NAF - Natural Features Supported

This chapter describes the Natural features supported by Natural Advanced Facilities and how these features can be used.

- DEFINE PRINTER Statement
- Using Con-form to Emphasize Text
- Hardcopy Facility - %H
- Using FETCH and STACK Statements
- ET/BT Logic
- Recovering after Abnormal Ends
- Batch Utilities NSPOBAT, SPPBATPR and SPPPRINT
- Special User Exits
- Load and Unload Programs SPPULDUS and SPPLODUS

DEFINE PRINTER Statement

The `DEFINE PRINTER` statement is used to assign a symbolic name to a report number and to control the allocation of a report to a logical destination (printer).

Note:
The `DEFINE PRINTER` statement is fully described in the Natural Statements documentation. This section only describes the clauses that apply for Natural Advanced Facilities.

This section also covers information on *Mixed Reports by using DEFINE PRINTER*.

The `DEFINE PRINTER` statement syntax for NAF below is different from the general `DEFINE PRINTER` statement as not all of the keywords apply:

```
DEFINE PRINTER ([logical-printer-name = ]n )
               [OUTPUT operand1 ]
               [ PROFILE operand2  ]
               [ DISP operand3     ]
               [ COPIES operand3   ]
               [ PRTY operand4     ]
                                    ... 4
```

`logical-printer-name` is the name which is to be allocated to printer *n*. This is the name which will be used for the *`rep`* notation in a `DISPLAY` or `WRITE` statement. The value for *`n`* may be in the range of `1-31`.

The OUTPUT operand is the destination within the online spooling system and is a logical printer (LPF). This logical printer must be defined on the spool file (see Function 31.2), but need not be part of the currently active user profile.

With the PROFILE clause, you specify as *operand2* the name of a printer control characters table as defined in the NTCC macro or on the spool file (see NTCC table, Function 31.8).

For the DISP clause, the possible values for *operand2* are DEL, HOLD and KEEP. If the DISP clause is omitted, the Disposition specified for the logical printer is used.

With the COPIES clause you specify the number of copies to be printed. The possible values for *operand3* are 1-255. If the COPIES clause is omitted, the number of copies specified for the logical printer is used.

With the PRTY clause, you specify the priority for spool-out. The possible values for *operand4* are 1-255 (where 1 is the lowest and 255 is the highest priority). If the PRTY clause is omitted, the priority specified for the logical printer is used.

In general, Natural Advanced Facilities uses the logical printer definitions in the physical main storage (not the ones on the spool file). One has to distinguish two different situations:

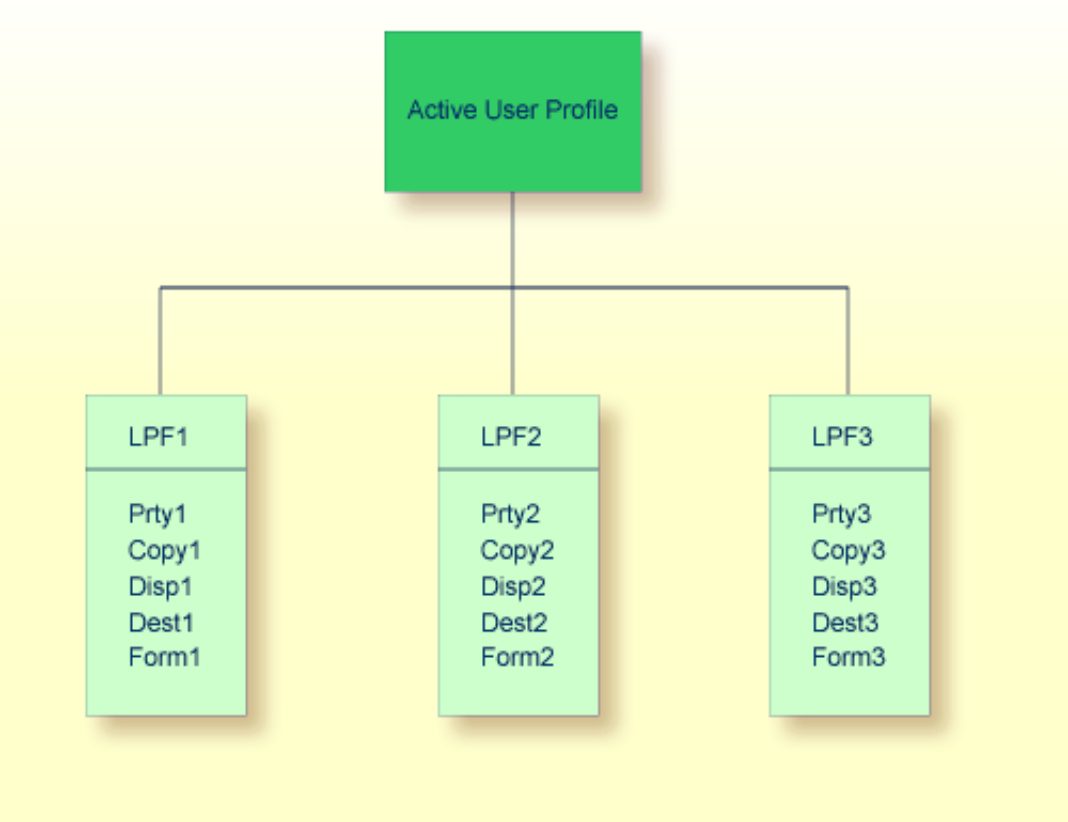
- the requested logical printer is contained in the active user profile,
- the requested logical printer is *not* contained in the active user profile.

