1 Preface

This chapter provides an overview of the functionality of the software product SDF and contains notes on how to use the manual.

1.1 Brief product description

The software product SDF (**S**ystem **D**ialog **F**acility) supports the input of commands and program statements in interactive and batch mode as well as from procedures.

As can be seen from the product name “System Dialog Facility”, SDF supports the user primarily in the context of interactive input: for entry from the data display terminal, the user can choose any of three levels of guided dialog and two levels of unguided dialog.

The user may shift temporarily from unguided to guided dialog. If minimum or medium dialog guidance is used, temporary changeover to the next higher guidance level is possible.

Input from a procedure file and in batch mode is the same as in unguided dialog.

With inputs in batch mode, from a procedure file or in unguided dialog, an expression may be entered in place of a command or statement section. SDF replaces this entry with the contents of the specified job variable, the value of an S variable expression or a procedure parameter.

SDF protects sensitive entries such as passwords by blanking at the time of input and by masking them out in the log.

SDF checks entered commands and statements as well as their operand values for syntax errors. This includes a check whether, for instance, a specified operand value complies with the BS2000 naming conventions for files. If SDF detects a syntax error, it conducts a correction dialog with the user.

SDF also supports a dialog to rectify inputs which are formally correct but unacceptable in content (semantic errors). This is only possible if provision has been made for such a dialog in the command or statement implementation.

SDF saves syntactically correct inputs. The user can view the saved inputs or display a specific input again in order to reenter it unchanged or with modifications.

For interactive input, frequently used operand values which are not set as default values can be defined as task-specific defaults. Thereafter they need no longer be explicitly specified.

In conjunction with SDF-A (**S**ystem **D**ialog **F**acility - **A**dministration), SDF offers the possibility of influencing the functional scope of commands and statements by:

– disabling commands and statements
– restricting the function set, e.g. disabling operands or operand values
– modifying commands or statements, e.g. changing the default value of an optional operand
– defining user-specific commands upon whose entry a procedure is called (implemented procedure).

Modification of the command/statement set can be performed such that its effect is valid throughout the system or just for certain users.

Commands, statements, operands and operand values can be disabled either for all operating modes or only for a particular mode, e.g. interactive operation.

SDF is capable of supporting a multilingual command language.

SDF dialog guidance is performed in the language selected for the output of system messages. Prompts, help texts, SDF messages and comments in menus are displayed in this language.
The standard syntax files supplied to the customer contain these texts in German and English. Apart from that it is possible to translate the English command language keywords into another national language by renaming them in the syntax file (with the aid of SDF-A). The command CREATE-FILE, for example, could become the German ERZEUGE-DATEI.

SDF has its own BS2000 command language (commands in SDF format). This replaces the former command language (commands in ISP format), which was in use up to and including BS2000 V9.5A. Commands in ISP format are still supported by SDF for reasons of compatibility. However, SDF's full functionality, e.g. user guidance, is available only when the new command language is used. Commands in SDF format are distinguished by being easy to read and to understand. They can be entered in highly abbreviated form.
The SDF-CONV utility routine enables the user to convert procedures which still contain ISP commands into procedures with the corresponding SDF commands (see the "SDF-CONV" manual [7]).

1.2 Target group

This manual is intended for all users of the dialog interface as well as DP managers and systems support staff responsible for making decisions regarding the use of SDF. It demonstrates the possibilities which SDF offers the user for the input of commands and statements and provides an overview of all SDF functions. Use of the manual requires a basic knowledge of BS2000. Further SDF capabilities, which can be used for systems support tasks, are described in the manuals “SDF Management” [5] and “SDF-A” [4]. Users wishing to utilize SDF features such as self-defined commands or the program interface may refer to the “SDF-A” manual [4].

1.3 Summary of contents

This manual introduces the BS2000 dialog interface provided by the **S**ystem **D**ialog **F**acility (SDF) and describes how the user can employ SDF to enter commands and statements. The manual comprises nine chapters, which are arranged as follows:

– Chapter 1 summarizes the range of SDF functions and gives hints on how to use the manual.
– Chapter 2 provides a brief introduction to working in the SDF environment and gives brief examples of possible input.
– Chapter 3 presents the BS2000 command language developed for SDF.
– Chapter 4 contains an overview of rules and regulations to be observed when entering commands or statements.
– Chapter 5 shows, with the aid of a sample session, how commands are entered with and without prompting (referred to in SDF as guided and unguided dialog respectively).
– Chapter 6 describes the syntax files accessed by SDF to process an input. This is followed by information on the functionality of software products supporting the generation and modification of syntax files.
– Chapter 7 contains the SDF-specific commands, standard statements and an overview of the program interfaces available to the user.
– Chapter 8 describes the output of the SHOW commands in structured S variables.
– Chapter 9 provides notes on the use of OMNIS and CHECKPOINT/RESTART in conjunction with SDF.

All literature references in the text are given in abbreviated form. Complete titles of every publication referred to can be found under “Related publications”.

1.4 Version compatibility

Description status

SDF V4.5A can be used with BS2000/OSD-BC V3.0 and higher. The present manual describes the use of SDF V4.5A in a system running BS2000/OSD-BC V5.0.
In particular, the operation examples and sample sessions have been generated under BS2000/OSD-BC V2.0.

LOGOFF procedure

LOGOFF procedures that start automatically can be defined to terminate a task. The definition can be applied to a single user or to the entire system. Systems support can define one call and one include procedure in the SDF parameter file as the system LOGOFF procedure (MODIFY-SDF-PARAMETERS command). The user can be provided with a user-specific call and/or include procedure via the standard file name just like for the LOGON procedure.

Command return codes

The command return code of the last command executed can be queried with the SHOW-RETURNCODE command.

The command return codes of the RESET-/SHOW-INPUT-DEFAULTS, RESTORE-SDF-INPUT and SHOW-INPUT-HISTORY commands have been changed and/or extended.

Syntax information

Aliases are also displayed now in the output of the SHOW-CMD command.

The functionality of the SHOW-CMD command is now also provided at the program level through the new standard statement SHOW-STMT.

Examples

The examples were created under BS2000/OSD V5.0. The Styleguide mode (FUNCTION-KEYS=*STYLE-GUIDE-MODE) was set in the menu-guided examples (especially in chapters 2 and 5) for the function key assignments. Users working in the old mode (because the terminal does not support the Styleguide mode, for example) should note that sometimes the function key assignments are different and there are generally fewer function keys available in the old mode. The current functions available (and the corresponding function keys) are always displayed under the NEXT line, however.

Overview of changes to the command interface:

Command	Change
RESET-INPUT-DEFAULTS	– Command return code changed/extended
RESTORE-SDF-INPUT	– Command return code changed/extended
SHOW-CMD	– CMD-NAME= also displays aliases
SHOW-INPUT-DEFAULTS	– Command return code changed/extended
SHOW-RETURNCODE	– New command; displays the command return code of the last command

README file

Information on any functional changes and additions to the current product version described in this manual can be found in the product-specific README file.
You will find the README file on your BS2000 computer under the file name SYSRME.*product.version.language* and for SDF V4.5 under SYSRME.SDF.045.E.
The user ID under which the README file is cataloged can be obtained from your systems support staff. The full path name is also output using the following command:

```
   /SHOW-INSTALLATION-PATH INSTALLATION-UNIT=SDF,LOGICAL-IDENTIFIER=SYSRME.E
```

You can view the README file using the `/SHOW-FILE` command or an editor, and print it out on a standard printer using the following command:

```
PRINT-DOCUMENT <path name>, LINE-SPACING=*BY-EBCDIC-CONTROL
```

Terms used

The present manual uses the terms “S procedure”, “non-S procedure”, “dialog block”, “S variable”, “predefined function” (“built-in function”), “S label” and “non-S label”:

– An S procedure is a procedure whose procedure/parameter declarations are formulated according to SDF-P rules and whose execution can be controlled with SDF-P language elements. In the event of errors, SDF-P error handling is initiated.

– A non-S procedure is a procedure which was created according to conventional rules (e.g. first command BEGIN-PROCEDURE). In the event of an error, the spin-off mechanism is triggered.

– A dialog block is a special form of interactive input in SDF-P.

– S variables are part of the variables concept of SDF-P.
Parameters of an S procedure are S variables.

- Built-in functions likewise belong to the SDF-P vaiables concept.
  They allow the user to process the contents of S variables or to request information on the current system environment.
- Non-S labels are the conventional labels, which can be prefixed to the command name.
- S labels are labels in S procedures which can be analyzed only by SDF-P.

2 Brief introduction to working with SDF

SDF is a command language which allows convenient interactive input of commands and program statements. Dialog guidance makes input easier, informs the user of the functions of individual operands as required, and permits a correction dialog if an input error is detected. Extensive knowledge of SDF command input is not required, as the brief examples in this section are sufficiently informative. Depending on your system configuration, your output screens may differ from those in the examples (e.g. in the range of commands offered).
SDF offers the following special features:

Mnemonics
The command language itself already simplifies the use of commands: the names of commands, operands and constant operand values have been chosen such that they reflect the function or use. In addition, analogous facts are described by means of identical keywords throughout the command set.

Abbreviation mechanism
Long inputs are not problematic. In interactive mode, SDF offers many ways of abbreviating your inputs. Most operands, for example, have a preset operand value, the default value. You need only specify these operands if you want to change their values. If any of the defaults do not correspond to the values you use most frequently, you can define your own default values for your task.
You can also abbreviate names: within names or partial names you can leave out from right to left as many characters as you wish, provided the name remains recognizable in its context. Bear in mind that a command name is unique among all command names, and an operand name is unique among all operand names of the command on the appropriate structural level.

Line or menu mode
You can enter your input either directly in line mode (unguided dialog, page 51) or by switching to a menu-driven input mode (guided dialog, page 57). Guided dialog supports three levels of guidance, which differ in the range of menu options they offer. You can change your input mode at any time.

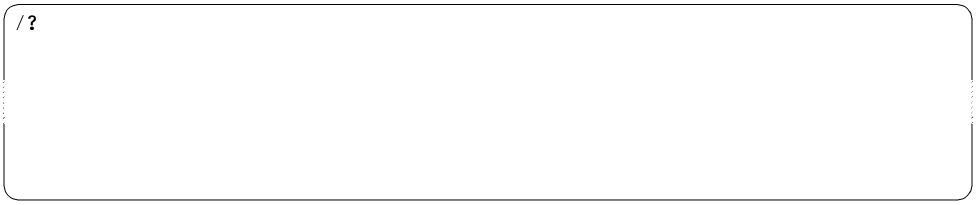
Error correction dialog
SDF checks your input for syntax errors before the command is executed. If it detects an error, it switches automatically to a correction dialog, depending on the level of guidance set.

Reconstruction of commands
SDF can save your inputs to a buffer. This enables you to retrieve any such saved input and use it again (see page 104).

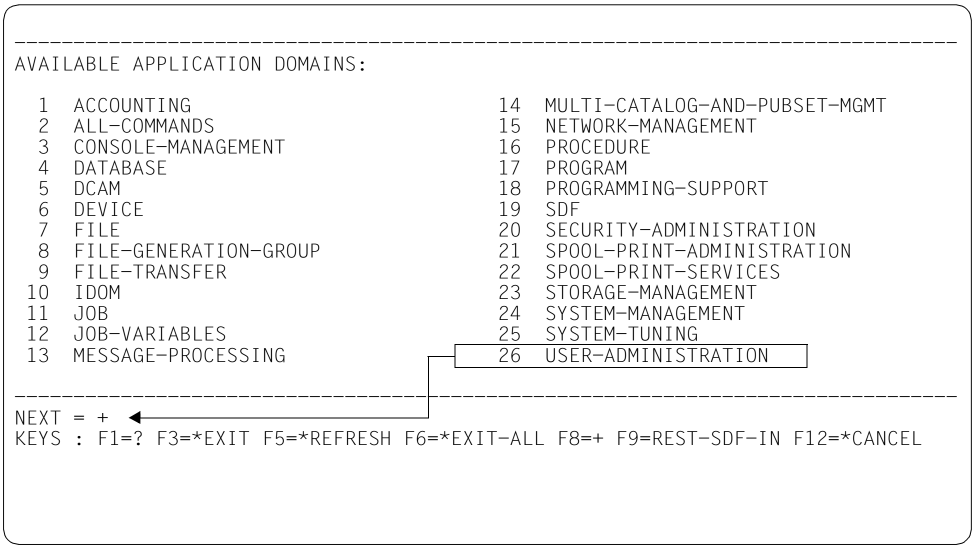
SDF information system (help)
You can call SDF help by entering a question mark. Depending on the position of the question mark within the input and the level of guidance set, SDF provides various forms of support during command input. These include a selection menu for certain commands, an operand form for a specific command, or information on a specific operand.
SDF offers the simplest form of support in unguided dialog:
By entering a question mark you can switch to guided dialog for the input of one command only. When you have entered the command you will be returned to the original input mode - unguided dialog. This type of input is called temporary guided dialog (see page 76).

Outputting application domains

If you enter a question mark and press the transmission key (DUE), SDF provides a selection of application domains (see the application domain menu on page 70). For the following example, select the application domain USER-ADMINISTRATION by entering in the NEXT input field (see NEXT line, page 63) the number of the application domain (26 here) and sending off the menu with DUE.



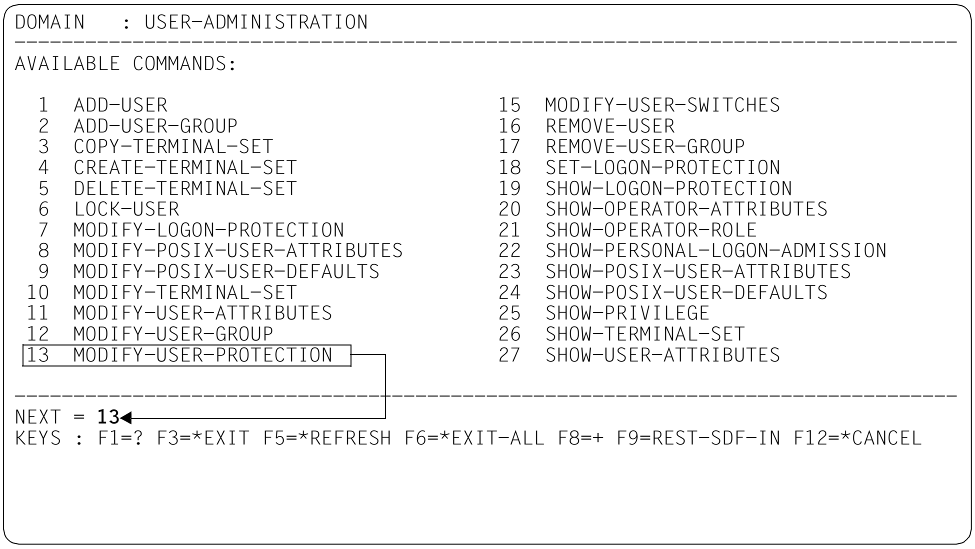
```
/?
```



```
--------------------------------------------------------------------------------
AVAILABLE APPLICATION DOMAINS:

  1  ACCOUNTING                            14  MULTI-CATALOG-AND-PUBSET-MGMT
  2  ALL-COMMANDS                          15  NETWORK-MANAGEMENT
  3  CONSOLE-MANAGEMENT                    16  PROCEDURE
  4  DATABASE                              17  PROGRAM
  5  DCAM                                  18  PROGRAMMING-SUPPORT
  6  DEVICE                                19  SDF
  7  FILE                                  20  SECURITY-ADMINISTRATION
  8  FILE-GENERATION-GROUP                 21  SPOOL-PRINT-ADMINISTRATION
  9  FILE-TRANSFER                         22  SPOOL-PRINT-SERVICES
 10  IDOM                                  23  STORAGE-MANAGEMENT
 11  JOB                                   24  SYSTEM-MANAGEMENT
 12  JOB-VARIABLES                         25  SYSTEM-TUNING
 13  MESSAGE-PROCESSING                    26  USER-ADMINISTRATION

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F8=+ F9=REST-SDF-IN F12=*CANCEL
```

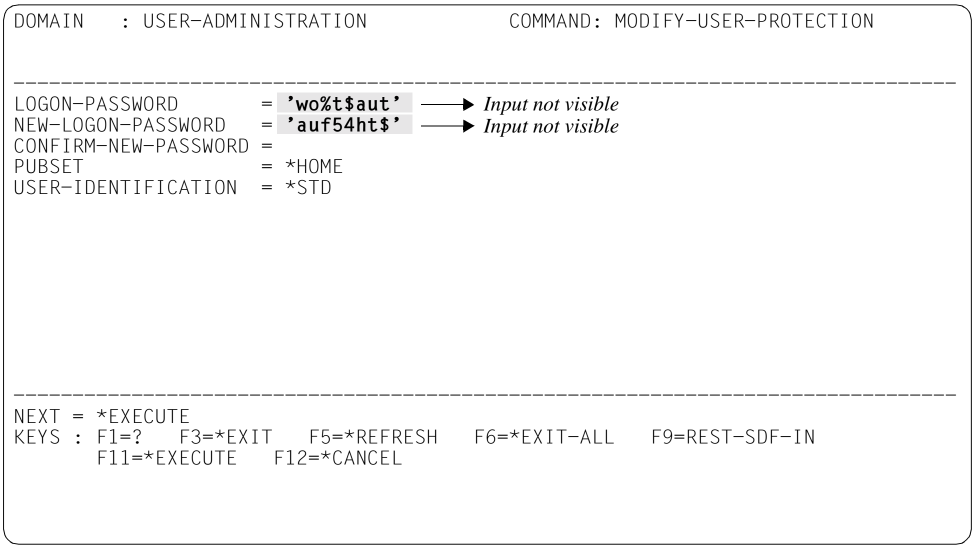
Changing your password

Supposing you want to change your password: you have already selected the application domain USER-ADMINISTRATION. A selection menu appears containing all the commands of this domain (see the command menu on page 71). Select the MODIFY-USER-PROTECTION command by entering the number 13 in the NEXT input field.



```
DOMAIN   : USER-ADMINISTRATION
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  ADD-USER                                 15  MODIFY-USER-SWITCHES
  2  ADD-USER-GROUP                           16  REMOVE-USER
  3  COPY-TERMINAL-SET                        17  REMOVE-USER-GROUP
  4  CREATE-TERMINAL-SET                      18  SET-LOGON-PROTECTION
  5  DELETE-TERMINAL-SET                      19  SHOW-LOGON-PROTECTION
  6  LOCK-USER                                20  SHOW-OPERATOR-ATTRIBUTES
  7  MODIFY-LOGON-PROTECTION                  21  SHOW-OPERATOR-ROLE
  8  MODIFY-POSIX-USER-ATTRIBUTES             22  SHOW-PERSONAL-LOGON-ADMISSION
  9  MODIFY-POSIX-USER-DEFAULTS               23  SHOW-POSIX-USER-ATTRIBUTES
 10  MODIFY-TERMINAL-SET                      24  SHOW-POSIX-USER-DEFAULTS
 11  MODIFY-USER-ATTRIBUTES                   25  SHOW-PRIVILEGE
 12  MODIFY-USER-GROUP                        26  SHOW-TERMINAL-SET
 13  MODIFY-USER-PROTECTION                   27  SHOW-USER-ATTRIBUTES

--------------------------------------------------------------------------------
NEXT = 13
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F8=+ F9=REST-SDF-IN F12=*CANCEL
```

In the operand form, enter your current password and the new password you want in the protected input fields LOGON-PASSWORD and NEW-LOGON-PASSWORD respectively, and send the form off with [DUE].

