the double quotation marks are not paired'
```

Entire Connection expects character strings to be in the Windows character set (ANSI).

Tag

A tag is a branch point within a procedure file. The name of a tag must always begin with a colon and can be up to 127 characters in length. A tag that is not unique results in an error message.

Examples for tags:

```
:START  
:LOGON-ERROR  
:QUIT
```

\$TIMEOUT and \$ESC are tags with specific meanings (see below).

\$TIMEOUT

When the \$TIMEOUT tag is used in a procedure file and a timeout condition occurs (for example, when the host does not respond within the defined period of time), control is automatically passed to the statement that follows the tag.

The system variable RESPONSE defines how long Entire Connection is to wait for a response from the host before processing is to continue with the next statement after the tag.

\$ESC

When the \$ESC tag is used in a procedure file and the user chooses **Cancel Procedure** from the **Utilities** menu (or the corresponding toolbar button), control is automatically passed to the statement that follows the tag.

If the \$ESC tag is not used in a procedure file and the user chooses the **Cancel Procedure** command, procedure file processing is aborted.

If a procedure file is executing a task that can be aborted, the following applies:

- When the task is aborted, control is passed back to the procedure file.
- When the user then chooses the **Cancel Procedure** command, control is passed to the \$ESC tag.

Comment

A comment line starts with an asterisk (*) in the first column. For example:

```
* This is a sample comment line  
*  
* The line before this one is also a comment line
```

A comment can also be inserted after a statement. In this case, it starts with a slash followed by an asterisk (/*). For example:

```
SET LOGON NO /* this is a comment after a statement  
:ERROR      /* this is a comment after a tag
```

Blank Line

You can use blank lines anywhere within a procedure file. Entire Connection ignores them.

Command Categories

The commands that can be used in a procedure file can be divided into the following categories:

- Host Communication
- Processing
- Input/Output
- Task Management
- Operating System Functions
- Environment Control

Host Communication

Using the following commands, you can maintain host sessions from within a procedure file:

Command	Description
CONNECT	Open a host session.
DISCONNECT	Close a host session.
EMULATE	Switch to terminal emulation mode.
QA	Record sessions, terminal emulation screens and user input in Entire Test Client format to disk.
REC_BUFF	Record data untranslated from the terminal emulation buffer to disk.
REC_SCR	Record terminal emulation screens to disk.
REC_XFER	Record data transfer buffers to disk.
REVEAL	Display the field attributes of the 3270 emulation and the value of the ASCII character at the current cursor position.
RSPMONITOR	Switch the response time monitor for terminal emulation on and off.
SUSPEND	Deactivate the current host session.
TYPE	Send simulated keyboard input to the host or PC.

Important:

The REC_BUFF and REC_XFER commands are used for Entire Connection problem resolution and should only be used with the assistance and direction of your technical support.

Processing

Using the following commands, you can determine the processing logic for a procedure file. For example, depending on the data transmitted from the host, you can execute different commands.

Command	Description
DECR	Subtract 1 from the global and local counter variables or screen position variables.
ELAPSETIME	Calculate the difference, in seconds, between two date and time stamps.
EXIT	Leave a procedure file and return to the previous procedure file or to Entire Connection.
GOTO	Branch to another location in the procedure file.
IF/IFNOT	Check a condition.
INCR	Add 1 to the global and local variable counters or screen position variables.
INPUT	Prompt the user for input.
LEARN	Create a procedure file in learn mode.
PAUSE	Suspend processing for a specific period of time.
PERFORM	Branch to another location in the procedure file and execute the statements defined at this location.
RESET	Reset a local or global variable to zero or blank.
RETURN	Continue processing with the statement that occurs directly after the PERFORM command.
SET	Assign a value to a local, global, or system variable.
SHIFT	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.
SLEEP	Suspend procedure file processing for a specific period of time.
TOGGLE	Toggle between two possible states of a system variable.
WAIT	Suspend processing of a procedure file until the user presses a key.
WAITFOR	Check a condition on the next terminal emulation screen that will be sent by the host.
WAITM	Suspend processing of a procedure file for a specific period of time.
WAITUNTIL	Suspend processing of a procedure file until a specific date and time.

Input/Output

Using the following commands, you can read files and write data to files.

Command	Description
CLOSE	Close a file.
OPEN-I	Open a file from which data are read (input file).
OPEN-O	Open a file into which data are written (output file).
READ	Read data from an input file.
WRITE	Write data to an output file.

Task Management

Using the following commands, you can schedule or execute other procedure files or tasks. You can break down complex operations into more manageable, separate procedure files, or execute a number of operations based upon external variables.

Command	Description
EXECTASK	Execute an Entire Connection task.
EXECUTE	Execute a procedure file.
SCHEDTOP	Schedule further tasks or procedure files on a first-in-first-out (FIFO) basis.
SCHEDULE	Schedule further tasks or procedure files on a last-in-first-out (LIFO) basis.

Operating System Functions

Using the following commands, you can invoke operating system functions from within a procedure file.

Example: using a procedure file, you can download data from the host in regular intervals. In order to write this data to a PC file, the procedure file must rename or erase the PC file that was created during the previous download. Otherwise, manual modifications are required prior to each execution of the procedure file.

Command	Description
CD	Change the current directory on the current drive.
CHDRIVE	Change the current drive.
CHMOD	Change the file attributes.
DOS	Execute a DOS command.
DOSDIR	Display directory information.
ERASE	Erase a file.
MD	Make a directory.
OS	Same as the DOS command.
POPDIR	Return to the position in the directory hierarchy that was saved using the PUSHDIR command.
PUSHDIR	Save the current position in the directory hierarchy.
RD	Remove a directory.
RENAME	Rename or move a file.

Environment Control

The following commands are available from within a procedure file:

Command	Description
BEEP	Sound the PC alarm.
CANCEL	Abort processing of a procedure file.
LOG	Write a message to the log file.
MSG	Display a message.
QUIT	Close an Entire Connection terminal.

Variables in Procedure Files

Procedure files can contain variables, where the values of the variables are determined either by the user or by the system.

For complex operations that involve nested procedure files, global variables that are accessed from multiple procedure files can be defined. A global variable can, for example, pass a return code from one procedure file to another. This prevents processing of subsequent program steps if a preceding procedure file has not been completed successfully.