If a DEFINE PRINTER statement for printer *n* requests a logical printer which is contained in the active user profile (see Example 1 below), logical printer definitions in storage are *not* overwritten. A subsequent Natural program will therefore encounter the same logical printer definitions. The logical printer definitions will be valid until another DEFINE PRINTER statement causes them to be overwritten.

If a DEFINE PRINTER statement for printer *n* requests a logical printer which is *not* contained in the active user profile (see Example 2 below), the *n*- th logical printer definition in storage will be overwritten by the requested one. A subsequent Natural program will encounter altered logical printer definitions as established by the previous DEFINE PRINTER statement (because it is assumed that the altered logical printer definitions are also to be used by subsequent Natural programs.)

The following examples demonstrate how the DEFINE PRINTER statement works.

Example 1 - Logical Printer Contained in the Active User Profile:

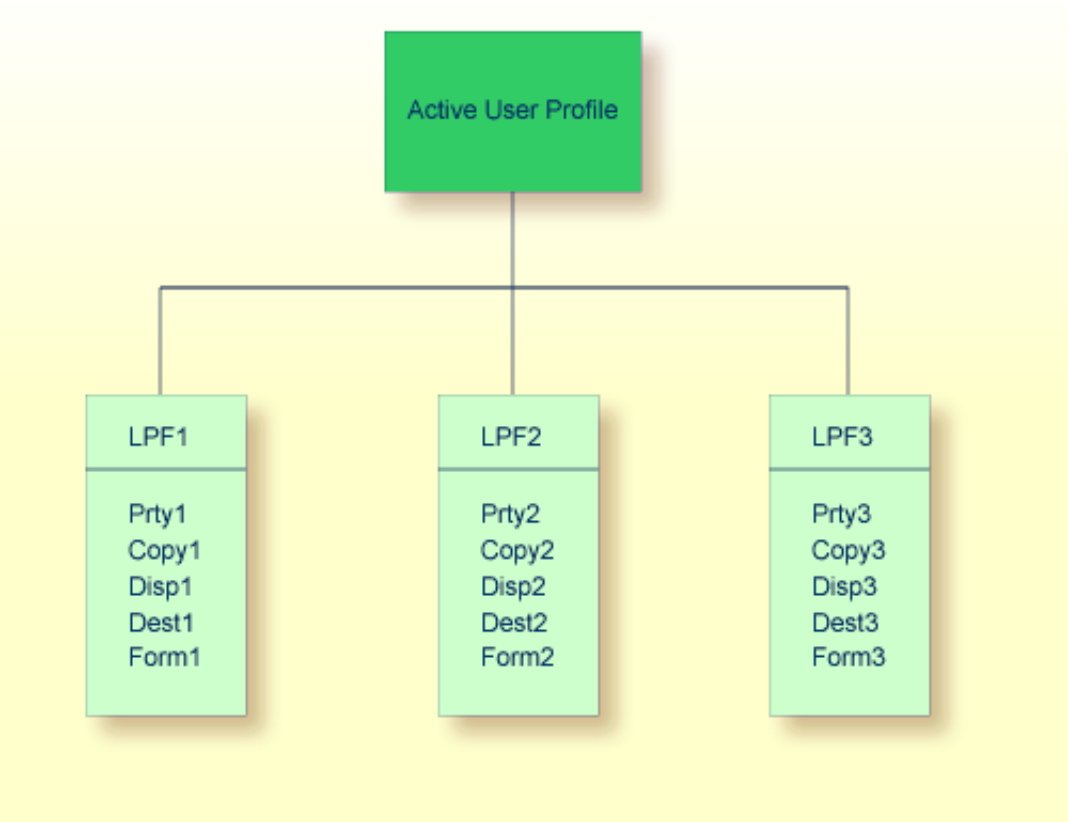



```
DEFINE PRINTER (2) OUTPUT 'LPF1'
...
...
WRITE (2) 'text'
...
WRITE (1) 'different text'
...
```