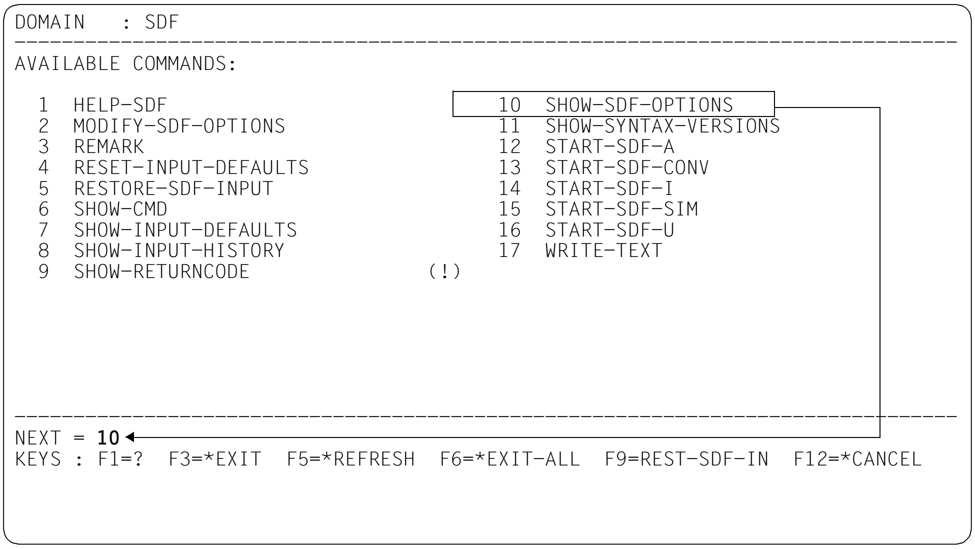


```
DOMAIN   : USER-ADMINISTRATION            COMMAND: MODIFY-USER-PROTECTION


--------------------------------------------------------------------------------
LOGON-PASSWORD        = 'wo%t$aut'   ——> Input not visible
NEW-LOGON-PASSWORD    = 'auf54ht$'   ——> Input not visible
CONFIRM-NEW-PASSWORD =
PUBSET                = *HOME
USER-IDENTIFICATION   = *STD










--------------------------------------------------------------------------------
NEXT = *EXECUTE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

The temporarily guided dialog is terminated and the slash reappears ready for the next command input.

Querying and modifying task-specific SDF options

You can find out the current SDF options for your task, such as the input mode, scope of logging, or command history. To do this, enter a question mark followed by DUE to open the selection menu with the application domains. The command you require belongs to domain number 19 (SDF). There you select the SHOW-SDF-OPTIONS command by entering the number 10 in the NEXT input field and sending off the menu with DUE.



```
DOMAIN   : SDF
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  HELP-SDF                               10  SHOW-SDF-OPTIONS
  2  MODIFY-SDF-OPTIONS                     11  SHOW-SYNTAX-VERSIONS
  3  REMARK                                 12  START-SDF-A
  4  RESET-INPUT-DEFAULTS                   13  START-SDF-CONV
  5  RESTORE-SDF-INPUT                      14  START-SDF-I
  6  SHOW-CMD                               15  START-SDF-SIM
  7  SHOW-INPUT-DEFAULTS                    16  START-SDF-U
  8  SHOW-INPUT-HISTORY                     17  WRITE-TEXT
  9  SHOW-RETURNCODE                   (!)





--------------------------------------------------------------------------------
NEXT = 10
KEYS : F1=?  F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F9=REST-SDF-IN  F12=*CANCEL
```

The operand form for the command appears. In the INFORMATION input field, which contains the default value *ALL, enter *user and send off the form with DUE.

```
DOMAIN   : SDF                            COMMAND: SHOW-SDF-OPTIONS


--------------------------------------------------------------------------------
INFORMATION          = *user














--------------------------------------------------------------------------------
NEXT = *EXECUTE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

SDF outputs the SDF options of your task. INFORMATION=*USER gives you only SDF options which you can modify yourself using the MODIFY-SDF-OPTIONS command.

```
%  USER      : *NONE
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *OLD-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
/
```

The INPUT-HISTORY output field with the value *ON indicates that the command history is activated. The INPUT-BUFFER-SIZE output field indicates that SDF saves only the last 20 inputs.
In one of the following examples, SDF accesses the command history. If INPUT-HISTORY has the value *OFF on your screen, you can activate the command history by, for example, making use of the abbreviation options and entering directly “mod-sdf-opt input=*on”. Alternatively, you can request the operand form in temporarily guided dialog and change the value for INPUT-HISTORY accordingly.

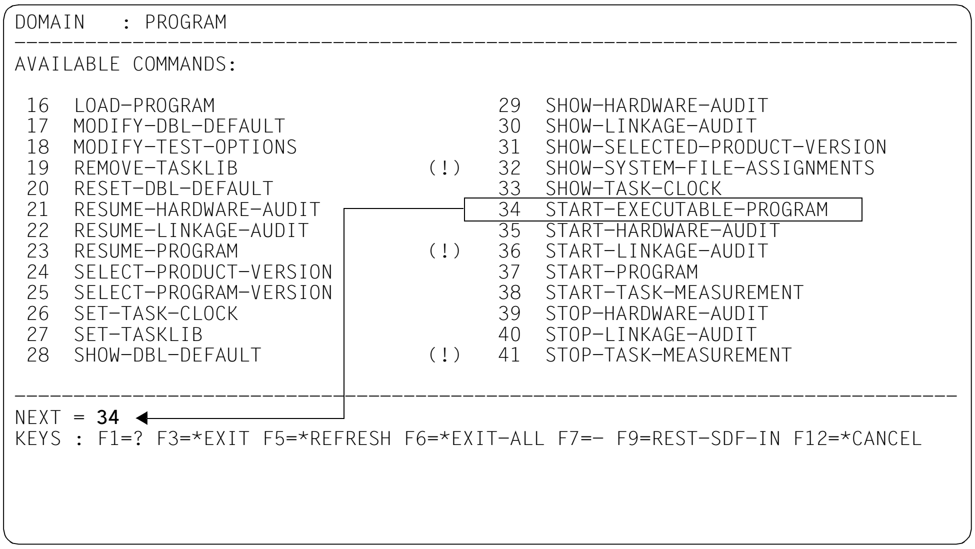
Creating a logon procedure

Write yourself a short procedure which, after logon, automatically displays all files which have not been accessed for at least 20 days. You must save the commands which comprise the procedure to a file with the default name SYS.SDF.LOGON.USERPROC under your user ID. The default name causes SDF to call the procedure automatically after logon (see the description of logon procedures on page 81). To create the procedure file, call the (editor) program EDT[1]. “?” followed by DUE opens the selection menu containing the application domains. “17” followed by DUE displays the commands of the PROGRAM domain. DUE takes you one screen further (the NEXT line is already preset to “+“). To start the program, enter “34” in the subsequent screen to issue the START-EXECUTABLE-PROGRAM[2] command and send off the menu with DUE.

```
DOMAIN    : PROGRAM
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  ASSIGN-SYSDTA                          15  LOAD-EXECUTABLE-PROGRAM
  2  ASSIGN-SYSIPT                          16  LOAD-PROGRAM
  3  ASSIGN-SYSLST                          17  MODIFY-DBL-DEFAULT
  4  ASSIGN-SYSOPT                          18  MODIFY-TEST-OPTIONS
  5  ASSIGN-SYSOUT                          19  REMOVE-TASKLIB            (!)
  6  CANCEL-PROGRAM                   (!)   20  RESET-DBL-DEFAULT
  7  COPY-SYSTEM-FILE                       21  RESUME-HARDWARE-AUDIT
  8  CREATE-DUMP                      (!)   22  RESUME-LINKAGE-AUDIT
  9  DELETE-SYSTEM-FILE                     23  RESUME-PROGRAM            (!)
 10  EOF                              (!)   24  SELECT-PRODUCT-VERSION
 11  HOLD-HARDWARE-AUDIT                    25  SELECT-PROGRAM-VERSION
 12  HOLD-LINKAGE-AUDIT                     26  SET-TASK-CLOCK
 13  INFORM-OPERATOR                        27  SET-TASKLIB

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F8=+ F9=REST-SDF-IN F12=*CANCEL
```

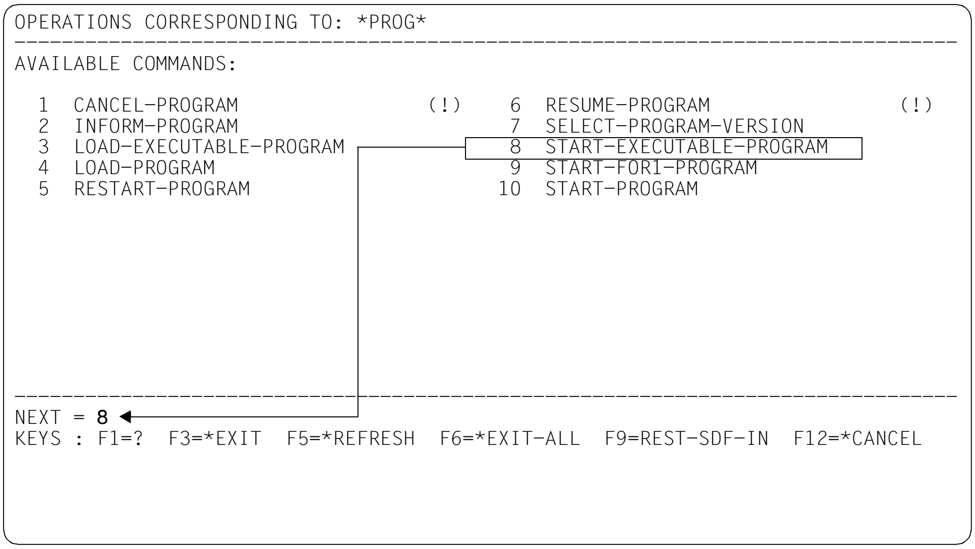
[1] The utility routine EDT can also be called with the START-EDT command.

[2] The command is available in BLSSERV V2.3 and higher and will replace the START-PROGRAM command in the long term.



```
DOMAIN    : PROGRAM
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 16  LOAD-PROGRAM                               29  SHOW-HARDWARE-AUDIT
 17  MODIFY-DBL-DEFAULT                         30  SHOW-LINKAGE-AUDIT
 18  MODIFY-TEST-OPTIONS                        31  SHOW-SELECTED-PRODUCT-VERSION
 19  REMOVE-TASKLIB                        (!)  32  SHOW-SYSTEM-FILE-ASSIGNMENTS
 20  RESET-DBL-DEFAULT                          33  SHOW-TASK-CLOCK
 21  RESUME-HARDWARE-AUDIT                      34  START-EXECUTABLE-PROGRAM
 22  RESUME-LINKAGE-AUDIT                       35  START-HARDWARE-AUDIT
 23  RESUME-PROGRAM                        (!)  36  START-LINKAGE-AUDIT
 24  SELECT-PRODUCT-VERSION                     37  START-PROGRAM
 25  SELECT-PROGRAM-VERSION                     38  START-TASK-MEASUREMENT
 26  SET-TASK-CLOCK                             39  STOP-HARDWARE-AUDIT
 27  SET-TASKLIB                                40  STOP-LINKAGE-AUDIT
 28  SHOW-DBL-DEFAULT                      (!)  41  STOP-TASK-MEASUREMENT

--------------------------------------------------------------------------------
NEXT = 34
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F7=- F9=REST-SDF-IN F12=*CANCEL
```

Alternatively, you can start, for example, with the input “*prog*?” instead of “?”. You will receive a selection menu containing all commands whose name contains the word PROGRAM (see section “Selection menu for commands and statements” on page 102). Enter “8” to select the START-EXECUTABLE-PROGRAM command.

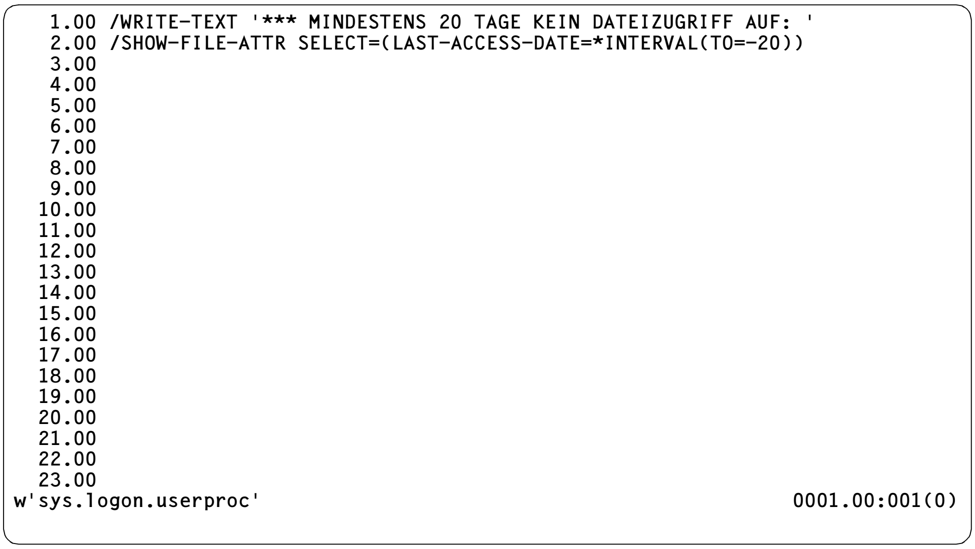


```
OPERATIONS CORRESPONDING TO: *PROG*
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  CANCEL-PROGRAM                        (!)   6  RESUME-PROGRAM                (!)
  2  INFORM-PROGRAM                              7  SELECT-PROGRAM-VERSION
  3  LOAD-EXECUTABLE-PROGRAM                     8  START-EXECUTABLE-PROGRAM
  4  LOAD-PROGRAM                                9  START-FOR1-PROGRAM
  5  RESTART-PROGRAM                            10  START-PROGRAM

--------------------------------------------------------------------------------
NEXT = 8
KEYS : F1=?  F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F9=REST-SDF-IN  F12=*CANCEL
```

In both cases you will receive the operand form of the START-EXECUTABLE-PROGRAM command. In the FROM-FILE input field, enter the program name $.EDT. This is the name of the file from which the program EDT will be loaded (in this case the file EDT in the system default ID; if the name differs from this, you can find it out from system administration).

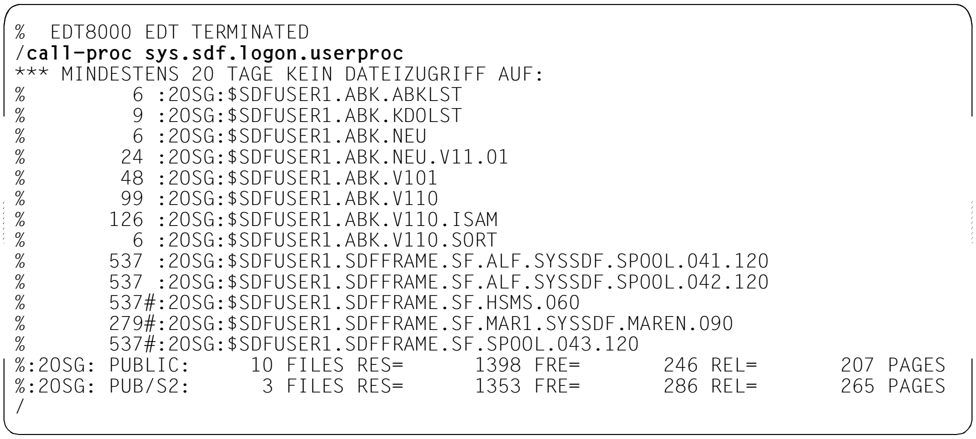
```
COMMAND  : START-EXECUTABLE-PROGRAM



--------------------------------------------------------------------------------
FROM-FILE              = $.edt
PROGRAM-PARAMETERS     = *NONE
DBL-PARAMETERS         = *STD
MONJV                  = *NONE
CPU-LIMIT              = *JOB-REST
TEST-OPTIONS           = *DBL-DEFAULT
RESIDENT-PAGES         = *PARAMETERS(MINIMUM=*STD,MAXIMUM=*STD)
VIRTUAL-PAGES          = *STD





--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Once EDT is loaded, the EDT screen mask is displayed.
In the input fields 1.00 to 2.00, enter the specified commands. The WRITE-TEXT command outputs the specified text to SYSOUT. It provides a comment on the file selection of the SHOW-FILE-ATTRIBUTES command.
To save the input as the file SYS.SDF.LOGON.USERPROC, enter the command “write'sys.sdf.logon.userproc'” in the last line of the EDT mask and send off the mask with DUE.



