

Executing Command Scripts

This subsection explains the following commands which can be used in, or in conjunction with, command scripts:

- CONTINUE session command
- MACPARM session command
- MESSAGE session command
- PAUSE session command
- PLAY function command
- RECORD session command
- REMARK session command

You will find examples of how to generate scripts using a macro under Generating Scripts using a Macro.

PLAY Function Command

Natural commands can be written and stored in a member. Such a member is called a command script. The script can be executed with the PLAY function command. The commands are then executed sequentially. Scripts executed by the PLAY command are written to the User Workpool (see the subsection Scripts in the User Workpool).

Command scripts can be stored as any of the following object types:

- Natural object
- PDS member
- Workpool output file (see below)
- VSE/ESA member
- Macro
- BS2000/OSD LMS element

Example:

Assume Natural member MYPROCA in library MYLIB contains the following lines:

```
ED MYLIB ( PROGA ) ; STOW ; END
ED MYLIB ( PROGB ) ; STOW ; END
ED MYLIB ( PROGC ) ; STOW ; END
```

The command:

```
PLAY MYLIB ( MYPROCA )
```

edits and stows the Natural programs PROGA, PROGB and PROGC.

- A script that has been interrupted by an included PAUSE command (see below) can be cancelled by the session command PLAY OFF. This command also deletes the workpool entry (see the subsection Scripts in the User Workpool).
- If an invalid command is detected in a command script during execution, and the script does not contain a CONTINUE statement, the script is stopped automatically and the invalid command is displayed in the Natural command line. You can correct the faulty command and reexecute it by pressing Enter. You can then continue the script by issuing the PAUSE session command (see below).
- If one of the played commands results in an error message (for example, if program PROGB in the above example does not contain correct Natural source, the same logic applies as for invalid commands).
- A command script held in a PDS, VSE/ESA or LMS member may not contain line numbers within the

member's lines (for example, in columns 73 to 80). Thus, when editing such a command script, it is recommended that you switch off the number mode of the Editor, either with the profile settings or by issuing one of the Editor commands AUTOREN OFF or UNREN. You should be aware that the line numbers could have been inherited unintentionally when copying lines from one EDIT session to another with the line commands mentioned in this documentation. Line numbers contained in script lines could result in unpredictable error messages during command script execution.

- It is possible to use nested PLAY commands in a command script. The contents of a member to be played is always entered at the top of the ##PLAY workpool session.

Example:

Assume the member EX1 contains the following lines:

```
EDIT NAT MYPROG
  PLAY NAT EXPLAY(CHG-AB)
EDIT NAT MYPROG1
  PLAY NAT EXPLAY(CHG-AB)
```

and assume member CHG-AB contains the following lines:

```
CHANGE 'A' 'B' all
  STOW
  END
```

The command:

```
PLAY EX1
```

starts an edit session with Natural member MYPROG, plays the commands in CHG-AB, and only then is the command EDIT MYPROG1 processed.

- In the COMMAND field in your user profile, you can specify a command that is executed every time you start Natural. You can specify a PLAY command in this profile field to start a command script when you log on (see also the description of the COMMAND field in the User Defaults subsection of Section Profile Maintenance).
- It might be useful when writing a command script to start it with RECORD ON and finish it with RECORD OFF. This causes all messages to be recorded in the User Workpool (see the RECORD session command below).

RECORD Session Command

You can record Natural commands using the RECORD session command. After you have issued the RECORD session command, all ensuing commands are recorded until you issue the RECORD OFF session command. The recorded commands can be found in the User Workpool in member ##RECORD. This member can be PLAYed.

If a command issued during the recording session causes a message, the message is also recorded in the User Workpool member ##RECORD, preceded by two asterisks **. The message is ignored when the member is executed with the PLAY command.

PAUSE Session Command

In some cases it is useful to interrupt a script being executed, for example in order to manipulate some data manually, and then continue executing the script. This can be done by writing the session command PAUSE in the script. When the script is executing, it stops at the place the PAUSE command was entered. To continue the script, simply issue the PAUSE command manually.

Example:

Assume the member MYPROCB contains the following lines:

```
EDIT MYJOB
  PAUSE
  SUB
  FOLLOW
  CAN
```

The command:

```
PLAY MYPROCB
```

starts an edit session with the member MYJOB and then stops in order to allow modification of the JCL. If you then issue the PAUSE command from the Natural command line, the JCL is submitted, job status messages are displayed (FOLLOW command) and the edit session is cancelled.

Note:

The PAUSE command must always be the last command or the only command in a script line.

Scripts in the User Workpool

A script which is executed by the PLAY command is stored in the User Workpool in an entry named ##PLAY. When a script is interrupted by a PAUSE command or an error, the lines not yet executed are in the workpool member ##PLAY and can be modified.

CONTINUE Session Command

The CONTINUE command can be used in command scripts to gain more flexible control in error situations. If no CONTINUE statement is in the command script, the script is set to PAUSE mode after an error.

If a CONTINUE statement (which can be compared to a label) is in the script, the following actions are taken:

1. RECORD ON is set internally if not activated by the user.
2. The command causing the error and the message is recorded.
3. All lines of the script until the next CONTINUE command are deleted and execution of these lines is skipped.
4. Processing continues with the next CONTINUE statement. All following statements are executed.
5. Termination resets RECORD to its previous value and informs the user if an error has occurred.