When the above Natural program is run, the output of the `WRITE (2)` statement is spooled by using the definitions of `LPF1`. The `WRITE (1)` statement uses `LPF1` by default, since there is no `DEFINE PRINTER (1)` statement. Therefore, both `WRITE (1)` and `WRITE (2)` statements use the same logical printer.

If a subsequent Natural program executes a `WRITE (2)` statement, and if the program does *not* contain a `DEFINE PRINTER (2)` statement, the output is spooled by using the definitions of `LPF2`.

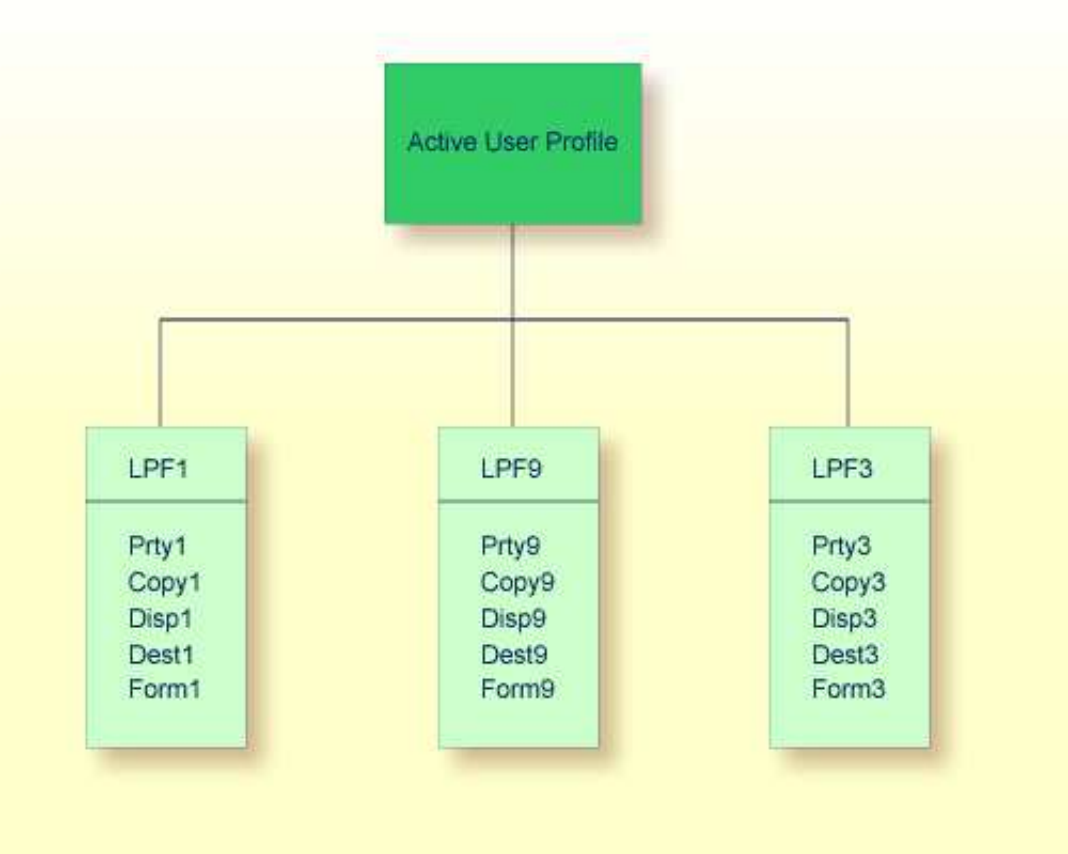
Example 2 - Logical Printer is Not Contained in the Active User Profile:



If the OUTPUT operand of the DEFINE PRINTER statement is *not* identical to one of the logical printer names in the active user profile, the values for the printer profile are overwritten with the new values specified in the OUTPUT operand.

```
...
DEFINE PRINTER (2) OUTPUT 'LPF9'
...
```

The OUTPUT operand of the above DEFINE PRINTER statement causes the definitions of LPF2 to be overwritten with the values assigned to LPF9, including the logical printer name. The active user profile is changed to the following:



This definition remains in effect for the user profile until another `DEFINE PRINTER (2)` statement is executed. Thus, the output of all subsequent `WRITE (2)` statements within the same Natural session is spooled by using the definitions of `LPF9`, if no other `DEFINE PRINTER (2)` statement is executed.

Mixed Reports by using DEFINE PRINTER

Natural programs sometimes need to create different reports for the same physical printer. In this case, attention has to be paid to the `OUTPUT` operand of the `DEFINE PRINTER` statement.

If you use `DEFINE PRINTER` statements for different logical printers but with the *same* `OUTPUT` operand (that is, the same logical printer name), only *one* report is created. This report contains the output of all `WRITE`, `PRINT` or `DISPLAY` statements in a mixed sequence.

Example:

```
DEFINE PRINTER (1) OUTPUT 'LPF1'     /* LPF same as in (0020)
DEFINE PRINTER (2) OUTPUT 'LPF1'     /* LPF same as in (0010)
WRITE (1) 'This is for report 1'
WRITE (2) 'This is for report 2'
WRITE (1) 'This is for report 1'
END
```

If you use DEFINE PRINTER statements for different logical printers and with *different* OUTPUT operands (that is, different logical printer names), *multiple* reports are created. To route these reports to the same physical printer, the same Destination/Form must be specified for the logical printers.

Example:

```
DEFINE PRINTER (1) OUTPUT 'LPF1'     /* LPF different from (0020)
DEFINE PRINTER (2) OUTPUT 'LPF2'     /* LPF different from (0010)
WRITE (1) 'This is for report 1'
WRITE (2) 'This is for report 2'
WRITE (1) 'This is for report 1'
END
```

Using Con-form to Emphasize Text

You can emphasize all or parts of printed reports by using Con-form instructions, for example, during a Con-nect session.

Support is provided for the following Con-form instructions:

Instruction	Description
`.BF`	The text lines between two `.BF` instructions are printed in boldface.
`.BP`	Text contained on the next input line is printed in boldface.
`.US`	Text contained on the next input line is underscored.
`char` `B`	Backspace to super-impose one character over another.
`char` `U1...` *`char`* `U0`	Text within the symbols `U1` and `U0` is underscored.
`char` `M1...` *`char`* `M0`	Text within the symbols `M1` and `M0` is printed in boldface.

`char` is any special character which has been defined as the escape character by using the following Con-form instruction:

```
.OP ESC=char
```

Note:
Text can only be printed in boldface if the Natural profile parameter `INTENS` (see the Natural *Parameter Reference* documentation) has been set to a value greater than `1`.