```
   1.00 /WRITE-TEXT '*** MINDESTENS 20 TAGE KEIN DATEIZUGRIFF AUF: '
   2.00 /SHOW-FILE-ATTR SELECT=(LAST-ACCESS-DATE=*INTERVAL(TO=-20))
   3.00
   4.00
   5.00
   6.00
   7.00
   8.00
   9.00
  10.00
  11.00
  12.00
  13.00
  14.00
  15.00
  16.00
  17.00
  18.00
  19.00
  20.00
  21.00
  22.00
  23.00
w'sys.logon.userproc'                                               0001.00:001(0)
```

The EDT mask is displayed again, confirming via message EDT0173 that the file SYS.SDF.LOGON.USERPROC has been created. If the file already exists, EDT asks beforehand whether it is to be overwritten.
To terminate EDT, enter “halt” in the last line and press $\boxed{\text{DUE}}$.
To check the procedure, call it up with “call-proc sys.sdf.logon.userproc”.

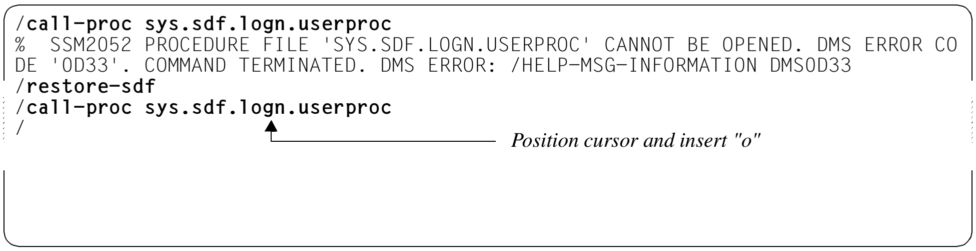


```
%  EDT8000 EDT TERMINATED
/call-proc sys.sdf.logon.userproc
*** MINDESTENS 20 TAGE KEIN DATEIZUGRIFF AUF:
%         6 :2OSG:$SDFUSER1.ABK.ABKLST
%         9 :2OSG:$SDFUSER1.ABK.KDOLST
%         6 :2OSG:$SDFUSER1.ABK.NEU
%        24 :2OSG:$SDFUSER1.ABK.NEU.V11.01
%        48 :2OSG:$SDFUSER1.ABK.V101
%        99 :2OSG:$SDFUSER1.ABK.V110
%       126 :2OSG:$SDFUSER1.ABK.V110.ISAM
%         6 :2OSG:$SDFUSER1.ABK.V110.SORT
%       537 :2OSG:$SDFUSER1.SDFFRAME.SF.ALF.SYSSDF.SPOOL.041.120
%       537 :2OSG:$SDFUSER1.SDFFRAME.SF.ALF.SYSSDF.SPOOL.042.120
%       537#:2OSG:$SDFUSER1.SDFFRAME.SF.HSMS.060
%       279#:2OSG:$SDFUSER1.SDFFRAME.SF.MAR1.SYSSDF.MAREN.090
%       537#:2OSG:$SDFUSER1.SDFFRAME.SF.SPOOL.043.120
%:2OSG: PUBLIC:     10 FILES RES=      1398 FRE=       246 REL=       207 PAGES
%:2OSG: PUB/S2:      3 FILES RES=      1353 FRE=       286 REL=       265 PAGES
/
```

This direct input makes use of the abbreviation mechanism (abbreviation of the command name CALL-PROCEDURE; specification of the procedure file as the first positional operand because NAME is the first operand in the command). Alternatively, you can use the method employed previously: Input “?” $\longrightarrow$ “16” (PROCEDURE) $\longrightarrow$ Select CALL-PROCEDURE $\longrightarrow$ Enter file name under FROM-FILE.

Reconstructing a command

Let us suppose you have made a mistake when calling the SYS.SDF.LOGON.USERPROC procedure (see previous example), e.g. you have entered an incorrect file name.
In this case, you can redisplay the errored input using “restore-sdf”, one of the possible abbreviations of the RESTORE-SDF-INPUT command.



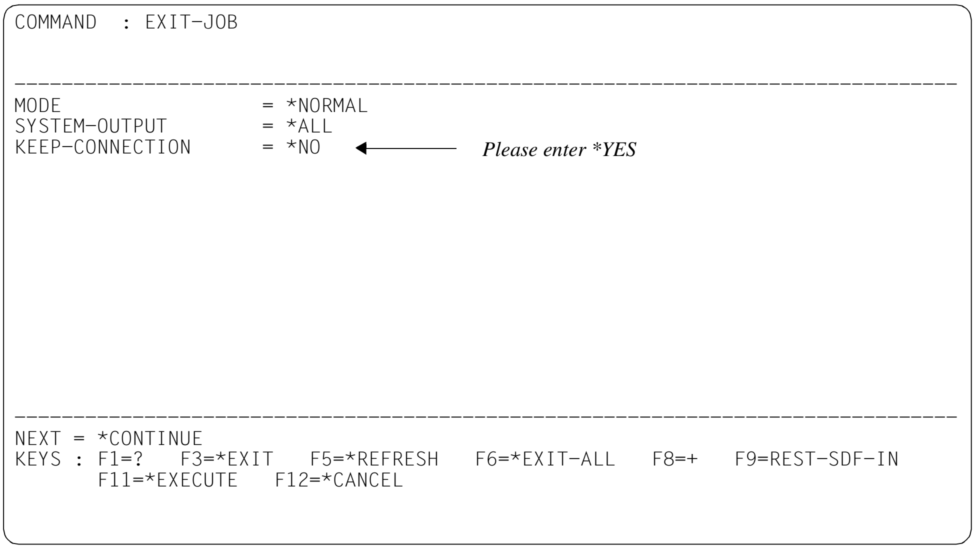
```
/call-proc sys.sdf.logn.userproc
%  SSM2052 PROCEDURE FILE 'SYS.SDF.LOGN.USERPROC' CANNOT BE OPENED. DMS ERROR CO
DE '0D33'. COMMAND TERMINATED. DMS ERROR: /HELP-MSG-INFORMATION DMS0D33
/restore-sdf
/call-proc sys.sdf.logn.userproc
/
```

Position cursor and insert "o"

Correct the file name in the command line by, for example, positioning the cursor after “sys.sdf.log”, pressing the $\boxed{\text{EFG}}$ (insert) key and inserting the missing letter “o”. Now send off this corrected command line with $\boxed{\text{DUE}}$.

Terminating an interactive task

Terminate the interactive task with the EXIT-JOB command. When you enter “exit-job?”, SDF changes to temporarily guided dialog and outputs the operand form of the command.

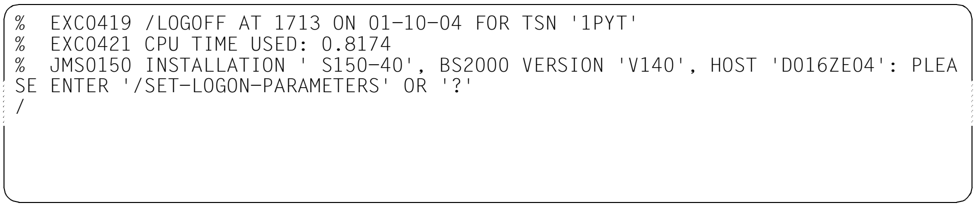


```
COMMAND  : EXIT-JOB


--------------------------------------------------------------------------------
MODE                    = *NORMAL
SYSTEM-OUTPUT           = *ALL
KEEP-CONNECTION         = *NO
















--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

You can also display the operand form by entering the following:
“?” → “11” (JOB) → Select EXIT-JOB.
As you will need to log on again in the following example, change the value in the KEEP-CONNECTION input field to *YES before you send off the menu with DUE. This ensures that the system remains connected when the task is finished and you can log on again immediately.

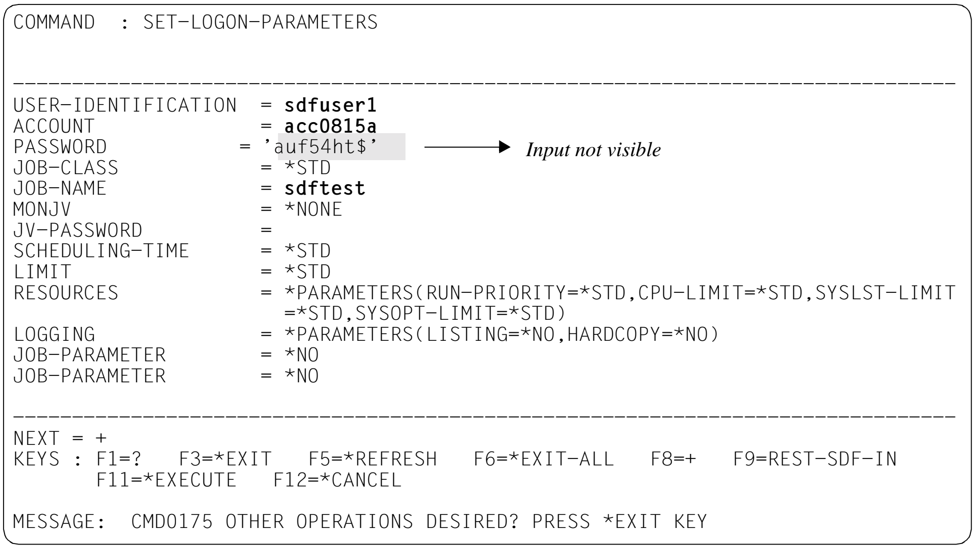
Starting an interactive task

When you have terminated your interactive task (see previous example), the system prompts you to log on again. Enter “?” and DUE.



```
%  EXC0419 /LOGOFF AT 1713 ON 01-10-04 FOR TSN '1PYT'
%  EXC0421 CPU TIME USED: 0.8174
%  JMS0150 INSTALLATION ' S150-40', BS2000 VERSION 'V140', HOST 'D016ZE04': PLEA
SE ENTER '/SET-LOGON-PARAMETERS' OR '?'
/
```

Because you must first log on to the system with SET-LOGON-PARAMETERS before entering further commands, SDF immediately displays the operand form. The input “set-logon-parameters?” has the same effect. Enter your user ID, account number and password. You can enter a job name if you wish.
In the last line SDF informs you that you can obtain an overview of all other commands with *EXIT. You can trigger an *EXIT by pressing the F2 key (see the function key assignment in the KEYS line[1]), or you can enter *EXIT in the NEXT line and then press the DUE key. With *EXIT you quit the operand form and switch to a selection menu which contains all the commands you may use before logging on. These, however, are only the commands EXIT-JOB and LOGOFF to terminate the task, and the SET-LOGON-PARAMETERS command for logging on.



```
COMMAND  : SET-LOGON-PARAMETERS


-------------------------------------------------------------------------------
USER-IDENTIFICATION  = sdfuser1
ACCOUNT              = acc0815a
PASSWORD            = 'auf54ht$'      ——————▶ Input not visible
JOB-CLASS            = *STD
JOB-NAME             = sdftest
MONJV                = *NONE
JV-PASSWORD          =
SCHEDULING-TIME      = *STD
LIMIT                = *STD
RESOURCES            = *PARAMETERS(RUN-PRIORITY=*STD,CPU-LIMIT=*STD,SYSLST-LIMIT
                       =*STD,SYSOPT-LIMIT=*STD)
LOGGING              = *PARAMETERS(LISTING=*NO,HARDCOPY=*NO)
JOB-PARAMETER        = *NO
JOB-PARAMETER        = *NO

-------------------------------------------------------------------------------
NEXT = +
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL

MESSAGE:  CMD0175 OTHER OPERATIONS DESIRED? PRESS *EXIT KEY
```

After checking the logon details, SDF starts your logon procedure.

[1] The “KEYS” field name is only displayed in the Styleguide mode. If you have set the old mode (because your terminal does not support the Styleguide mode, for example), then the possible entries and corresponding function key assignments will be displayed anyway, but there are fewer function keys available. Note that in the old mode *EXIT corresponds to the K1 key.

```
%  JMS0066 JOB 'SDFTEST' ACCEPTED ON 01-10-04 AT 17:14, TSN = 1PY8
*** MINDESTENS 20 TAGE KEIN DATEIZUGRIFF AUF:
%         6 :20SG:$SDFUSER1.ABK.ABKLST
%         9 :20SG:$SDFUSER1.ABK.KDOLST
%         6 :20SG:$SDFUSER1.ABK.NEU
%        24 :20SG:$SDFUSER1.ABK.NEU.V11.01
%        48 :20SG:$SDFUSER1.ABK.V101
%        99 :20SG:$SDFUSER1.ABK.V110
%       126 :20SG:$SDFUSER1.ABK.V110.ISAM
%         6 :20SG:$SDFUSER1.ABK.V110.SORT
%       537 :20SG:$SDFUSER1.SDFFRAME.SF.ALF.SYSSDF.SPOOL.041.120
%       537 :20SG:$SDFUSER1.SDFFRAME.SF.ALF.SYSSDF.SPOOL.042.120
%       537#:20SG:$SDFUSER1.SDFFRAME.SF.HSMS.060
%       279#:20SG:$SDFUSER1.SDFFRAME.SF.MAR1.SYSSDF.MAREN.090
%       537#:20SG:$SDFUSER1.SDFFRAME.SF.SPOOL.043.120
%:20SG: PUBLIC:     10 FILES RES=      1398 FRE=       246 REL=       207 PAGES
%:20SG: PUB/S2:      3 FILES RES=      1353 FRE=       286 REL=       265 PAGES
/
```

If you are already familiar with the SET-LOGON-PARAMETERS command, you can also log onto the system directly. To do this, enter the command with the necessary specifications and send it off with DUE. For security reasons you should enter the operand value *SECRET instead of your password. SDF will then provide a protected input field for you to enter your password.

```
/set-logon-par user=sdfuser1,account=acc0815a,password=*secret,job-name=sdftest

ENTER SECRET OPERAND (PASSWORD): 'auf54ht$'          Input not visible
/
```

Notes

The examples in this chapter could only show you some of the most important input options. We have only shown you input in unguided dialog and the switch to temporarily guided dialog.
However, SDF also allows you to work permanently in guided dialog. A sample session with all modes of guidance is described in chapter “Examples of command input in interactive mode” on page 117.
The input options, in particular abbreviation rules, guidance levels, menu control and the definition of task-specific default values, are described in chapter “Input of commands and statements” on page 51.

The functional scope (permitted commands, operands and operand values) available to you is defined in syntax files. The meanings of the syntax files are described in chapter “SDF syntax files” on page 155.

If your procedures still contain commands from the previous command language ISP, the SDF-CONV utility routine supports you in converting these to SDF commands. If you enter an ISP command followed by a question mark in interactive mode, SDF helps to find the corresponding ISP command: if there is just one corresponding SDF command, its operand form will be displayed; if exact mapping is not possible, SDF outputs a list of possible SDF commands.

3 SDF command language

A separate command language has been developed for the software product SDF: the SDF command language. As of BS2000 V10.0A, the SDF command language supersedes the old ISP command language, whose commands continue to be supported for reasons of compatibility. The full SDF functionality, however, can only be used for commands and statements with SDF syntax. ISP commands are merely subjected to a name check.

3.1 Format of the SDF command language

The syntax description of SDF commands and statements is contained in the syntax files (see chapter “SDF syntax files” on page 155ff):

– Commands begin with a slash which in dialog mode is set by the system. SDF expects command input from the logical system file SYSCMD.
– SDF statements, i.e. statements intended for programs with an SDF interface, begin with two slashes which in dialog mode are set by the system. SDF expects statement input from the logical system file SYSSTMT which is assigned the same file as system file SYSDTA.

Input data, i.e. data, parameters and statements read in by programs without the SDF interface, are not analyzed by SDF. Such programs expect input data from the logical system file SYSDTA.

Commands/statements consist of the following elements:
– command/statement name
– operand names
– possible operand values (keyword values and variable values)
– additional help texts explaining the command/statement and its operands.

Command/statement names, operand names and keyword values are contained as keywords in the syntax description. Keywords are specified as such (see section “Abbreviation of names” on page 94ff). Variable operand values are described by means of data types (for a list see table 2 on page 33ff). The data type defines from which character set and according to which rules an operand value is formed. SDF checks whether the specification for a variable operand value lies within the range of permissible values as defined by the data type.

A variable operand value can also be defined as a single value (constant value) of a data type. Instead of the data type, the syntax description then contains the relevant single value (e.g. the names of the S variable streams in the command ASSIGN-STREAM: STREAM-NAME = SYSVAR / SYSMSG / SYSINF).

Keywords usually consist of several portions (or partial names) connected by hyphens. As a rule, the respective names are taken from the ordinary English language environment and are chosen in such a way that analogous facts are described by means of identical keywords throughout the command set.
Commands always start with a verb followed by the associated object to which the activity is to be applied, e.g. MODIFY-FILE-ATTRIBUTES causes the attributes of a file to be modified.

There are also pairs of opposite activities, e.g. CREATE-XY and DELETE-XY for the creation/deletion of an object XY.

Commands are assigned to different application domains, depending on their respective uses. A command may occur in more than one domain.

Every operand has a name and at least one possible value. Operands may be hierarchically subordinate to an operand value, in which case this operand value initiates a structure which contains all the lower-ranking operands. The operands of a structure are enclosed in parentheses. Further structures may be initiated within a structure; this phenomenon is referred to as structure nesting.

Example of a list:

Format (excerpt from SHOW-FILE-ATTRIBUTES):

,**SEL**ECT = **<u>*ALL</u>** / [***BY-ATTR**IBUTES](...)

[***BY-ATTR**IBUTES](...)

| ,**FILE-STRUC**TURE = **<u>*ANY</u>** / list-poss(5): ***PAM** / ***SAM** / ***ISAM** / ***BTAM** / ***NONE**

Input:

```
/show-file-attributes select=*by-attributes(file-structure=(*sam,*isam))
```

The FILE-STRUCTURE operand defines a specific file attribute and is subordinate to the operand value (the structure) *BY-ATTRIBUTES.

Concurrent specification of several operand values in the form of a list may be permissible. Operand values permitted as list elements are enumerated following “list-poss(n):” (see the SDF syntax representation, page 32), where “n” indicates the maximum number of list elements allowed.

Example of a list:

Format (excerpt from SHOW-FILE-ATTRIBUTES):

FILE-STRUCTURE = ***ANY** / list-poss(5): ***PAM** / ***SAM** / ***ISAM** / ***BTAM** / ***NONE**

Input:

```
/show-file-attributes select=*by-attributes(file-structure=(*sam,*isam))
```

The FILE-STRUCTURE operand defines the file attributes SAM and ISAM.

Most operands are preset with a default value. This operand value is used if no explicit specification is made (see section “Default values” on page 96).

Conventions for input

The following should be noted when entering commands and statements:

- SDF interprets inputs in accordance with the standard code table **EBCDIC.DF.03** (e.g. uppercase/lowercase conversion).
  XHCS[1], if available, permits the use of extended character sets (CCS[2]). The coding of dollar sign ($), number sign (#), commercial at (@), exclamation mark (!), circumflex (^) parentheses and angle brackets, greater than and less than signs, asterisk (*), slash (/), colon, comma, semicolon, apostrophe and quotation mark (") in an extended character set must comply with the coding in the standard code table.
  SDF interprets additional characters of an extended character set only within the data types <c-string> and <text>, i.e. uppercase/lowercase conversion makes use of an XHCS-supplied code table for that extended character set.
  SDF rejects all additional characters used within any of the other data types and reports a syntax error.

  The following distinctions are made when using an extended character set:

  – Input at the data display terminal:
    The CCS from the user entry is used for command input, provided 8-bit mode has been activated with the MODIFY-TERMINAL-OPTIONS command.
    For statement input, the CCS specified in the relevant macro call (CMDRST or RDSTMT, CMDTST or TRSTMT, or CMDCST or CORSTMT) is used. If no CCS has been specified, the same applies as for command input.

  – Input from a file or PLAM library member:
    For command input, the CCS of the input file is used, or the CCS of the PLAM library member, provided SYSDTA has been assigned with the ASSIGN-SYSDTA command.
    For statement input, the CCS specified in the relevant macro call must be identical with the CCS for SYSDTA inputs. The statement will be rejected if the two CCS are not identical.
    During a procedure interrupt, the CCS selected for input at the data display terminal is used.

  – Input from an S variable:
    If SYSDTA is assigned to a compound S variable, the CCS selected for input at the data display terminal is used in dialog mode, while the CCS **EDF03IRV** is used in batch mode.

  – The macro calls CMD and TRCMD do **not** support extended character sets.

[1] The subsystem XHCS (Extended Host Code Support) allows 8-bit code processing.