Example:

```
KEYS 3 PAUSE
  HELP VERIFY
  MESSAGE 7480
  TECH
  .....
  CONTINUE
  REMARK PROCESSING WILL CONTINUE HERE AFTER ERROR
  KEYS 3 END
```

The above script modifies the user profile. By using the CONTINUE command it makes sure that, after execution of the script, PF3 is reset to its default value from the user profile, even if errors have occurred during execution of the script.

MACPARM Session Command

The MACPARM command is used in command scripts to put data on the Natural stack which is read by a macro using an input statement later in the command script.

This avoids prompting by the macro for parameters, when using macros in command scripts. The MACPARM command must be the only command in a source line.

The command format is:

```
MACPARM p1
```

Explanation of parameters:

Parameter	Meaning
p1	Maximum length of this parameter is 50 bytes and it can contain blanks.

Examples:

The commands:

```
MACPARM LS PDS JW(A*)
PLAY MAC MAC1
```

pass the command LS PDS JW(*) to macro MAC1.

Another useful example can be found in the member VERIFY in the Natural Example Library.

MESSAGE Session Command

The MESSAGE command can be used in command scripts to display a text during execution of a script on the screen and to interrupt the active command script. The MESSAGE command must be the only command in a source line.

The command format is:

```
MESSAGE p0,p1,...pn
```

Explanation of parameters:

Parameter	Meaning
p0	Must be a 4-digit error message number. First, the user library SYSISPFU is searched for the message text. If it does not exist, it is taken from the system library SYSISPS1.
p1,...pn	Optional parameters which are used to replace variable parameters (:1: :n:) in the text. Parameters must usually be separated with your parameter delimiter, usually a comma (,) and can contain blanks.

Examples:

The command:

```
MESSAGE 6812,MYPROG
```

results in the following message, if no text for this number is available in the user library SYSISPFU:

```
Member MYPROG not found
```

The command:

```
MESSAGE 6809,Please enter some text
```

results in the following message, if no text for this number is available in the user library SYSISPFU:

```
Please enter some text
```

Another useful example can be found in member VERIFY in the Natural Example Library.

REMARK Session Command

The REMARK command is used in command scripts to document the command script. The REMARK command must be the only command in a source line.

The command format is:

```
REMARK text
```

Example:

```
REMARK The following command extracts all members  
REMARK including the string Adabas  
LIST PDS JW(*) SC=Adabas
```

Generating Scripts using a Macro

A command script can be generated by a macro. With this mechanism, scripts can be created dynamically.

Example: Generate prompt for CHANGE

For example, executing the following macro with the PLAY command generates a prompt for a CHANGE command to be used on a member, with a choice of a STOW or SAVE command after the change is made:

```

§ RESET #MEMBER(A8) #FROM(A16) #TO(A16) #STOW(A1)
§ INPUT(AD=MI) 'Change' #FROM 'To' #TO 'in member' #MEMBER
§      / 'Stow?' #STOW
EDIT NAT §#MEMBER
CHANGE '§#FROM' '§#TO' all
§ IF #STOW NE ' ' DO
STOW
§ DOEND
§ ELSE DO
SAVE
§ DOEND
END

```

Example: Installation Verification

The following macro generates a command script which verifies Natural installation. It also provides you with a working example of how the commands CONTINUE, MACPARM, MESSAGE and REMARK function:

```

§ DEFINE DATA LOCAL
§ 1 #SUB-SYSTEM (A3)
§ 1 #SUB-SYSTEM-INDEX(N1)
§ 1 #SUBSYS-ARRAY (4)
§ 2 #SUBSYS-LONG (A10) INIT <'NATURAL', 'OS/390', 'VSE', 'BS2000/OSD'>
§ 2 REDEFINE #SUBSYS-LONG
§ 3 #SUBSYS-SHORT (A3)
§ END-DEFINE
§ IF *DATA GT 0
§ INPUT #SUB-SYSTEM
§ EXAMINE #SUB-SYSTEM TRANSLATE INTO UPPER
§ EXAMINE #SUBSYS-SHORT(*) FOR #SUB-SYSTEM INDEX #SUB-SYSTEM-INDEX
§ END-IF
§ IF #SUB-SYSTEM-INDEX EQ 0
§ #SUB-SYSTEM-INDEX := 1 /* NATURAL is default
§ END-IF
§ #SUB-SYSTEM := #SUBSYS-SHORT(#SUB-SYSTEM-INDEX)
KEYS 3 PAUSE
HELP VERIFY
MESSAGE 7480,§#SUBSYS-LONG(#SUB-SYSTEM-INDEX)
TECH
END
§ IF #SUB-SYSTEM-INDEX EQ 1
PLAY NAT VERIFYO
§ ELSE
MACPARM §#SUB-SYSTEM-INDEX
PLAY MAC VERIFYS
§ END-IF
CONTINUE
REMARK Processing will continue here after error
REMARK and pf3 will be reset to its initial profile value
KEYS 3 INITIAL
RECORD OFF

```