Hardcopy Facility - %H

The Natural terminal command `%H`, when issued in response to a prompt, produces output from Natural reports and communication screen layouts on a printer. The output will be routed to the first `FREE` physical printer allocated for the defined logical printer for hardcopy (see Functions 12, 30.5, 31.1 and 33). The `%H` command is effective for the current page and is automatically disabled at the end of the program output.

If %H is used for a map created by an `INPUT` statement, the complete contents of the page buffer is spooled to the output device.

The same applies for the SET CONTROL 'H' statement.

Using FETCH and STACK Statements

The NATSPOOL nucleus closes reports when Natural returns to command mode. If a Natural application program uses a STACK COMMAND statement to load another Natural program, command mode is entered and reports created by the invoking program are closed and printed according to their Disposition (provided that Natural is in ET status, that is, no user END OF TRANSACTION statement is pending). However, if a FETCH statement is used, command mode is not entered internally and reports created by the invoking program are closed and printed only when the invoked program ends. This allows more than one Natural program to be involved in the creation of a single report.

Example 1 - Using a STACK Statement:

```
* PGM-1
WRITE (1) 'output from PGM-1'
STACK COMMAND 'PGM-2'
END

* PGM-2
INPUT 'something' F1 (A8) (AD=MI)
END
```

When Program PGM-1 is executed, the report created by the WRITE (1) statement is closed and printed immediately.

Example 2 - Using a FETCH Statement:

```
* PGM-1
WRITE (1) 'output from PGM-1'
FETCH 'PGM-2'
END

* PGM-2
WRITE (1) 'output from PGM-2'
INPUT 'something' F1 (A8) (AD=MI)
END
```

When Program PGM-1 is executed, the report created by the WRITE (1) statement is not printed immediately. When the INPUT statement in Program PGM-2 is executed, the report status is LOAD (the report is not yet closed). The report is closed and printed only after PGM-2 ends. The output created by PGM-2 is written to the same report as the output of PGM-1.

ET/BT Logic

The NATSPOOL nucleus attempts to issue an Adabas ET command for a given report only when the close request is executed. The close request is executed when Natural returns (internally) to command mode or when a CLOSE PRINTER statement is executed. At that time it is checked whether Natural is in ET status, that is, if a user END OF TRANSACTION statement is pending. The NATSPOOL nucleus issues an ET command only if Natural is in ET status (if no user ET is pending). This ensures that reports are stored completely on the spool file and that no interference with user transaction logic occurs.

Special attention has to be paid to the order of the END OF TRANSACTION and CLOSE PRINTER (*rep*) statements, as shown in Examples 6 to 8 below.

When creating long reports on an Adabas spool file, the transaction time limit for ET logic users (ADARUN TT parameter) must be appropriately defined. When the time limit is exceeded, the report is backed out from the spool file.

The hold queue size (ADARUN NH parameter) must be large enough to prevent Response Code 145 (HOLD QUEUE OVERFLOW) during creation of a report.

The data protection area (ADARUN LP parameter) must be large enough to prevent Response Code 9.

Example 1:

```
READ (1) PERSONNEL BY NAME
  UPDATE ...
    WRITE (1) 'string'
  END OF TRANSACTION
END
```

The PERSONNEL file is updated, and the report is printed. An ET is issued by Natural (not by NATSPOOL).

Example 2:

```
READ (1) PERSONNEL BY NAME
  UPDATE ...
    WRITE (1) 'string-1'
  END OF TRANSACTION
    WRITE (1) 'string-2'
END
```

The PERSONNEL file is updated, and the report is printed.

The END OF TRANSACTION statement forces Natural to issue an ET. Once this ET is executed, the PERSONNEL file is updated, the output *string-1* is stored on the spool file, and the report is in status LOAD. Since Natural is now in ET status, NATSPOOL issues another ET to store the output *string-2*. The report status is set to TOBE.

If an interruption occurs between the execution of the END OF TRANSACTION and the subsequent ET of NATSPOOL, the output *string-2* will be backed out from the spool file and the report will remain in status LOAD. If the program above had no END OF TRANSACTION and the CLEAR key were pressed, the entire report would be backed out. This is because Natural issues a BT when the CLEAR key is pressed.