[2] Coded Character Set

- The first character of a command input is the slash (/); a statement is preceded by two slashes (//).
  In the event of input at the display terminal, the slash(es) appear(s) automatically as a system prompt. In the event of input from a procedure file, the slash(es) must be included in the input records.
- A label may be placed between slash and command name to identify the command line as a branch destination within procedures. At least one blank must separate the label from the command name. Labels have different formats for S and non-S procedures:
  - The S label consists of a name with up to 255 characters (corresponding to <structured-name 1..255>), which is followed by a colon. S labels do not belong to the command. They are analyzed by SDF-P only and can only be used in S procedures.
  - The non-S label consists of a leading period and a name with up to 8 characters (corresponding to <name 1..8>).
- The command/statement name must be separated from the ensuing operand(s) by at least one blank.
- Operands have to be separated from each other by a comma.
- Operand values within a list must be separated by a comma. If several list elements are specified, the enumeration must be enclosed in parentheses.
- Operands may be entered either as keyword operands or as positional operands. In keyword operands, an equals sign links operand name and operand value. In positional operands, only the operand value is specified, the assignment being determined via the relative position of the operand within the input stream (see page 97).
- Further blanks between keywords, variable operand values, commas and equals signs are possible for documentation purposes and are ignored.
- Strings enclosed in quotes are interpreted as comments and ignored. Comments can be used in the same way as further blanks, but they are not permitted in front of labels.
- End-of-line comments:
  In S procedures the character string &* introduces an end-of-line comment, i.e. all subsequent characters up to the end of the line are ignored. This means especially that continuation characters, semicolons and &-expressions lose their special meaning.

- Continuation lines:
  A hyphen as the last character of an input record (with any number of trailing blanks) is interpreted as a continuation character. The ensuing input line thus becomes the continuation line of the preceding command or statement.
  If input is from the display terminal, the system displays "/" or "//" as a prompt requesting continuation of input. If commands/statements are entered from procedure/ENTER files, the continuation line must start with "/" or "//",
  The continuation character must be written within the range of columns 2 to 72 if CONTINUATION=*NEW-MODE has been set (see SHOW-SDF-OPTIONS or MODIFY-SDF-OPTIONS). In the case of CONTINUATION=*OLD-MODE the continuation character can only be in column 72. Any characters following after column 72 are ignored. The maximum total length of a command (including any continuation lines) may be 16364 bytes including any blanks and comments. The total length of ISP commands (which are still supported) is 4096 bytes.

  Input records for a statement may be longer than 72 characters and may have a continuation character in any column as of column 2. The maximum length of a statement is 16364 bytes.

  In interactive mode (terminal input) the input record length and the position of any continuation character depends on the size of the input buffer of the display terminal, but must not exceed 16364 bytes.

  In S procedures, INPUT-FORMAT=*FREE-RECORD-LENGTH can be defined by means of the SET-PROCEDURE-OPTIONS command, which means that the length of command input records is optional as long as it does not exceed the maximum length of 16364 bytes. The continuation character may be entered in any column as of column 2.

- For input from non-S procedure files or ENTER files, keywords must be in uppercase format. The same is true of the values of procedure parameters, S variables and job variables if they are to replace part of the input.

- Using a semicolon between two commands permits the simultaneous input of a number of commands within one input record (with the possibility of continuation lines). This option exists in interactive mode and S procedures only.

3.2 SDF syntax representation

The following example shows the representation of the syntax of a command in a manual. The command format consists of a field with the command name. All operands with their legal values are then listed. Operand values which introduce structures and the operands dependent on these operands are listed separately.

```
HELP-SDF                                                        Alias: HPSD

 GUIDANCE-MODE = *NO / *YES
,SDF-COMMANDS = *NO / *YES
,ABBREVIATION-RULES = *NO / *YES
,GUIDED-DIALOG = *YES (...)
    *YES(...)
      |  SCREEN-STEPS = *NO / *YES
      | ,SPECIAL-FUNCTIONS = *NO / *YES
      | ,FUNCTION-KEYS = *NO / *YES
      | ,NEXT-FIELD = *NO / *YES
,UNGUIDED-DIALOG = *YES (...) / *NO
    *YES(...)
      |  SPECIAL-FUNCTIONS = *NO / *YES
      | ,FUNCTION-KEYS = *NO / *YES
```

This syntax description is valid for SDF V4.5A.The syntax of the SDF command/statement language is explained in the following three tables.

Table 1: Notational conventions

The meanings of the special characters and the notation used to describe command and statement formats are explained in table 1.

Table 2: Data types

Variable operand values are represented in SDF by data types. Each data type represents a specific set of values. The number of data types is limited to those described in table 2.

The description of the data types is valid for the entire set of commands/statements. Therefore only deviations (if any) from the attributes described here are explained in the relevant operand descriptions.

Table 3: Suffixes for data types

Data type suffixes define additional rules for data type input. They contain a length or interval specification and can be used to limit the set of values (suffix begins with *without*), extend it (suffix begins with *with*), or declare a particular task mandatory (suffix begins with *mandatory*). The following short forms are used in this manual for data type suffixes:

cat-id	cat
completion	compl
correction-state	corr
generation	gen
lower-case	low
manual-release	man
odd-possible	odd
path-completion	path-compl
separators	sep
temporary-file	temp-file
underscore	under
user-id	user
version	vers
wildcard-constr	wild-constr
wildcards	wild

The description of the ‘integer’ data type in table 3 contains a number of items in italics; the italics are not part of the syntax and are only used to make the table easier to read.
For special data types that are checked by the implementation, table 3 contains suffixes printed in italics (see the *special* suffix) which are not part of the syntax.

The description of the data type suffixes is valid for the entire set of commands/statements. Therefore only deviations (if any) from the attributes described here are explained in the relevant operand descriptions.

Metasyntax

Representation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters denote keywords (command, statement or operand names, keyword values) and constant operand values. Keyword values begin with *.	**HELP-SDF** **SCREEN-STEPS** = <u>***NO**</u>
UPPERCASE LETTERS in boldface	Uppercase letters printed in boldface denote guaranteed or suggested abbreviations of keywords.	**GUID**ANCE-**MODE** = ***Y**ES
=	The equals sign connects an operand name with the associated operand values.	**GUID**ANCE-**MODE** = <u>***NO**</u>
< >	Angle brackets denote variables whose range of values is described by data types and suffixes (see tables 2 and 3).	**SYNTAX-F**ILE = <filename 1..54>
<u>Underscoring</u>	Underscoring denotes the default value of an operand.	**GUID**ANCE-**MODE** = <u>***NO**</u>
/	A slash serves to separate alternative operand values.	**NEXT-FIELD** = <u>***NO**</u> / ***Y**ES
(...)	Parentheses denote operand values that initiate a structure.	,**UNGUID**ED-**DIA**LOG = <u>***Y**ES</u> (...) / ***NO**
[ ]	Square brackets denote operand values which introduce a structure and are optional. The subsequent structure can be specified without the initiating operand value.	**SELECT** = [***BY-ATTR**IBUTES](...)

Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	,**GUID**ED-**DIA**LOG = <u>***Y**ES</u> (...)
		  <u>***Y**ES</u>(...)
		    | **SCREEN-STEPS** = <u>***NO**</u> /
		    | ***Y**ES

Table 1: Metasyntax (part 1 of 2)

Representation	Meaning	Examples

|	A vertical bar identifies related operands within a structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical bars preceding an operand corresponds to the depth of the structure.	**SUP**PORT = ***TAPE**(...)
		    ***TAPE(...)**
		    | **VOL**UME = ***ANY**(...)
		    | ***ANY**(...)
		    | | ...
		    | |

,	A comma precedes further operands at the same structure level.	**GUID**ANCE-**MODE** = ***NO** / ***Y**ES ,**SDF-COM**MANDS = ***NO** / ***Y**ES
list-poss(n):	The entry “list-poss” signifies that a list of operand values can be given at this point. If (n) is present, it means that the list must not have more than n elements. A list of more than one element must be enclosed in parentheses.	list-poss: ***SAM** / ***ISAM** list-poss(40): <structured-name 1..30> list-poss(256): ***OMF** / ***SYSLST**(...) / <filename 1..54>
Alias:	The name that follows represents a guaranteed alias (abbreviation) for the command or statement name.	**HELP-SDF** Alias: **HPSDF**

Table 1: Metasyntax (part 2 of 2)

Data types

Data type	Character set	Special rules
alphanum-name	A…Z 0…9 $, #, @	
cat-id	A…Z 0…9	Not more than 4 characters; must not begin with the string PUB
command-rest	freely selectable	
composed-name	A…Z 0…9 $, #, @ hyphen period catalog ID	Alphanumeric string that can be split into multiple substrings by means of a period or hyphen. If a file name can also be specified, the string may begin with a catalog ID in the form :cat: (see data type filename).
c-string	EBCDIC character	Must be enclosed within single quotes; the letter C may be prefixed; any single quotes occurring within the string must be entered twice.
date	0…9 Structure identifier: hyphen	Input format: yyyy-mm-dd jjjj: year; optionally 2 or 4 digits mm: month tt: day
device	A…Z 0…9 hyphen	Character string, max. 8 characters in length, corresponding to a device available in the system. In guided dialog, SDF displays the valid operand values. For notes on possible devices, see the relevant operand description.
fixed	+, - 0…9 period	Input format: [sign][digits].[digits] [sign]: + or - [digits]: 0...9 must contain at least one digit, but may contain up to 10 characters (0...9, period) apart from the sign.

Table 2: Data types (part 1 of 6)

Data type	**Character set**	**Special rules**
filename	A…Z 0…9 $, #, @ hyphen period	Input format: $[:\text{cat}:][\$\text{user.}]\left\{\begin{array}{l}\text{file}\\ \text{file(no)}\\ \text{group}\\ \text{group}\left\{\begin{array}{l}(*\text{abs})\\ (+\text{rel})\\ (-\text{rel})\end{array}\right\}\end{array}\right\}$:cat: optional entry of the catalog identifier; character set limited to A...Z and 0...9; maximum of 4 characters; must be enclosed in colons; default value is the catalog identifier assigned to the user ID, as specified in the user catalog. $user. optional entry of the user ID; character set is A…Z, 0…9, $, #, @; maximum of 8 characters; first character cannot be a digit; $ and period are mandatory; default value is the user's own ID. $. (special case) system default ID file file or job variable name; may be split into a number of partial names using a period as a delimiter: $\text{name}_1[.\text{name}_2[...]]$ name_i does not contain a period and must not begin or end with a hyphen; file can have a maximum length of 41 characters; it must not begin with a $ and must include at least one character from the range A...Z.

Table 2: Data types (part 2 of 6)

Data type	**Character set**	**Special rules**
filename (contd.)		#file (special case) @file (special case) \# or @ used as the first character indicates temporary files or job variables, depending on system generation. file(no) tape file name no: version number; character set is A...Z, 0...9, $, #, @. Parentheses must be specified. group name of a file generation group (character set: as for “file”) $\text{group}\left\{\begin{array}{l}\text{(*abs)}\\ \text{(+rel)}\\ \text{(-rel)}\end{array}\right\}$ (*abs) absolute generation number (1-9999); * and parentheses must be specified. (+rel) (-rel) relative generation number (0-99); sign and parentheses must be specified.
integer	0…9, +, -	+ or -, if specified, must be the first character.
name	A…Z 0…9 $, #, @	Must not begin with 0...9.

Table 2: Data types (part 3 of 6)

Data type	**Character set**	**Special rules**
partial-filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.][partname.] :cat: see filename $user. see filename partname optional entry of the initial part of a name common to a number of files or file generation groups in the form: name_1.[name_2.[...]] name_i (see filename). The final character of “partname” must be a period. At least one of the parts :cat:, $user. or partname must be specified.
posix-filename	A...Z 0...9 special characters	String with a length of up to 255 characters; consists of either one or two periods or of alphanumeric characters and special characters. The special characters must be escaped with a preceding \ (backslash); the / is not allowed. Must be enclosed within single quotes if alternative data types are permitted, separators are used, or the first character is a ?, ! or ^. A distinction is made between uppercase and lowercase.

posix-pathname	A...Z
	0...9
	special characters
	structure identifier:
	slash
		Input format: [/]part_1/.../part_n
		where part_i is a posix-filename;
		max. 1023 characters;
		must be enclosed within single quotes if alternative data types are permitted, separators are used, or the first character is a ?, ! or ^.

Table 2: Data types (part 4 of 6)

Data type	**Character set**	**Special rules**
product-version	A…Z 0…9 period single quote	Input format: [[C]’][V][m]m.naso[’] correction status: o release status: a where m, n, s and o are all digits and a is a letter. Whether the release and/or correction status may/must be specified depends on the suffixes to the data type (see suffixes without-corr, without-man, mandatory-man and mandatory-corr in table 3). product-version may be enclosed within single quotes (possibly with a preceding C). The specification of the version may begin with the letter V.
structured-name	A…Z 0…9 $, #, @ hyphen	Alphanumeric string which may comprise a number of substrings separated by a hyphen. First character: A...Z or $, #, @
text	freely selectable	For the input format, see the relevant operand descriptions.
time	0…9 structure identifier: colon	Time-of-day entry: Input format: { hh:mm:ss / hh:mm / hh } hh: hours mm: minutes ss: seconds } Leading zeros may be omitted
vsn	a) A…Z 0…9 b) A…Z 0…9 $, #, @	a) Input format: pvsid.sequence-no max. 6 characters pvsid: 2-4 characters; PUB must not be entered sequence-no: 1-3 characters b) Max. 6 characters; PUB may be prefixed, but must not be followed by $, #, @.

Table 2: Data types (part 5 of 6)

Data type	**Character set**	**Special rules**
x-string	Hexadecimal: 00…FF	Must be enclosed in single quotes; must be prefixed by the letter X. There may be an odd number of characters.
x-text	Hexadecimal: 00…FF	Must not be enclosed in single quotes; the letter X must not be prefixed. There may be an odd number of characters.

Table 2: Data types (part 6 of 6)

Suffixes for data types

Suffix	Meaning
x..y *unit*	With data type “integer”: interval specification x minimum value permitted for “integer”. x is an (optionally signed) integer. y maximum value permitted for “integer”. y is an (optionally signed) integer. *unit* with “integer” only: additional units. The following units may be specified: *days* *byte* *hours* *2Kbyte* *minutes* *4Kbyte* *seconds* *Mbyte* *milliseconds*
x..y *special*	With the other data types: length specification For data types catid, date, device, product-version, time and vsn the length specification is not displayed. x minimum length for the operand value; x is an integer. y maximum length for the operand value; y is an integer. x=y the length of the operand value must be precisely x. *special* Specification of a suffix for describing a special data type that is checked by the implementation. “special” can be preceded by other suffixes. The following specifications are used: *arithm-expr* arithmetic expression (SDF-P) *bool-expr* logical expression (SDF-P) *string-expr* string expression (SDF-P) *expr* freely selectable expression (SDF-P) *cond-expr* conditional expression (JV) *symbol* CSECT or entry name (BLS)
with	Extends the specification options for a data type.
-compl	When specifying the data type “date”, SDF expands two-digit year specifictions in the form yy-mm-dd to: 20jj-mm-tt if jj < 60 19jj-mm-tt if jj ≥ 60
-low	Uppercase and lowercase letters are differentiated.
-path-compl	For specifications for the data type “filename”, SDF adds the catalog and/or user ID if these have not been specified.

Table 3: Data type suffixes (part 1 of 7)

Suffix	Meaning	
with (contd.)		
-under	Permits underscores (_) for the data type “name”.	
-wild(n)	Parts of names may be replaced by the following wildcards. n denotes the maximum input length when using wildcards. Due to the introduction of the data types posix-filename and posix-pathname, SDF now accepts wildcards from the UNIX world (referred to below as POSIX wildcards) in addition to the usual BS2000 wildcards. However, as not all commands support POSIX wildcards, their use for data types other than posix-filename and posix-pathname can lead to semantic errors. Only POSIX wildcards or only BS2000 wildcards should be used within a search pattern. Only POSIX wildcards are allowed for the data types posix-filename and posix-pathname. If a pattern can be matched more than once in a string, the first match is used.	
	BS2000 wildcards	Meaning
	*	Replaces an arbitrary (even empty) character string. If the string concerned starts with *, then the * must be entered twice in succession if it is followed by other characters and if the character string entered does not contain at least one other wildcard.
	Termina-ting period	Partially-qualified entry of a name. Corresponds implicitly to the string “./*”, i.e. at least one other character follows the period.
	/	Replaces any single character.
	<s_x:s_y>	Replaces a string that meets the following conditions: – It is at least as long as the shortest string (s_x or s_y) – It is not longer than the longest string (s_x or s_y) – It lies between s_x and s_y in the alphabetic collating sequence; numbers are sorted after letters (A...Z, 0...9) – s_x can also be an empty string (which is in the first position in the alphabetic collating sequence) – s_y can also be an empty string, which in this position stands for the string with the highest possible code (contains only the characters X’FF’)
	<s_1,…>	Replaces all strings that match any of the character combinations specified by s. s may also be an empty string. Any such string may also be a range specification “s_x:s_y” (see above).

Table 3: Data type suffixes (part 2 of 7)

Suffix	**Meaning**	
with-wild(n) (contd.)	-s	Replaces all strings that do not match the specified string s. The minus sign may only appear at the beginning of string s. Within the data types filename or partial-filename the negated string -s can be used exactly once, i.e. -s can replace one of the three name components: cat, user or file.
	Wildcards are not permitted in generation and version specifications for file names. Only system administration may use wildcards in user IDs. Wildcards cannot be used to replace the delimiters in name components cat (colon) and user ($ and period).	
	POSIX wildcards	Meaning
	*	Replaces any single string (including an empty string). An * appearing at the first position must be duplicated if it is followed by other characters and if the entered string does not include at least one further wildcard.
	?	Replaces any single character; not permitted as the first character outside single quotes.
	[c_x-c_y]	Replaces any single character from the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters.
	[s]	Replaces exactly one character from string s. The expressions [c_x-c_y] and [s] can be combined into [s_1c_x-c_ys_2].
	[!c_x-c_y]	Replaces exactly one character not in the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters. The expressions [!c_x-c_y] and [!s] can be combined into [!s_1c_x-c_ys_2].
	[!s]	Replaces exactly one character not contained in string s. The expressions [!s] and [!c_x-c_y] can be combined into [!s_1c_x-c_ys_2].

Table 3: Data type suffixes (part 3 of 7)

Suffix	**Meaning**
with (contd.)	
wild-constr(n)	Specification of a constructor (string) that defines how new names are to be constructed from a previously specified selector (i.e. a selection string with wildcards). See also with-wild. n denotes the maximum input length when using wildcards. The constructor may consist of constant strings and patterns. A pattern (character) is replaced by the string that was selected by the corresponding pattern in the selector. The following wildcards may be used in constructors:
	Wildcard / **Meaning**
	* — Corresponds to the string selected by the wildcard * in the selector.
	Terminating period — Corresponds to the partially-qualified specification of a name in the selector; corresponds to the string selected by the terminating period in the selector.
	/ or ? — Corresponds to the character selected by the / or ? wildcard in the selector.
	<n> — Corresponds to the string selected by the n-th wildcard in the selector, where n is an integer.
	Allocation of wildcards to corresponding wildcards in the selector: All wildcards in the selector are numbered from left to right in ascending order (global index). Identical wildcards in the selector are additionally numbered from left to right in ascending order (wildcard-specific index). Wildcards can be specified in the constructor by one of two mutually exclusive methods: 1. Wildcards can be specified via the global index: <n> 2. The same wildcard may be specified as in the selector; substitution occurs on the basis of the wildcard-specific index. For example: the second “/” corresponds to the string selected by the second “/” in the selector

Table 3: Data type suffixes (part 4 of 7)

Suffix	**Meaning**
with-wild-constr(n) (contd.)	The following rules must be observed when specifying a constructor: – The constructor can only contain wildcards of the selector. – If the string selected by the wildcard <...> or [...] is to be used in the constructor, the index notation must be selected. – The index notation must be selected if the string identified by a wildcard in the selector is to be used more than once in the constructor. For example: if the selector “A/” is specified, the constructor “A<n><n>” must be specified instead of “A//”. – The wildcard * can also be an empty string. Note that if multiple asterisks appear in sequence (even with further wildcards), only the last asterisk can be a non-empty string, e.g. for “****” or “*//*”. – Valid names must be produced by the constructor. This must be taken into account when specifying both the constructor and the selector. – Depending on the constructor, identical names may be constructed from different names selected by the selector. For example: “A/*” selects the names “A1” and “A2”; the constructor “B*” generates the same new name “B” in both cases. To prevent this from occurring, all wildcards of the selector should be used at least once in the constructor. – If the constructor ends with a period, the selector must also end with a period. The string selected by the period at the end of the selector cannot be specified by the global index in the constructor specification.

Table 3: Data type suffixes (part 5 of 7)

Suffix	Meaning
with-wild-constr(n) (contd.)	Examples:

Selector	Selection	Constructor	New name
A//*	AB1 AB2 A.B.C	D<3><2>	D1 D2 D.CB
C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<3>.XY<2>	G.A.D.XYA G.A.D.XYB G.B.F.XYA G.B.F.XYB
C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<2>.XY<2>	G.A.A.XYA G.A.B.XYB G.B.A.XYA G.B.B.XYB
A//B	ACDB ACEB AC.B A.CB	G/XY/	GCXYD GCXYE GCXY. [1] G.XYC

[1] The period at the end of the name may violate naming conventions (e.g. for fully-qualified file names).

without	Restricts the specification options for a data type.
-cat	Specification of a catalog ID is not permitted.
-corr	Input format: [[C]'][V][m]m.na[''] Specifications for the data type product-version must not include the correction status.
-gen	Specification of a file generation or file generation group is not permitted.
-man	Input format: [[C]'][V][m]m.n[''] Specifications for the data type product-version must not include either release or correction status.
-odd	The data type x-text permits only an even number of characters.
-sep	With the data type "text", specification of the following separators is not permitted: ; = ( ) < > ␣ (i.e. semicolon, equals sign, left and right parentheses, greater than, less than, and blank).
-temp-file	Specification of a temporary file is not permitted (see #file or @file under filename).

Table 3: Data type suffixes (part 6 of 7)

Suffix	**Meaning**
without (contd.)	
-user	Specification of a user ID is not permitted.
-vers	Specification of the version (see “file(no)”) is not permitted for tape files.
-wild	The file types posix-filename and posix-pathname must not contain a pattern (character).
mandatory	Certain specifications are necessary for a data type.
-corr	Input format: [[C]’][V][m]m.naso[’] Specifications for the data type product-version must include the correction status and therefore also the release status.
-man	Input format: [[C]’][V][m]m.na[so][’] Specifications for the data type product-version must include the release status. Specification of the correction status is optional if this is not prohibited by the use of the suffix without-corr.
-quotes	Specifications for the data types posix-filename and posix-pathname must be enclosed in single quotes.

Table 3: Data type suffixes (part 7 of 7)

3.3 Language for SDF output

All SDF output occurs in the language that has been set for the output of system messages. The syntax files supplied by Fujitsu Siemens Computers contain texts for the respective language codes (D=German, E=English). These texts are:

– command/statement-specific help texts
– help texts on data types
– SDF-specific messages (e.g. command input request “%KDO:” or “%CMD:”)
– identifiers used in guided dialog (names of input fields, identifiers of data types and their suffixes)

The system language is globally defined at system generation. In addition, system administration can specify the desired language for each user ID in the user catalog; this entry supersedes the global setting made at system generation time. Users may obtain the currently applicable setting by issuing the SHOW-USER-ATTRIBUTES command.
The user may also change the system language for a particular task by means of the MODIFY-MSG-ATTRIBUTES command.

If the language code set is not defined in the syntax files activated by a task, i.e. the global information in the activated syntax files does not contain any general text for the language code set, then SDF uses the language set at system generation time and informs the user of this fact by means of message CMD0159.

If the activated syntax files do contain global information for the language code set, then SDF outputs the command/ statement-specific help texts defined for this language code. Any help texts that may not be defined for this language code are output in the language set at system generation time.

3.4 Command return code

SDF provides the user with information on the analysis of command input and execution in the form of a command return code. This command return code is comparable to return codes at program level. It enables users to take specific action in response to typical error situations.

Structure of command return codes

Command return codes are comprised of three parts:

- the main code, a message code that can be specified with the HELP-MSG-INFORMATION command in order to obtain detailed information.
- subcode 1, which assigns the error situation to an error class that indicates how serious an error is. Subcode 1 has a *decimal* value.
  The following five error classes are defined in BS2000:
  - Class A: no error

    The value is zero. Processing can proceed normally.
  - Class B: syntax error

    The value is a number between 1 and 31. The command was entered with incorrect syntax. The command can be entered again after the syntax error has been corrected.
  - Class C: internal error (system error)

    The value is 32. Input can be repeated if the internal error has been recovered.
  - Class D: errors that cannot be assigned to any other error class.

    The value is a number between 64 and 127. Evaluate the main code to determine how to proceed.
  - Class E: the command cannot be executed at the moment.

    The value is a number between 128 and 130. Command input can be repeated without modification. The command will be executed after a wait time has elapsed. The wait time may be short, long or indefinite.
    128 indicates a short wait time, considered practical in interactive mode.
    129 indicates a long wait time, considered practical in batch mode.

    130 indicates an indefinite wait time and uncertainty as to whether the error will be recovered at all.

- subcode 2, which can contain additional information (value is not zero). Subcode 2 has a **decimal** value. In the event of an error (subcode 1 is not zero), there are no rules regarding the use of subcode 2. The value of subcode 2 may be zero, 1 or 2 if no error has occurred. Subcode 2 with the value 1 indicates that the requested function was already available before the command was issued. Subcode 2 with the value 2 indicates an exception situation and should be classified as a warning.

Command return codes can only be evaluated with SDF-P resources in *S procedures* and dialog blocks (see the “SDF-P” manual [6]).
For information on evaluation see also the descriptions of the IF-BLOCK-ERROR command and of the builtin functions MAINCODE, SUBCODE1 and SUBCODE2 in Volumes 1 and 4 of the “Commands, volume 6” manual [2].

Representation of command return codes

Command return codes are represented in tabular form in the order: subcode 2, subcode 1, main code, meaning.
A subcode 2 with the value zero (i.e. no additional information present) is not listed in the table.
If guaranteed messages are output in connection with a specific command return code, the heading “Meaning” is supplemented by “/ Guaranteed messages”, and the message codes of the guaranteed messages are given after the meaning. The user can output the meanings of these message codes with the HELP-MSG command.

Example:

(SC2)	SC1	Maincode	Meaning / Guaranteed messages
	0	CMD0001	Command terminated without errors
1	32	CMD0500	Errored syntax description in current syntax file Guaranteed message: CMD0500
	64	CMD0554	Command execution not successful. Guaranteed messages: CMD0300, CMD0302, CMD0490, CMD0508, CMD0509, CMD0552, CMD0554, CMD0555, CMD0579

General command return codes

Command return codes which a BS2000 command may pass upon execution are part of the respective command descriptions.
The table below contains general command return codes that are automatically passed by SDF if

- SDF senses an error prior to command execution (e.g. syntax error)
- the command cannot be executed
- the command, i.e. the execution module itself, does not pass a command return code; in this case the command description does not contain any specific command return codes.

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	0	CMD0002	Command executed with warning 4)
1	0	CMD0026	Command no longer supported
2	0	CMD0093	Procedure ended in test mode 1)
2	0	CMD0201	End of file reached 1)
2	0	CMD0214	End of program 3)
2	1	CMD0211	SDF transfer area too small
	1	CMD0202	Syntax error 5) 6)
	1	CMD0205	Spin-off 1) 2)
	1	CMD2201	Parameter error
	2	CMD0200	Command presently not available
	2	CMD2202	Subsystem not defined
	3	CMD2203	Installation error
	32	CMD0221	Internal SDF error
	32	CMD2009	Error during output to variables
	64	CMD0216	Requisite privilege missing
	65	CMD2241	Subsystem not loaded
	66	CMD2242	Subsystem not connected
	128	CMD2280	Subsystem not available for a short period
	129	CMD2281	Subsystem not available for a long period
	130	CMD2282	Subsystem not available for an indefinite period
	130	NBR0748	After /SHUTDOWN, the only command that can be entered at the operator terminal is /SHOW-PENDING-MSG

1) Command return codes CMD0093, CMD0201 and CMD0205 are not possible if the command is issued via the CMD macro.

2) Command return code CMD0205 is passed if spin-off was initiated. Example: end of program with TERM macro call with UNIT=STEP, and no command return code has been set with the CMDRC macro call during the program run.

3) Command return code CMD0214 cannot be passed to the calling program by the CMD macro because the program has been terminated by CMD macro execution.

4) Command return code CMD0002 is passed by SDF if a command triggers spin-off despite error-free execution. This compatibly supports the previous spin-off behavior of commands (and thus error handling in non-S and ENTER procedures). In S procedures in which ERROR-MECHANISM=*BY-RETURNCODE applies (see the SET-/MODIFY-PROCEDURE-OPTIONS command), error recovery is not initiated automatically since the related subcode 1 has the value zero.

5) Command return code CMD0202 is passed by SDF if a command does not initiate spin-off on errored execution. CMD0202 indicates syntactical and semantic errors detected during command execution. This compatibly supports the previous spin-off behavior of commands (and thus error handling in non-S and ENTER procedures).
In S procedures in which ERROR-MECHANISM=*BY-RETURNCODE applies (see the SET-/MODIFY-PROCEDURE-OPTIONS command), error recovery is initiated since the related subcode 1 has a value other than zero.

6) Command return code CMD0202 is passed by SDF from the execution module for both syntactical and semantic errors.

Notes

– In user programs the CMDRC macro call can be used to set a command return code, which is retained up to the next CMDRC macro call. On program termination the current command return code from the program is passed to the caller and triggers error recovery in S procedures for subcode 1 with a value other than zero if ERROR-MECHANISM =*BY-RETURNCODE was specified (see the SET-/MODIFY-PROCEDURE-OPTIONS command).

– As of SDF-P V2.1A, return codes set in the program can be evaluated like command return codes after each statement (see the BEGIN-BLOCK command in the “SDF-P” manual [6]).

4 Input of commands and statements

Commands and statements may be entered at the data display terminal either in guided or unguided dialog. To perform a given input it is possible to switch temporarily from unguided to guided dialog.
It is also possible to store the commands/statements in a file or a compound S variable for subsequent input by a procedure or in batch mode.

The command/statement input may contain expressions which SDF will replace by the contents of a job variable or S variable or, in procedures, by the value of a procedure parameter (see section “Replacing expressions in the input” on page 87).

Most of the information contained in this chapter can be displayed on the screen by means of the HELP-SDF command (see page 164).

4.1 Input in unguided dialog

There are two forms of unguided dialog, the EXPERT form and the NO form. The appropriate selection is made via the GUIDANCE operand of the MODIFY-SDF-OPTIONS command. The two forms differ primarily in the question whether or not SDF initiates a dialog to correct errored entries. Changeover to a different dialog form is possible (see pages 57 and 76).

4.1.1 EXPERT form

After processing of the LOGON command, user guidance is preset to the EXPERT form, provided no other specification was made in the global information of the activated syntax files. If the user is in a different dialog mode, he may switch to the EXPERT form by entering MODIFY-SDF-OPTIONS GUIDANCE=*EXPERT.

SDF displays “/” to request command input (or “//” to request statement input). In the event of an error, error messages are displayed but SDF does not initiate a correction dialog. Redisplaying the input via RESTORE-SDF-INPUT is possible, as is temporarily switching over to guided dialog to repeat the input (see page 76).

4.1.2 NO form

MODIFY-SDF-OPTIONS GUIDANCE=*NO is entered to switch to the NO form.

SDF displays “%CMD:” to request command input (or “%STMT:” to request statement input). In the event of an error, SDF displays error messages and initiates a correction dialog. Entry of a question mark causes a temporary switchover to guided dialog just for the next entry (see page 76). The last input can be redisplayed by means of the RESTORE-SDF-INPUT command (see page 196).

4.1.3 Special entries

!	as an operand value sets the default value for this operand. Subsequent characters need not be deleted.
!	as the first character of the command/statement name defines the specified operand values as task-specific default values of the command/statement. The command/statement is not executed.
?	as an operand value outputs information on all valid operand values. In the NO form, the input is redisplayed and the user is requested to enter the operands. “?” after a command name causes a changeover to temporarily guided dialog (see page 76).
??	as an operand value outputs information on the data types permitted as an operand value.
^	as the operand value of a “secret“ operand requests a blanked input field for confidential entry of the operand value.
-	as the final character indicates that the command or statement is continued in the next input line.

$\boxed{\text{LZF}}$ key

deletes all characters in the input line from the cursor onwards. (LZF = abbreviated German for “delete character string”.)

$\boxed{\text{LZE}}$ key

enables blocked input (see section “Blocked input” on page 55). (LZE = abbreviated German for “logical end-of-line”.)

The same operand may be specified more than once in the same command, but only the last specification of this operand will be accepted.

If the operand value *SECRET is entered (implicitly via default value, or explicitly) SDF outputs a blanked input field for masked entry of the operand value (see page 76).

4.1.4 Function keys

Certain functions can be executed simply by pressing a function key. The effects of the function keys depend on how the SDF option FUNCTION-KEYS is set.
There are two assignment modes: the previous ‘Old mode’ (*OLD-MODE) and the new ’Styleguide mode’ (*STYLE-GUIDE-MODE), which offers greater functionality. The terminal or terminal emulation, however, must support Styleguide mode and be generated accordingly.

*OLD-MODE	
Function key	**Effect**
K1	Exit function Exits the current program. A message prompts the user to confirm whether the program should be terminated. During a correction dialog at command/statement entry (see the NO form), the function terminates only the correction dialog. Equivalent to F3 in Styleguide mode.
K2	Interrupt function Interrupts a current program or a procedure, or aborts output of a command.
F1	Exit-all function Exits the current program. A message prompts the user to confirm whether the program should be terminated. During a correction dialog at command/statement entry (see the NO form), the function terminates only the correction dialog. Equivalent to F6 in Styleguide mode.

Table 4: Function key assignment (Old mode) in unguided dialog

*STYLE-GUIDE-MODE	
Function key	**Effect**
K2	Interrupt function Interrupts a current program or a procedure, or aborts output of a command.
F1	Help function Switches to temporarily guided dialog. Equivalent to entering “?”.
F3	Exit function Exits the current program. A message prompts the user to confirm whether the program should be terminated. During a correction dialog at command/statement entry (see NO form), the function terminates only the correction dialog. Equivalent to K1 in Old mode.

Table 5: Function key assignment (Styleguide mode) in unguided dialog (part 1 of 2)

*STYLE-GUIDE-MODE	
Function key	**Effect**
F6	Exit-all function Exits the current program. A message prompts the user to confirm whether the program should be terminated. During a correction dialog at command/statement entry (see the NO form), the function terminates only the correction dialog. Equivalent to F1 in Old mode.
F9	Redisplays the last command or statement entered. Equivalent to entering “RESTORE-SDF-INPUT” without operands (default: INPUT=*LAST-CMD or *LAST-STMT).
F12	Cancel function Exits the current program. A message prompts the user to confirm whether the program should be terminated. During a correction dialog at command/statement entry (see the NO form), the function terminates only the correction dialog.

Table 5: Function key assignment (Styleguide mode) in unguided dialog (part 2 of 2)

Note

In Styleguide mode, the function keys K1, K3, F2, F4, F5, F7, F8, F10, F11 und F13 through F24 are not supported. Pressing an unsupported function key causes an error message to be displayed.

4.1.5 Blocked input

Both in the EXPERT and the NO form of unguided dialog, a number of commands or statements can be issued at the same time (blocked input). Blocked input permits one or more logical lines to be entered, each line usually containing one command or statement and ending with logical end-of-line (LZE key). Blocked input ends with the end marker (EM key). SDF redisplays each analyzed command/statement of the blocked input **unchanged** with the prefix “(BL)” prior to execution.
If a command or statement is invalid and the error is not interactively corrected, any subsequent commands/ statements are no longer analyzed, SDF displays the corresponding logical input lines **unchanged** on the screen together with message CMD0122. Passwords are not masked out in the log.

It is also permissible to enter more than one command/ statement in a logical input line, provided they are separated by semicolon. The contents of a single logical input line are always treated as one logical input stream: it is not permissible to enter commands read from SYSCMD and statements read from SYSSTMT in one and the same logical input line.

Example

```
/add-file-link link=sortin01,file-name=datei.1<
add-file-link link=sortin02,file-name=datei.2<
add-file-link link=sortout,file-name=out.sort1-2<
start-sort;sort-record;end<
show-file-attr out.sort1-2;show-file-attr datei.;show-file-link link=sort*◁
(BL) /add-file-link link=sortin01,file-name=datei.1
(BL) add-file-link link=sortin02,file-name=datei.2
(BL) add-file-link link=sortout,file-name=out.sort1-2
(BL) start-sort
%  BLS0523 ELEMENT 'SRT80', VERSION '078' FROM LIBRARY ':2OSH:$TSOS.SYSLNK.SORT.
078' IN PROCESS
%  BLS0524 LLM 'SRT80', VERSION '078' OF '2001-08-24 15:20:53' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2001.ALL RIGHTS RESERVED
%  SRT1001  18:49:13/000000.00 SORT/MERGE STARTED, VERSION 07.8A00/BS2000V14.0
%  SRT1130  PLEASE ENTER SORT STATEMENTS
(BL) sort-record
(BL) end
%  SRT1016  SORT/MERGE INPUT RECORDS:.............................688 (FROM 01)
%  SRT1016  SORT/MERGE INPUT RECORDS:...........................2.003 (FROM 02)
%  SRT1017  RECORDS TO BE SORTED/MERGED:........................2.691
%  SRT1030  SORT/MERGE OUTPUT RECORDS:..........................2.691
%  SRT1002  18:49:14/000000.19 SORT/MERGE COMPLETED
(BL) show-file-attr out.sort1-2
%        66 :2OSG:$USER1.OUT.SORT1-2
%:2OSG: PUBLIC:      1 FILE  RES=        66 FRE=         2 REL=         0 PAGES
(BL) show-file-attr datei.
%PLEASE ACKNOWLEDGE
%        21 :2OSG:$USER1.DATEI.1
%        48 :2OSG:$USER1.DATEI.2
%:2OSG: PUBLIC:      2 FILES RES=        69 FRE=         4 REL=         0 PAGES
(BL) show-file-link link=sort*
%-- LINK-NAME --------- FILE-NAME --------------------------------------------
%   SORTOUT             :2OSG:$USER1.OUT.SORT1-2
%   SORT01              :2OSG:$USER1.DATEI.1
%   SORT02              :2OSG:$USER1.DATEI.2
/
```

4.2 Input in guided dialog

In its full scope, guided dialog is only possible with commands that are defined in the syntax files assigned and whose operands are permitted for dialog guidance.