The report can be recovered (that is, status TOBE can be forced) by issuing the function code RC with Function 10.

Example 3:

```
READ (1) PERSONNEL BY NAME
  UPDATE ...
  WRITE (1) 'string'
END
```

The PERSONNEL file and the report are in HOLD status. If the user presses the CLEAR key or terminates the Natural session, the report is backed out from the spool file.

No ET is issued, and the report can neither be canceled nor recovered by another user.

Example 4:

```
READ (1) PERSONNEL BY NAME
  UPDATE ...
  WRITE (1) 'string'
BACKOUT TRANSACTION
END
```

The update to the PERSONNEL file *and* the report are backed out. No ET is issued.

Example 5:

```
READ (1) PERSONNEL BY NAME
  WRITE (1) 'string'
END
```

The report is printed, and an ET is issued by NATSPOOL (not by Natural).

Example 6:

```
READ (1) PERSONNEL BY NAME
  UPDATE ...
    WRITE (1) 'string'
  END OF TRANSACTION
  CLOSE PRINTER (1)
END
```

The PERSONNEL file is updated, and the report is printed as soon as the CLOSE PRINTER statement is executed. An ET is issued by Natural (not by NATSPOOL).

Example 7:

```
READ (1) PERSONNEL BY NAME
  UPDATE ...
    WRITE (1) 'string'
  CLOSE PRINTER (1)
  END OF TRANSACTION  /* issued too late
END
```

During close processing (forced by the CLOSE PRINTER statement), the PERSONNEL file is still in hold. Since the start of printing is triggered during close processing, the report is not printed, but remains on the spool file in status TOLA (END OF TRANSACTION issued too late). An ET is issued by Natural (not by NATSPOOL).

Example 8:

```
READ (1) PERSONNEL BY NAME
  UPDATE ...
    WRITE (1) 'string'
  CLOSE PRINTER (1)
END
```

During close processing (forced by the CLOSE PRINTER statement), the PERSONNEL file is still on hold. Since the start of printing is triggered during close processing, the report is not printed, but remains on the spool file in status NOCL (not closed, END OF TRANSACTION missing).

No ET is issued, and the report can neither be canceled nor recovered by another user.

Recovering after Abnormal Ends

If the TP monitor terminates abnormally while NATSPOOL is printing, the report is not lost. The report remains on the spool file, with status ONPR.

After the TP monitor has been restarted, the report can be recovered by issuing the function code RC with Function 10. This forces the report status TOBE. Thereafter, the printer can be restarted to print the report.

Batch Utilities NSPOBAT, SPPBATPR and SPPPRINT

These utility modules are provided to read the spool file and print reports in batch mode. The modules are cataloged as SPPBATPR, NSPOBAT, and SPPPRINT in the Natural system library SYSPOOL.

Below is information on:

- NSPOBAT
- SPPBATPR
- SPPPRINT
- Examples - Form A and Destination ROOM*
- Deleting Reports without Printout

NSPOBAT

Reports printed with NSPOBAT are selected by their Destination/Form identification, similar to Function 10 (Reports/Queues).

All reports selected with Disposition D, K or H are printed as many times as requested when creating the report. Reports with other Dispositions are not printed.

To delete all reports with Disposition D or H from the spool file after the print job has terminated, specify PURGE as the last parameter. Reports with other Dispositions are not deleted.

SPPBATPR

Reports printed with SPPBATPR are selected by their Destination/Form identification, similar to Function 10 (Reports/Queues).

All reports selected with Disposition D or K are printed as many times as requested when creating the report.

To delete all reports with Disposition D from the spool file after the print job has terminated, specify PURGE as the last parameter.

SPPPRINT

Reports printed with SPPPRINT are selected by their Destination/Form, Disposition, user ID and by the number of days (age in days) or the creation date.

Number of days determines that all reports are printed that exceed the number of days specified as storage limit for the spool file.

The creation date determines that all reports with a creation date earlier than the date defined are printed.