Blanked input fields are displayed for the entry of sensitive operand values such as passwords.

In dialog with minimum or medium guidance the user can call up explanatory information on specific operands.

If the user enters a command/statement with all the necessary operand values in the NEXT line of a form or menu, the command/statement is executed without further dialog.

4.2.1 Levels of user guidance

There are three levels of guided dialog. They differ in the scope of information that SDF displays. They are selected via the GUIDANCE operand of the MODIFY-SDF-OPTIONS command. By entering a question mark in the NEXT line it is possible to switch temporarily to the next-higher guidance level.

Maximum user guidance

Entering MODIFY-SDF-OPTIONS GUIDANCE=*MAXIMUM causes a switch to maximum user guidance.

SDF displays the default values of the optional operands, lists of all the permissible operand values accompanied by explanatory remarks, and help texts for application domains, commands, statements and operands.

Medium user guidance

Entering MODIFY-SDF-OPTIONS GUIDANCE=*MEDIUM causes a switch to medium user guidance.

SDF displays the default values of the optional operands and lists of all the permissible operand values without any further explanatory remarks. It displays help texts for application domains, commands and statements only.

Minimum user guidance

Entering MODIFY-SDF-OPTIONS GUIDANCE=*MINIMUM causes a switch to minimum user guidance.

SDF displays only the default values of the optional operands. It does not display lists of all the permissible operand values or help texts.

4.2.2 Screen format

In guided dialog, SDF offers the user a screen mask with the desired information or with the appropriate input fields. The screen is always divided into three areas, whose contents may vary according to the status of the dialog.

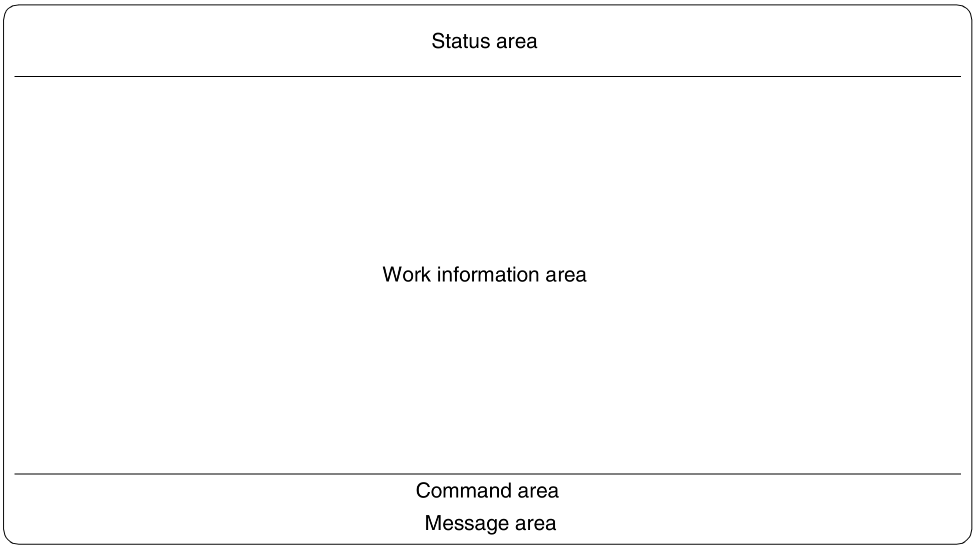


Figure 1: Screen format

The status area may contain:

- the name of the selected application domain
- additionally, the name of the selected command
- additionally, the name of the structure-initiating operand value, if the associated operands are displayed
- the operands accepted up to this point, chained into a string which is truncated on the left if it is longer than two display lines.

The work information area may contain:

- a list of all permissible application domains
- a list of all permissible commands of a selected application domain
- the names of all valid operands of a selected command, followed by an input line in which the default value is already preset; additional information on the operand is displayed in accordance with the current user guidance level.

The command area contains:

- the so-called NEXT line; following the NEXT operand, an input line for valid statements to control the menu guidance is displayed with a preset statement (all permissible control statements are listed)
- Additional information is displayed in the following lines depending on the SDF option FUNCTION-KEYS.
  - In the old mode (FUNCTION-KEYS=*OLD-MODUS) all control statements permitted are displayed. Function keys that correspond to an authorized control statement are also displayed.
  - In the Styleguide mode (FUNCTION-KEYS=*STYLE-GUIDE-MODE) the control statements permitted are only displayed in the guidance level GUIDANCE=*MEDIUM or *MAXIMUM. The function keys permitted are displayed in a separate line after the keyword KEYS in all guidance levels.

The message area contains the last error message output by SDF if applicable.

In this manual, the displayed screens are called **menus** if the work information area contains only information on application domains, commands or statements to be selected. The screens are called **operand forms**, however, if the work information area lists all the operands for which a value can be specified. The display of menus or operand forms may be distributed over several screens, depending on the output scope. Screen switchover (also called paging or scrolling) is effected via the control statements ”+” or ”–” in the NEXT line of the command area.

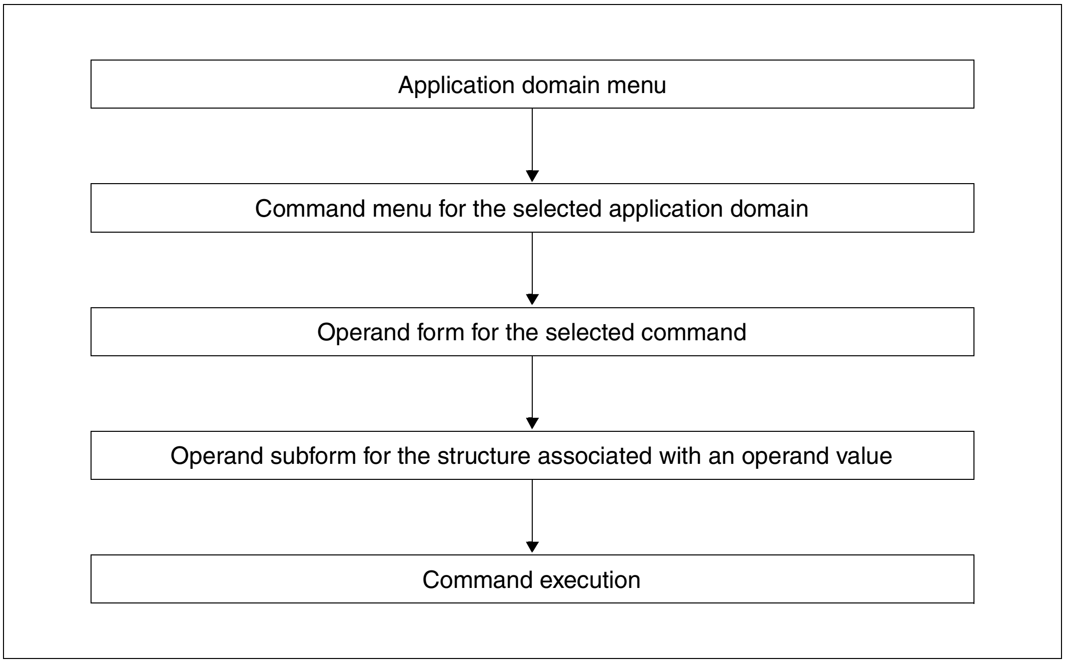

Application domain menu
Command menu for the selected application domain
Operand form for the selected command
Operand subform for the structure associated with an operand value
Command execution

Figure 2: Path from the application domain menu to command execution

4.2.3 Special entries

!	as an operand value sets the default value for this operand. Subsequent characters need not be deleted.
?	as an operand value outputs information on all valid operand values. In the NO form, the input is redisplayed and the user is requested to enter the operands. “?” after a command name causes a changeover to temporarily guided dialog (see page 76).
??	as an operand value outputs information on the data types permitted as an operand value.
^	as the operand value of a “secret” operand requests a blanked input field for confidential entry of the operand value.
([<operand>,...]	after a structure-initiating operand outputs a subform for the structure. Specified operand values are copied automatically into the subform.

([<operand>,...])

after a structure-initiating operand suppresses output of the subform for the structure. The specified values are copied over; for non-specified values, the relevant default values are set.

– as the final character indicates that the command or statement is continued in the next input line.

LZF key

deletes all characters in the input line from the cursor onwards. (LZF = abbreviated German for “delete character string”.)

4.2.4 Input in the NEXT line

The NEXT line can be used to enter control statements for menu guidance or a command/ statement in accordance with the EXPERT form of unguided dialog. The logical-line-end character (LZE key) is illegal here. The LZF key may be used to delete character strings following the desired input.

The following list contains all possible entries. Which entries are actually permitted depends on the screen contents displayed; a selection of the most important permissible entries is shown below the NEXT line.

`+, ++, -, --`	pages backward and forward in the menu (or form). ”–” is equivalent to the F7 key and ”+” to the F8 key in Styleguide mode.
*EXECUTE	executes the current operation (command or statement). Equivalent to the F3 key in Old mode or the F11 key in Styleguide mode.
*CONTINUE	outputs a subform if the current form contains at least one structure-initiating operand value. Otherwise, the current operation is executed.
*TEST	checks inputs for syntax errors. Equivalent to the F2 key in Old mode.
*CANCEL	cancels the current menu (or operand form) and switches to the higher-ranking menu or, in the case of a subform, to the higher-ranking operand form. The previous specifications in the subform are discarded. Within a statement menu in guided dialog, *CANCEL terminates the program after requesting confirmation. Equivalent to the F12 key in Styleguide mode.
*EXIT	terminates the current menu (or operand form) and switches to the higher-ranking menu. Within a statement menu in guided dialog, *EXIT terminates the program after requesting confirmation. Equivalent to the K1 key in Old mode or the F3 key in Styleguide mode.
*EXIT-ALL	cancels the current menu (or operand form) and, in guided dialog, switches to the highest-ranking menu or, in temporarily guided dialog, to command/statement input in unguided dialog. Within a statement menu in guided dialog, *EXIT-ALL terminates the program after requesting confirmation. Equivalent to the F1 key in Old mode or the F6 key in Styleguide mode.

*REFRESH	restores the last screen displayed with the original values. Equivalent to the K3 key in Old mode or the F5 key in Styleguide modus.
*DOM-MENU	switches to the application domain menu. If *DOM-MENU is entered in an operand form, the current operation is executed before the switch is made.
<number>	if entered in an application domain menu: displays the command menu of the appropriate application domain; if entered in a command or statement menu: displays the operand form of the appropriate command or statement.
(<domain>)	displays the command menu of the application domain <domain>. If (<domain>) is specified in an operand form, the current operation is executed before the switch is made.
<command>?	executes the current operation and then displays the operand form of the command <command>. Operand values which have already been specified are incorporated in the form.
<command>	executes the current operation and then the command <command> as well. <command> also implies entry of the associated operands.
!<command>	executes the current operation and then defines task-specific default values for the command <command>. <command> also implies entry of the associated operands.
<statement>?	executes the current operation and then displays the operand form of the statement <statement>. Operand values which have already been specified are incorporated in the form.
<statement>	executes the current operation and then the statement <statement> as well. <statement> also implies entry of the associated operands.
!<statement>	executes the current operation and then defines task-specific default values for the statement <statement>. <statement> also implies entry of the associated operands.
?	switches to the next-higher guidance level for the purpose of the current input. Equivalent to the F1 key in Styleguide mode.
*DOWN(<operand>)	displays the subform for the specified operand value <operand> if it is the top level of a structure.
*UP	switches from the subform back to the higher-ranking operand form. In contrast to *CANCEL, the previous specifications in the subform are retained.

4.2.5 Function keys

The user can execute certain functions simply by pressing a function key. Function keys can also be used to enter a number of inputs in the NEXT line. The effects of the functions depend on how the SDF option FUNCTION-KEYS is set (using the MODIFY-SDF-OPTIONS command).
There are two assignment modes: the previous ‘Old mode’ (*OLD-MODE) and the new ’Styleguide mode’ (*STYLE-GUIDE-MODE), which offers greater functionality. The terminal or terminal emulation, however, must support Styleguide mode and be generated accordingly. The table below shows the assignment of the function keys:

*OLD-MODE	
Function key	**Effect**
K1	Exit function. Exits the current menu (or operand form) and switches to the higher-ranking menu. Within a statement menu in guided dialog, K1 terminates the program after requesting confirmation. Equivalent to *EXIT in the NEXT line or F3 in Styleguide mode.
K2	Interrupt function. Interrupts a current program or a procedure, or aborts output of a command.
K3	Refresh function. Restores the last screen displayed with the original values. Equivalent to *REFRESH in the NEXT line or F5 in Styleguide mode.
F1	Exit-all function. Exits the current menu (or operand form) and switches in guided dialog to the highest-ranking menu or, in temporarily guided dialog, to command/statement input of unguided dialog. Within a statement menu in guided dialog, F1 terminates the program after requesting confirmation. Equivalent to *EXIT-ALL in the NEXT line or F6 in Styleguide mode.
F2	Test function. Checks inputs for syntax errors. Equivalent to *TEST in the NEXT line.
F3	Execute function. Executes the current operation (command or statement). Equivalent to *EXECUTE in the NEXT line or F11 in Styleguide mode.

Table 6: Function key assignment (Old mode) in guided dialog

*STYLE-GUIDE-MODE	
Function key	**Effect**
K2	Interrupt function. Interrupts a current program or a procedure, or aborts output of a command.
F1	Help function. Switches to temporarily guided dialog. Equivalent to entering “?” in the NEXT line.
F3	Exit function. Exits the current menu (or operand form) and switches to the higher-ranking menu. Within a statement menu in guided dialog, F3 terminates the program after requesting confirmation. Equivalent to *EXIT in the NEXT line or K1 in Old mode.
F5	Refresh function. Restores the last screen displayed with the original values. Equivalent to *REFRESH in the NEXT line or K3 in Old mode.
F6	Exit-all function. Terminates the current menu (or operand form) and switches in guided dialog to the highest-ranking menu or, in temporarily guided dialog, to command/ statement input of unguided dialog. Within a statement menu in guided dialog, F6 terminates the program after requesting confirmation. Equivalent to *EXIT-ALL in the NEXT line or F1 in Old mode.
F7	Scroll backward. Scrolls backward in menus and forms which extend over more than one screen. Equivalent to “-” in the NEXT line.
F8	Scroll forward. Scrolls forward in menus and forms which extend over more than one screen. Equivalent to “+” in the NEXT line.
F9	Redisplays the last command or statement entered. Equivalent to entering “RESTORE-SDF-INPUT” without operands (default: INPUT=*LAST-CMD or *LAST-STMT).
F11	Execute function. Executes the current operation (command or statement). Equivalent to *EXECUTE in the NEXT line or F3 in Old mode.

Table 7: Function key assignment (Styleguide mode) in guided dialog (part 1 of 2)

*STYLE-GUIDE-MODE	
Function key	**Effect**
F12	Cancel function. Cancels the current menu (or operand form) and switches to the higher-ranking menu or, in the case of a subform, to the higher-ranking operand form. The previous specifications in the subform are discarded. Within a statement menu in guided dialog, F12 terminates the program after requesting confirmation. Equivalent to *CANCEL in the NEXT line.

Table 7: Function key assignment (Styleguide mode) in guided dialog (part 2 of 2)

Note

In Styleguide mode, the function keys K1, K3, F2, F4, F10, F11 and F13 through F24 are not supported. Pressing an unsupported key causes an error message to be displayed.
Depending on the situation, the function keys F7, F8 and F11 may not always be available.

4.2.6 Effects of the quit functions

The two figures below show how the quit functions *CANCEL, *EXIT and *EXIT-ALL work in different screen masks in system mode and in program mode. You can execute a quit function either by entering the control statement of the same name or by pressing the corresponding function key.

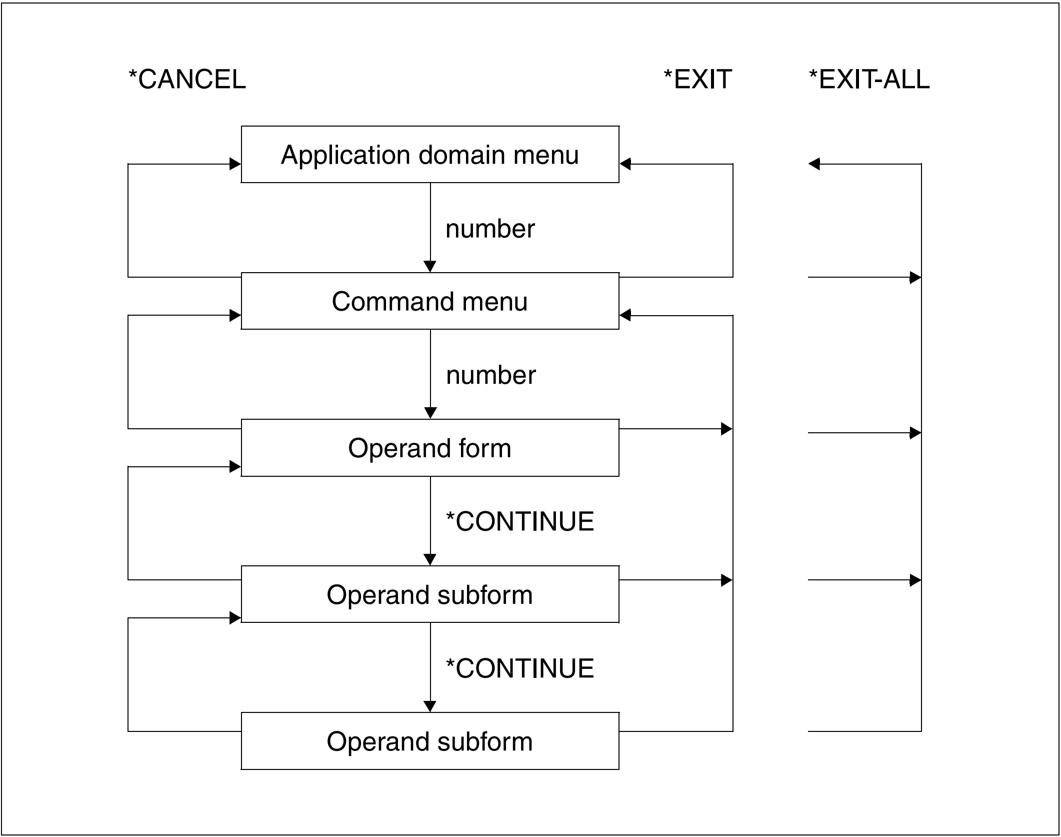


Figure 3: The quit functions *CANCEL, *EXIT and *EXIT-ALL in system mode (guided dialog)

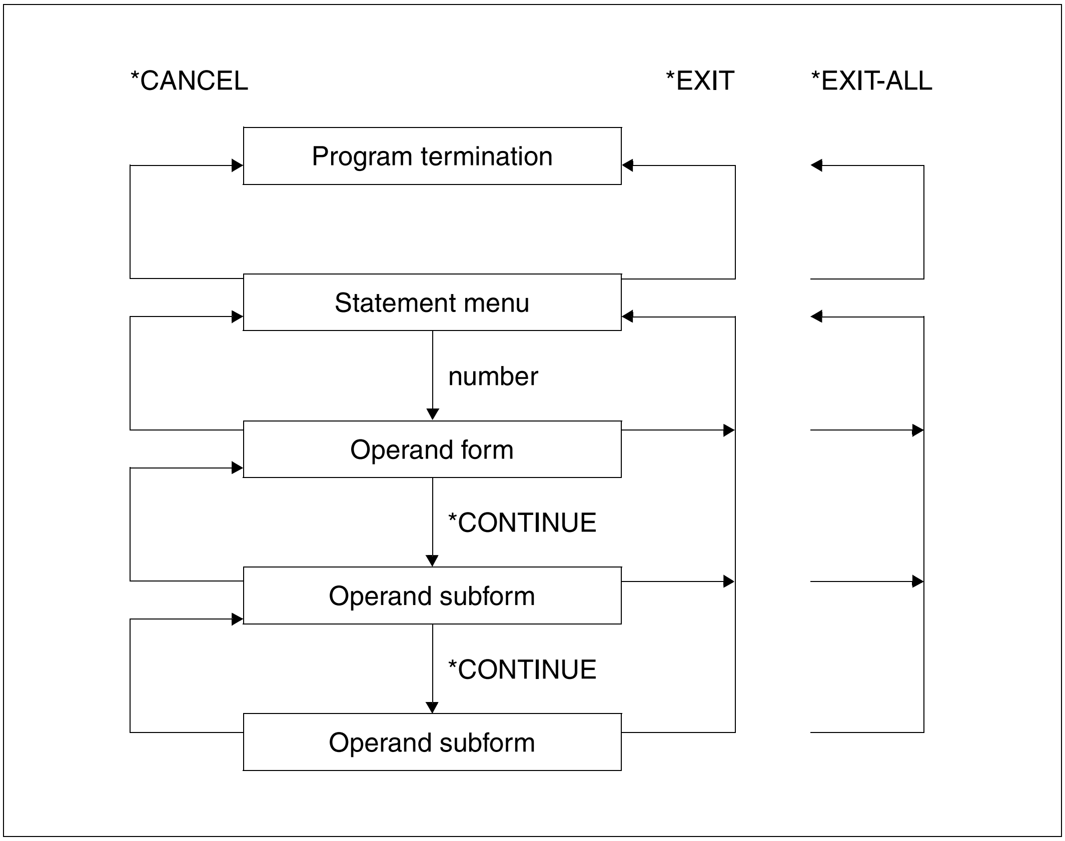


Figure 4: The quit functions *CANCEL, *EXIT and *EXIT-ALL in program mode (guided dialog)

4.2.7 Application domain menu

Every BS2000 command is assigned to at least one of the following application domains:

ACCOUNTING
ALL-COMMANDS 1)
AUTOMATED OPERATING
CONDITIONAL-JOB-CONTROL
CONSOLE-MANAGEMENT
DATA-BASE
DCAM
DEVICE
ERROR-LOGGING
FILE
FILE-GENERATION-GROUP
FILE-TRANSFER
JOB
JOB-VARIABLES
MESSAGE-PROCESSING
MULTI-CATALOG-AND-PUBSET-MGMT 2)
NETWORK-MANAGEMENT
PREVENTIVE-DIAGNOSTIC-SUPPORT
PROCEDURE
PROGRAM
PROGRAMMING-SUPPORT
RSO-SPOOL-ADMINISTRATION 3)
SDF
SECURITY-ADMINISTRATION
SPOOL 4)
SPOOL-PRINT-ADMINISTRATION
SPOOL-PRINT-SERVICES
STORAGE-MANAGEMENT
SUBSYSTEM-MANAGEMENT 5)
SYSTEM-MANAGEMENT
SYSTEM-TUNING
USER-ADMINISTRATION
UTILITIES
VM2000-VIRTUAL-MACHINE

The above domains correspond to the status of BS2000/OSD-BC V5.0A. Domains without contents (commands disabled or not implemented) are not displayed.

Comments:

1) The application domain ALL-COMMANDS contains all commands which can be input in the dialog.
2) The application domain MULTI-CATALOG-AND-PUBSET-MGMT will replace MULTI-CATALOG in the long term.
3) The application domains SPOOL-PRINT-ADMINISTRATION will replace RSO-SPOOL-ADMINISTRATION in the long term.
4) The application domains SPOOL-PRINT-SERVICES will replace SPOOL in the long term.
5) The application domain SYSTEM-MANAGEMENT will replace SUBSYSTEM-MANAGEMENT in the long term.

SDF offers the user a menu listing these application domains. The user selects the application domain relevant to his task. SDF then offers him a command menu for the selected domain from which to choose the command he requires.

SDF displays the application domain menu in command mode only. It is supplied in the following cases:

– after starting guided dialog, provided that no domain has yet been set

– after entering a question mark in unguided dialog

– after entering *DOM-MENU in the NEXT line of a menu or form

– after entering *CANCEL in the NEXT line of a command menu or after pressing the F12 key (only in Styleguide mode)

– after entering *EXIT in the NEXT line of a command menu or pressing the K1 or F3 key (depending on the FUNCTION-KEYS setting)

– after entering *EXIT-ALL in the NEXT line or pressing the F1 or F6 key (depending on the FUNCTION-KEYS setting) in guided dialog. This input or function key terminates a temporarily guided dialog.

4.2.8 Command/statement menu

For each application domain, there is a menu listing all the commands assigned to it. For each program that has an SDF interface, there is a menu listing all the permissible statements.

SDF offers the user one of these menus. The user selects the command (or statement) relevant to his task. SDF then displays an operand form for the selected command (or statement). If the command selected has no operands, it is marked “(!)” or “(EXECUTED IMMEDIATELY!)” in the command menu and is executed immediately without a form being displayed first.

The user is supplied with a command or statement menu in the following cases:

– after selecting a domain from the application domain menu

– after starting a program, provided that several statements are permitted immediately after the program start

– after entering (<domain>) in the NEXT line of a menu or form (command menu for the application domain <domain>)

- after entering "*CANCEL" in the NEXT line of an operand form or after pressing the [F12] key (in Styleguide mode)
- after entering *EXIT in the NEXT line of the operand form or after pressing the [K1] or [F3] key (depending on the FUNCTION-KEYS setting)
- after execution of a command or statement in guided dialog (command mode: menu for the last application domain set; program mode: only if several statements are permitted for the next processing step)
- in command mode after starting guided dialog if an application domain has already been set
- in program mode after starting guided dialog during program execution
- in program mode after entering a question mark in unguided dialog, provided that the next statement in the program can be chosen from various options

4.2.9 Operand form

For every command and every statement with operands there is an operand form listing the associated operands. Depending on the definition in the syntax file, the operands of a structure (see page 24) may, under minimum and medium guidance, either be integrated in the form or listed in a separate subform. If SDF specifies on the form an operand value associated with a structure for which there is a separate subform, an empty set of parentheses following the value draws attention to the existence of the subform. Even where a subform exists, the user may enter the operands of the structure in the higher-ranking form. They should be enclosed in parentheses and placed immediately after the operand value which initiates the structure.

SDF offers the user a form in which to enter the values for the operands of a command or statement. This form is preset with the default values of the optional operands. Once a form has been sent off, SDF either displays another form (subform for a structure) or causes the command or statement to be executed. The user is supplied with an operand form in the following cases:

- after selecting a command or statement in a menu
- after entering <command>? or <statement>? in unguided dialog or in the NEXT line of a form or menu
- after entering !<command>? or !<statement>? in unguided dialog or in the NEXT line of a form or menu
- if, in guided dialog, a program expects a specific statement (operand form for a statement)

- in program mode after entering a question mark in unguided dialog if the program expects a specific statement for the next processing step
- if, in unguided dialog (NO form), a request to correct an errored entry is answered with a question mark
- if an incorrect entry is made in guided dialog

If the user sends off an operand form with a question mark instead of an operand value, SDF offers the same form with additional information.

Switchover to a lower-ranking operand form occurs in the following cases:

- after entering at least one structure-initiating operand value in the form and entering *CONTINUE in the NEXT line (display of a subform)
- after entering a structure-initiating operand value in the form and entering *DOWN(<operand>) in the NEXT line of a form (switch to the subform for an individual structure)

Switchover to a higher-ranking operand form occurs in the following cases:

- after entering *UP in the NEXT line of a subform
- after entering "–" in the NEXT line of the first page of the first of a sequence of subforms (or pressing the $\boxed{\text{F7}}$ key in Styleguide mode)
- after entering *CANCEL in the NEXT line of a subform (or pressing the $\boxed{\text{F12}}$ key in Styleguide mode)

The specifications made previously in the subform are retained during switchover with *UP or "–", but in the case of switchover with *CANCEL they are discarded.

4.2.10 Special entries in the operand form

The default values preset in the input lines of the operand form can either be accepted or overwritten with user-specified values. Any value already set must be completely overwritten when entering a new operand value. Any characters not overwritten by the new value must be erased either using the LZF key or by overwriting them with blanks. If a list of operand values is entered (list-possible), the parentheses may be omitted.
Further entries are possible:

?	as an operand value calls up a help text and a display of the range of values for the relevant operand. If SDF has displayed the message "`PLEASE CORRECT THE INCORRECT OPERAND(S)`" after an incorrect input has been made, the question mark calls up detailed error messages in addition. The remainder of the line need not be deleted.
??	as an operand value supplies information on the data types permitted as an operand value. The remainder of the line need not be deleted.
!	as an operand value inserts the default value for this operand (this is important if the default value preset in the form has been overwritten). The remainder of the line need not be deleted.
^	as an operand value of a "secret" operand requests a blanked input field for confidential entry of the operand value.
([<operand>,...]	following a structure-initiating operand value displays the subform for the associated structure. It is preset with any operands entered after the open parenthesis.
([<operand>,...])	following a structure-initiating operand value suppresses the subform and inserts the default values for any operands of the structure that are not specified.
–	as the last character in an input line causes SDF to display a continuation line.
LZF key	deletes all characters in the input line from the cursor onwards (LZF = abbreviated German for "delete character string").

4.2.11 Positioning within operand forms

When an operand form is first displayed, the cursor is positioned to the beginning of the first input line. Positioning to a different line is achieved by moving the cursor as follows:

↓	The cursor moves to the same position in the following input line. Starting from the NEXT line, the cursor returns to the first input line.
↑	The cursor moves to the same position in the preceding input line. Starting from the first input line, the cursor moves to the beginning of the line. Starting from the NEXT line, the cursor moves to the beginning of the NEXT line.
⇥	The cursor moves to the beginning of the following input line, skipping any continuation lines of the current line. Starting from the NEXT line, the cursor returns to the first input line.
⇤	Starting from the beginning of an input line, the cursor moves to the beginning of the preceding input line. Starting from any other position, it moves to the beginning of the current input line.
↖	The cursor moves to the beginning of the first input line.
⌝	The cursor moves to the beginning of an input line or continuation line. Starting from the first input line, it moves to the beginning of that line. Starting from the NEXT line, it moves to the beginning of the NEXT line.
↵	The cursor moves to the beginning of the following input line or continuation line. Starting from the NEXT line, it returns to the first input line.

4.3 Input in temporarily guided dialog

While in the EXPERT or the NO form of unguided dialog, it is possible to switch to temporarily guided dialog for entering a particular command or statement. The user prompting of this dialog level is equivalent in scope to that of the minimum guidance level (see page 57). Once the entry has been made, the user returns to the original form of unguided dialog.

While in command mode in unguided dialog, the user can request an application domain menu by entering “?” and select the application domain in which he wishes to work. SDF then displays the command menu for this application domain.

While in the program mode of unguided dialog, entering “?” normally causes SDF to display a statement menu. If no more than one statement is meaningful at the time of entering “?”, SDF displays the operand form for that statement instead of the statement menu.

The operand form for the given <command> or <statement> is called up by entering “<command>?[<operand>,...]” or “<statement>?[<operand>,...]”.
The form is preset with any operand values which may have been specified and with the default values of the optional operands. If the user presses the K1 key in Old mode or the F3 key in Styleguide mode, or enters *EXIT in the NEXT line of the form, SDF displays the application domain menu (or, in program mode, the statement menu).

If <command> is a command in ISP format, SDF has three possible responses:

– The command is entered in the form <command>? and corresponds exactly to one command in SDF format. SDF displays the operand form of the corresponding command in SDF format.

– The command is entered in the form <command>? and corresponds to a number of commands in SDF format. SDF displays a menu with the corresponding commands.

– The command is entered in the form <command>?<operand>,... . SDF displays the operand form for the ISP command entered. This form comprises only the OPERANDS operand, which is preset with the operand string already entered. Further keyword or positional operands may be specified in the input field. For ISP commands, only one operand is defined, which permits the enumeration of all ISP operands (defined as <command-rest>). SDF does not analyze the input on the operand level, i.e. syntax errors are not discovered until the executing module rejects them.

While in the NO form of unguided dialog, any error in the input causes SDF to display the faulty operand and to request the user to correct the error. If the user reacts by entering “?”, SDF displays the operand form for the command (or statement) which is to be corrected.

It is possible to terminate temporarily guided dialog and return to unguided dialog by pressing either the F1 key (Old mode) or the F6 key (Styleguide mode) or entering *EXIT-ALL in the NEXT line of the menu or form.

The last input can be redisplayed using the RESTORE-SDF-INPUT command or statement or by pressing the F9 key (in Styleguide mode) (see page 196).

4.3.1 Effects of the quit functions

The two figures below show how the quit functions *CANCEL, *EXIT and *EXIT-ALL work in different screen masks in system mode and in program mode. The user can execute a quit function either by entering the control statement of the same name or by pressing the corresponding function key.

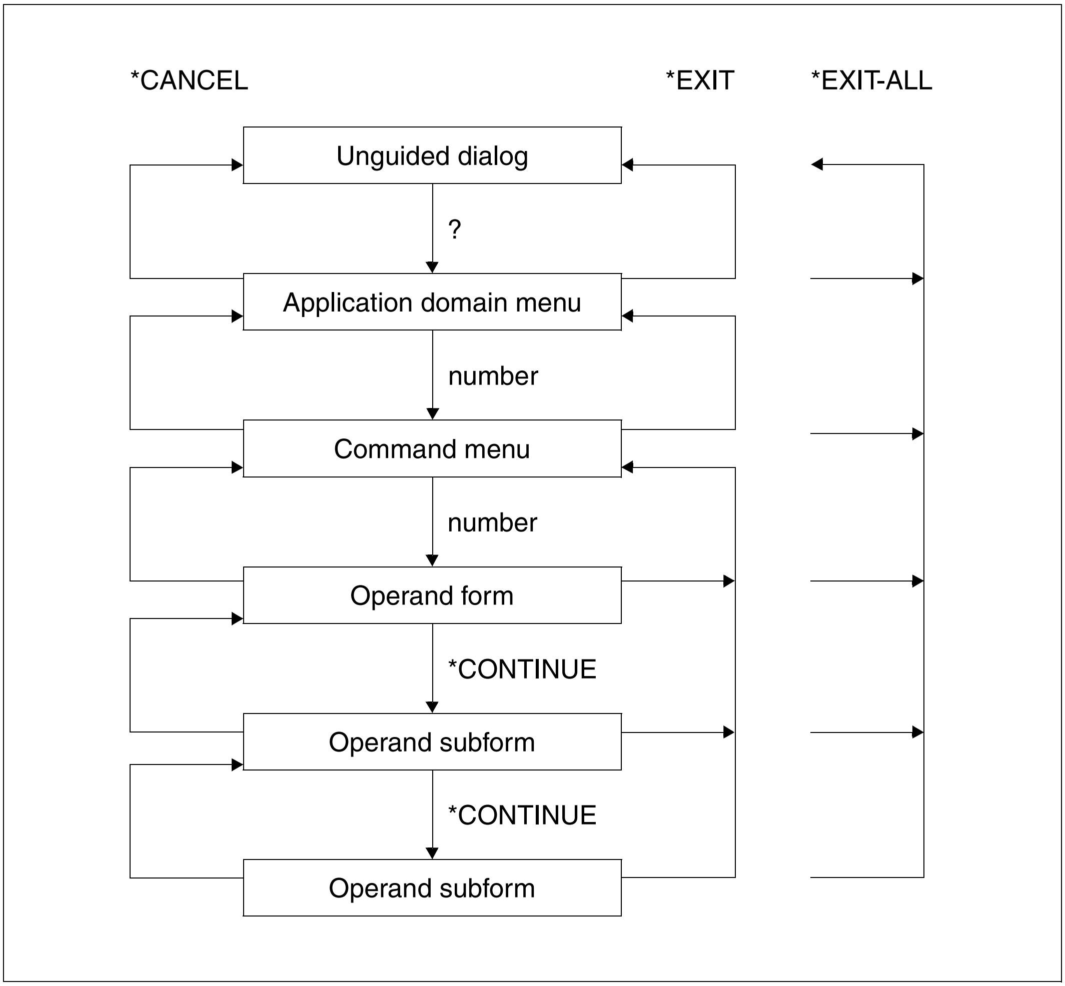


Figure 5: The quit functions *CANCEL, *EXIT and *EXIT-ALL in system mode (temporarily guided dialog)

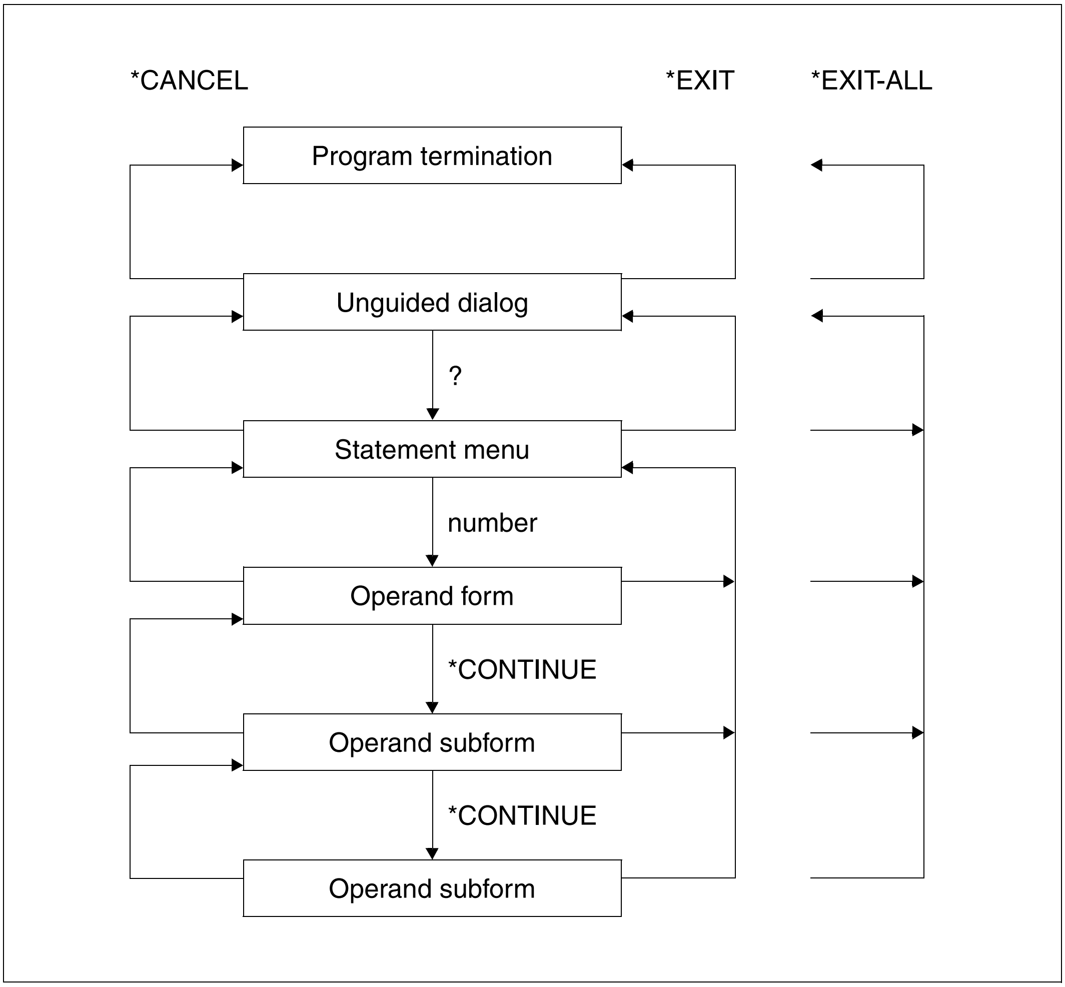


Figure 6: Effects of the functions *CANCEL, *EXIT and *EXIT-ALL in program mode (temporarily guided dialog)

4.4 Further input options

In addition to input from a data display terminal, input is also possible from procedures and in batch mode. For this purpose, commands and statements are stored in a file or in compound S variables, observing the format rules that apply to the creation of procedures or ENTER files.