All selected reports are printed as many times as requested when creating the report.

To delete all reports from the spool file after the print job has terminated, specify PURGE as the last parameter.

Examples - Form A and Destination ROOM*

In the following examples, all reports from the spool file with Form A and a destination that begins with ROOM are printed.

Example - Batch Execution under z/OS:

```
//SPPBATPR  JOB SPPBATPR,CLASS=G,MSGCLASS=X
//LIST      EXEC PGM=NATBATCH,PARM='IM=F,FSPOOL=(,XXX)'
//STEPLIB   DD   DSN=NATURAL.V41.LOAD,DISP=SHR
//          DD   DSN=ADABAS.V61.LOAD,DISP=SHR
//DDCARD    DD   DSN=NATURAL.V41.SOURCE(ADAPARM),DISP=SHR
//CMPRINT   DD   SYSOUT=X
//CMPRT01   DD   SYSOUT=X
//CMSYNIN   DD   *
LOGON SYSPOOL
SPPBATPR DESTINATION=ROOM*,FORM=A
FIN
/*
```

Example - Batch Execution under z/VSE:

```
// JOB SPPBATPR
// OPTION PARTDUMP
// ASSGN SYS010,SYSLST
// ASSGN SYS000,SYSRDR
// EXEC NATBATCH,PARM='SYSRDR'
IM=F,FSPOOL=(XXX,XXX)
/*
ADARUN SVC=XXX,DA=XXX,DEVICE=XXX,MODE=MULTI
/*
LOGON SYSPOOL
SPPBATPR DESTINATION=ROOM*,FORM=A
FIN
/*
/&
```

Example - Batch Execution under BS2000/OSD:

Note:
Forms mode must be IM=F.

```
/.SPPBATPR LOGON
/SYSFILE SYSOUT=LST.SPPBATPR
/EXEC NATB41
LOGON SYSPOOL
SPPBATPR DESTINATION=ROOM*,FORM=A
FIN
/LOGOFF
```

Deleting without Printout

If you want to delete reports by using NSPOBAT, SPPBATPR or SPPPRINT without printing them, in z/OS, instead of allocating CMPRINT to SYSOUT, you can allocate CMPRINT to DUMMY. In z/VSE, provide a dummy assignment for SYSLST (//ASSGN SYSLST,IGN) .

Example 1:

```
LOGON SYSPOOL
SPPBATPR DESTINATION=xxxxxxxx,FORM=y,PURGE=PURGE
FIN
```

All reports with destination *xxxxxxxx* and form *yare* deleted from the spool file after they have been printed.

Example 2:

All reports from the spool file are printed, and deleted after printing.

```
LOGON SYSPOOL
SPPBATPR DESTINATION=*,FORM=*,PURGE=PURGE
FIN
```

Special User Exits

- USPINIT
- USPEXIT
- USPSER01

USPINIT

After starting the MENU program in the SYSPOOL library, this subprogram is invoked. You can use this subprogram to define your own settings, authorizations, etc. You must not modify the settings for the message line (%M) and PF-key line (%Y).

USPEXIT

After the SYSPOOL application is terminated, this subprogram is invoked. You can use this subprogram to control your environment.

USPSER01

This subprogram is used by the spool server during a Natural session. It receives control before a block is sent to the printer. The delivered source contains all parameter information. When you modify data in this subprogram, the modified data are sent to the printer unchecked.

If your printer requires a different user exit, with the parameter information supplied with USPSER01 you can write a user exit subprogram that meets your requirements and catalog it in the Natural system library SYSPRINT. To assign the user exit to the printer, use Function 31.4 and enter the name of the user exit subprogram in the field Server Exit. Your user exit will then take over control before a block is sent to the printer.

For further details, see Function 31.4 (Printer) in the section Objects - Function 31.

Load and Unload Programs SPPULDUS and SPPLODUS

The Natural administrator can use the program SPPULDUS to unload objects (user profiles, logical printers, physical printers, etc.) from a spool file into Work File 3. SPPULDUS is supplied in the library SYSPOOL.

The following functions codes are available for SPPULDUS:

Function Code	Description
1	Unload user profile.
2	Unload logical printer.
3	Unload allocation table.
4	Unload physical printer.
5	Unload header pages.
6	Unload application.
7	Unload cluster.
8	Unload NTTC table.
9	Unload calendar.
A	Unload message headers (BS2000/OSD only).
*	Unload all items.

To load objects from Work File 3 into a spool file, the administrator can use the program SPPLODUS. SPPLODUS is supplied in the library SYSPOOL.

In batch mode, SPPLODUS can also be used to load objects into a spool file from a user-created work file assigned to CMWKF03. To modify multiple objects in batch, first unload objects into this work file by using SPPULDUS, then modify objects by using any edit/change tool, and finally reload the objects by using SPPLODUS. See the Natural Advanced Facilities online help for information on the layout of the unloaded objects (on the help menu, select Function 99 and then Function 1).

Note:
When executing SPPLODUS, none of the above listed function codes need to be specified.

Example - SPPULDUS with Function Code * under z/OS:

```
//SPPULDUS  JOB  CLASS=K,MSGCLASS=X
//SPPULDUS  EXEC PGM=NAT41OBT,REGION=2000K,PARM='FSPOOL=(XXX,XXX),IM=D'
//STEPLIB   DD   DSN=NATURAL.V41.LOAD,DISP=SHR
//          DD   DSN=ADABAS.V61.LOAD,DISP=SHR
//DDCARD    DD   DSN=NATURAL.V41.SOURCE(ADAPARM),DISP=SHR
//CMPRINT   DD   SYSOUT=X
//CMWKF03   DD   DSN=NAF41.UNLOAD,DISP=SHR
//CMSYNIN   DD   *
LOGON SYSPOOL
SPPULDUS
*
.
FIN
```

Example - SPPULDUS with Function Code * under z/VSE:

```
// JOB SPPULDUS
// OPTION LOG
// ASSGN SYSLST,00E
// EXEC PROG=ADAV61LB
// EXEC PROG=ALL41LB
// ASSGN SYS000,READER
// EXEC NATBATCH,SIZE=NATBATCH,PARM='SYSRDR'
FSPOOL=(XXX,XXX),IM=D
/*
ADARUN DA=XXX,SVC=XXX,TNAE=XXX,TT=XXX
/*
LOGON SYSPOOL
SPPULDUS
*
.
FIN
/*
/&
```

The sample SPPULDUS execution job leads to the following output:

```
NEXT LOGON SYSPOOL
LOGON ACCEPTED TO LIBRARY SYSPOOL
NEXT SPPULDUS

DATA *
UNLOADED USER PROFILES            :              17
UNLOADED LOGICAL PRINTERS         :              33
UNLOADED ALLOCATIONS              :             101
UNLOADED PHYSICAL PRINTERS        :              48
UNLOADED EJECT CONTROLS           :              51
UNLOADED MESSAGE HEADERS          :               2
```

```
DATA .
NEXT FIN
NAT9995 NATURAL SESSION TERMINATED NORMALLY
```

Example - SPPULDUS with Function Code * under BS2000/OSD:

```
/.ULDUS LOGON
/REMARK ******************************
/REMARK *** Unload Spool File       ***
/REMARK ******************************
/SYSFILE SYSLST=LI.LST.ULDUS
/SYSFILE SYSOUT=LI.OUT.ULDUS
/FILE DA,UNLOAD,LINK=W03
/SETSW ON=2
/SYSFILE SYSIPT=(SYSCMD)
/SYSFILE SYSDTA=(SYSCMD)
/EXEC NAT41B
AUTO=ON,ETID=’ ’
/EOF
LOGON SYSPOOL
SPPULDUS
*
.
FIN
/SYSFILE SYSLST=(PRIMARY)
/SYSFILE SYSOUT=(PRIMARY)
/LOGOFF NOSPOOL
```

Example - SPPLODUS under z/OS, z/VSE and BS2000/OSD:

```
.
.
.
LOGON SYSPOOL
SPPLODUS
FIN
.
.
.
```