4.4.1 Input from procedures

Input from procedures is the same as in unguided dialog. Letters must be entered in uppercase notation. Commands must be preceded by a slash, statements by two slashes. A command or statement may take up one or more lines. The continuation character used is the hyphen, which indicates that a continuation line is present. Each continuation line starts with a slash (or, in the case of statements, with two slashes).

Depending on the mode setting, the continuation character for commands either must be entered in column 72 exactly (Old mode) or may be entered in any column from 2 through 72 (New mode). The setting can be queried by means of the SHOW-SDF-OPTIONS command and is defined via the options *OLD-MODE and *NEW-MODE in the CONTINUATION operand of the MODIFY-SDF-OPTIONS command.
In S procedures, the line length can be defined by means of the command SET-PROCEDURE-OPTIONS INPUT-FORMAT=*FREE-RECORD-LENGTH. In this case, the continuation character may be entered in column 2 or any following column and the line length may exceed 72 characters.

In the case of statements, the continuation character may occupy any position as of column 2, and the line length can be more than 72 characters.

In non-S procedures, input records contradicting the SDF syntax description activate the spin-off mechanism; in S procedures, the SDF-P error recovery mechanism is activated. During an interactive procedure run, however, a correction dialog for errored procedure commands or statements can be initiated if the appropriate specification has been made in the PROCEDURE-DIALOG operand of the MODIFY-SDF-OPTIONS command. The current setting can be queried with the aid of the SHOW-SDF-OPTIONS command.

How to define task-specific default values is described on page 110ff.

Implemented procedures

Users can define their own commands in a user syntax file (see pages 155 and 161).
A user-defined command requires its own syntax description containing the command name, the operand names, and the permissible operand values. In this definition, the command name is linked to the name of a procedure, which is called as soon as the “new” command is entered. The syntax description defines the values to be supplied for the

procedure parameters with the SDF procedure call. Those parameter values to be supplied by the command initiator are set according to the specified operand values.
The procedure parameters are supplied in the defined manner via the operands of the command. The user is thus able to equip a procedure with an SDF interface providing all the benefits of the SDF command language. In particular all guidance levels, checking and correction of the specified operand values, information on permissible values with additional explanations, and masked input of sensitive values are possible.
The definition of user-own commands has to be performed with the aid of the software product SDF-A (see example in the “SDF-A” manual [4]).

Logon procedures

Logon processing offers the possibility of automatic execution of logon procedures. LOGON procedures can be used throughout the system or specifically for a certain user. The call can be made in a call or an include procedure in both cases (corresponds to a call with the command CALL-PROCEDURE or INCLUDE-PROCEDURE. See also the “SDF-P” manual [6]).
LOGON procedures may not contain any DO commands and may not be terminated using the ENDP-RESUME command. Furthermore, they should be uninterruptible.

Users can create their own logon procedure in the form of both a call and an include procedure for all jobs running under their user ID. An include procedure must be started before the corresponding call procedure when this is done.
The call procedure must be cataloged under the standard name $userid.SYS.SDF.LOGON.USERPROC and the include procedure under the standard name $userid.SYS.SDF.LOGON.USERINCL for the automatic call.

Systems support staff can provide system-wide logon procedures (call and include procedures) that are valid for all users (see the “SDF Management” manual [5]).
A system LOGON procedure is not started for tasks in which a group syntax file is assigned without evaluating the system syntax file.

The user’s own logon procedures are not started until termination of the system logon procedure. Command input is not possible until termination of the logon procedure(s).

LOGON procedures are ignored without a warning in the following cases:

– The procedure file is only cataloged but does not use any storage space.
– The task is an RFA task.
– The task has no privileges other than the HARDWARE-MAINTENANCE, SECURITY-ADMINISTARTION, SAT-FILE-MANAGEMENT and SAT-FILE-EVALUATION privileges.

In addition, no LOGON procedure is started for some system tasks (e.g. when initializing the system).

LOGOFF procedures

You can have the LOGOFF procedure run automatically. LOGOFF procedures can be set with system-wide scope or just for a certain user or users. The call can be made as a call procedure or include procedure in both cases (corresponding to a call with the command CALL-PROCEDURE or INCLUDE-PROCEDURE, see also the “SDF-P“ manual [6]). LOGOFF procedures may not contain DO commands and may not be terminated with the ENDP-RESUME command. Furthermore, they should be uninterruptible.

The user can create a call procedure as well as an include procedure as his user LOGOFF procedure for all jobs running under his user ID. An include procedure is started before the corresponding call procedure.
The call procedure must be cataloged under the standard name $userid.SYS.SDF.LOGOFF.USERPROC and the include procedure under the standard name $userid.SYS.SDF.LOGOFF.USERINCL for the automatic call.
If an EXIT-JOB or LOGOFF command is called in the procedure, then processing of the user LOGOFF procedure terminates and any system LOGOFF procedures defined are then started.

Systems support can make the system LOGOFF procedure (call and include procedure) available for all users by making the appropriate entry in the SDF parameter file (see the “SDF Management“ manual [5]). System LOGOFF procedures are only started after the user LOGOFF procedures defined have run. User and group syntax files are deactivated before they are started.
A system LOGOFF procedure is not started for tasks in which a group syntax file is assigned without evaluating the system syntax file.

LOGOFF procedures are ignored without a warning in the following cases:

– The procedure file is only cataloged but does not use any storage space.
– The task was canceled with CANCEL-JOB or FORCE-JOB-CANCEL
– The task is an RFA task.
– The task has no privileges other than the HARDWARE-MAINTENANCE, SECURITY-ADMINISTARTION, SAT-FILE-MANAGEMENT and SAT-FILE-EVALUATION privileges.

No LOGOFF procedure is started for some system tasks (e.g. when initializing the system).

Procedure dialog

SDF dialog guidance can be permitted in procedures, i.e. SDF handles procedure input like dialog input. This allows, for example, the correction dialog to be used if syntax errors are found in input. In addition the user can ensure the masked entry of passwords in a procedure (operand value *SECRET in the procedure).
The procedure dialog can be used if the SDF option PROCEDURE-DIALOG is set to *YES before the procedure is called or in the procedure. To use the correction dialog you must also set an appropriate guidance level (e.g. GUIDANCE= *NO).

Interruptibility of procedures

The possibility of procedure interruption via the $\boxed{\text{K2}}$ function key can be disabled by making the appropriate specification in the INTERRUPTION-ALLOWED operand of the BEGIN-PROCEDURE command (non-S procedures) or the SET-PROCEDURE-OPTIONS command (S procedures). This protects procedures from undesired interruption (e.g. permitting access to protected files within a procedure). The uninterruptibility attribute is retained in the case of procedure nesting.

Interruptibility of programs in procedures

A program that is loaded within an uninterruptible procedure is only in certain cases protected (implicitly) against interruption. A program interruption can thus be considered by the procedure as intentional. An interruption by the user who called the procedure, however, is undesirable.

On the other hand, during program execution processing statuses can occur in which no type of interruption (even from within a procedure) is desirable (e.g. processing of sensitive data). In this case the program must protect itself implicitly against interruption.

Within a procedure or a program, the following events can request an interrupt (see next page).

Control entity	**Event**	**Before event**		**After event**	
		Mode	**Input level**	**Mode**	**Input level**
Procedure	K2 key	proc	cmd	dia	cmd
Program	K2 key	proc	prog	dia	cmd
		dia	prog	dia	cmd
	BKPT macro	any	prog	same	cmd
	CMD macro	any	prog	mclp	cmd
	//HOLD-PROGRAM	any	prog	same	cmd
	//EXECUTE-SYSTEM-CMD	any	prog	mclp	cmd
	/HOLD-PROGRAM	proc	prog	proc	cmd

Meaning:		
	proc	procedure
	dia	dialog
	cmd	command input
	prog	program input
	mclp	command input via CMD macro
	any	command or program input
	same	same mode as before event occurred

The interrupt request for a program running in a procedure is thus handled as follows:

	Interrupt allowed			
	Program	**no**	**yes**	
Interrupt via	**Procedure**	**yes / no**	**no**	**yes**
$\boxed{\text{K2}}$ key		R	R	A
$\boxed{\text{K2}}$-STXIT routine		A,RE	A,RI	A
BKPT macro		A,RE	A,RI	A
CMD macro		A,RE	A,RI	A
Other macro calls		A,RE	A,RI	A
//HOLD-PROGRAM		R	C	C
//EXECUTE-SYSTEM-CMD		R	C	A
/HOLD-PROGRAM		R	C	C

Meaning:	A	The interrupt is accepted by the system.
	RE	The interrupt should be rejected by a program that is explicitly uninterruptible (CLISET).
	RI	The interrupt should be rejected by a program that is implicitly uninterruptible.
	R	The interrupt is rejected by the system.
	C	The interrupt is rejected by the system if the statements are not read from the procedure file (SYSSTMT ≠ SYSCMD).

Rejection means:

– $\boxed{\text{K2}}$ is ignored and processing is resumed.
– The command or standard statement HOLD-PROGRAM or the command BEGIN-BLOCK PROGRAM-INPUT=*MIXED-WITH-CMD generates the EOF condition in the program.
– The standard statement EXECUTE-SYSTEM-CMD is rejected and activates the spin-off mechanism at statement level.
– The standard statement HOLD-PROGRAM is rejected in all cases if it is not read from the procedure file. Otherwise, different input sources of statements and commands could lead to inconsistencies in the procedure execution.

The following functions are only supported as of BS2000/OSD-BC ≥ V2.0:

– the standard statements EXECUTE-SYSTEM-CMD and HOLD-PROGRAM
– the CLISET macro, which explicitly declares a program to be uninterruptible
– the CLIGET macro, which is used to query whether the program-calling procedure is uninterruptible.

4.4.2 Input in batch mode

Input in batch mode is the same as in unguided dialog. Letters must be entered in uppercase notation. Commands must be preceded by a slash, statements by two slashes. A command or statement may take up one or more lines. The continuation character used is the hyphen, which indicates that a continuation line is present. Each continuation line starts with a slash (or, in the case of statements, with two slashes).

Depending on the setting of the SDF option CONTINUATION, the continuation character for commands must either be entered in column 72 exactly (Old mode) or may be entered in any column from 2 through 72 (New mode). The setting can be queried by means of the SHOW-SDF-OPTIONS command and modified via the appropriate specification for the CONTINUATION operand of the MODIFY-SDF-OPTIONS command.
In the case of statements, the continuation character may occupy any position as of column 2, and the line length can be more than 72 characters.
Input records contradicting the SDF syntax description activate the spin-off mechanism, since a correction dialog is not possible in batch mode.

It is not possible to define task-specific default values.

4.5 Replacing expressions in the input

The value of a procedure parameter, an S variable expression or a job variable can substitute a command/statement portion. The expression to be substituted is marked by a “&” character, followed immediately by the name of the procedure parameter, job variable or S variable expression enclosed in parentheses. SDF substitutes the actual value for the expression prior to execution of the command/statement and performs a syntax check on the resulting input. This substitution of expressions is permissible in unguided dialog, in procedures and in batch mode. In (temporarily) guided dialog, it is restricted to the NEXT line and to input for operand values.

Substituting procedure parameters

In non-S procedures, expressions of the form “&parameter”, when used in commands, are replaced by the value assigned to *parameter* in the command BEGIN-PROCEDURE or CALL-PROCEDURE or during prompting. When used in input data (read from SYSDTA), such expressions are replaced only if an escape character (#, @, & or *) has been defined in the ESCAPE-CHARACTER operand of the BEGIN-PROCEDURE command and the expression begins with this character.
When used in statements (read from SYSSTMT), such expressions are currently replaced as in commands. It is, however, advisable to define ESCAPE-CHARACTER=’&’, because the handling of such expressions may be modified.

The following restrictions apply to substituting procedure parameters:

– No substitution is possible within CJC command sequences.
– In procedures or ENTER files, procedure parameters cannot replace the slash introducing commands or the two slashes introducing statements, the period introducing non-S labels, the semicolon separating commands or the continuation character.
– Expressions cannot be nested.
– A double “&” or escape character inhibits substitution, the second “&” or escape character is ignored.

Substituting job variables

This function requires the chargeable subsystem JV to be loaded.

Expressions to be replaced by job variables are specified as follows:

– directly via the job variable name in the form “`&(jv-name)`”
– indirectly via the job variable link name in the form “`&(*jv-link)`” after the link name has been assigned to the job variable by means of the command `SET-JV-LINK LINK-NAME = jv-link, JV-NAME = jv-name`.

The following restrictions apply to substituting job variables:

– An expression can be replaced only by a job variable in its full length.
– Read access must have been granted to the job variable value to be substituted for the expression, otherwise the input will be rejected as a syntax error.
– No substitution is possible within CJC command sequences.
– Job variables cannot be substituted for input data. SDF treats statements intended for programs with an SDF interface like commands and not like input data.
– In procedures or ENTER files, job variables cannot replace the slash introducing commands or the two slashes introducing statements, the period introducing non-S labels, the semicolon separating commands or the continuation character.
– Job variables cannot be used as procedure parameters. This restriction can be circumvented, for instance, by using a link name (see example 2 on page 92).
– Expressions cannot be nested.
– In the dialog and in S procedures, job variable substitution in the manner outlined above is effected only if no identical S variable or builtin function is known. This mechanism can be replaced, however, by the builtin function JV( ), in which case the appropriate entry would take the form “`&(JV(JV-NAME=<c-string 1..54>))`”. See the builtin function JV( ) in the “SDF-P” manual [6] or in the “Commands, volume 6” manual [2]. If the job variable name contains the catalog and/or user ID, only job variable substitution is possible.

Substituting S variable expressions

In dialog mode and in S procedures, expressions of the form “&(expression)”, when used in commands, are replaced by the value of *expression*, where *expression* may be the name of an S variable, a builtin function or a valid S variable expression. If the expression is to be substituted by an S variable whose name does not contain any period, it may also have the form “&s-variable” (i.e. without parentheses) .
Procedure parameters of an S procedure are S variables and are replaced in expressions also.

When used in input data (read from SYSDTA), such expressions are replaced only if an escape character (#, @, & or *) has been defined in the DATA-ESCAPE-CHARACTER operand of the SET-PROCEDURE-OPTIONS command and the expression begins with this character.
When used in statements (read from SYSSTMT), such expressions are currently replaced as in commands.

Expressions may be nested.

The following restrictions apply to substituting S variable expressions:

- Control flow commands cannot be generated.
- The S variable expression to be replaced is converted to type STRING.
- S labels cannot be generated.
- No substitution is possible within CJC command sequences.
- In procedures or ENTER files, S variable expressions cannot replace the slash introducing commands or the two slashes introducing statements, the period introducing non-S labels, the semicolon separating commands or the continuation character.
- A double “&” or escape character inhibits substitution, the second “&” or escape character is ignored.

Examples

1. Substituting job variables in dialog mode:

```
/cre-jv jv=cmd                         ——————————————————————————————— (1)

/mod-jv jv=cmd,set-value='SHOW-FILE-ATTR'  ————————————————— (2)

/&(cmd)                        ——————————————————————————————— (3)
%        3 :2OSG:$USER1.ALT.SYS.LOGON.USERPROC.X1
%       51 :2OSG:$USER1.ALT.SYSSDF.USER.EXAMPLE.1
%       21 :2OSG:$USER1.DATEI.1
%       48 :2OSG:$USER1.DATEI.2
%       84 :2OSG:$USER1.DATEI.3
%       66 :2OSG:$USER1.OUT.SORT1-2
%        3 :2OSG:$USER1.PROC.JV
%:2OSG: PUBLIC:     7 FILES RES=      276 FRE=       39 REL=      21 PAGES

/mod-jv jv-=egon,set-value='-FILE-ATTR F-NAME=PROC.'  —————————— (4)

/set-jv-link link-name=walter,jv-name=egon  ——————————————— (5)

/sh&(*walter)
%        3 :2OSG:$USER1.PROC.JV  ——————————————————————————————— (6)
%:2OSG: PUBLIC:     1 FILE  RES=        3 FRE=        2 REL=       0 PAGES
```

(1) The job variable name ’CMD’ is declared.

(2) The job variable CMD is assigned the value ’SHOW-FILE-ATTR’ (abbreviated form of the SHOW-FILE-ATTRIBUTES command).

(3) After the command has been sent off, the variable string is replaced with the command defined in the job variable, and the command is executed.

(4) The value of job variable CMD is changed. It now contains only part of the command name (’-FILE-ATTRIBUTES’) and the partially qualified file name ’PROC.’.

(5) Job variable CMD is assigned the link name ’WALTER’.

(6) After the command has been sent off, the variable string is replaced with the command portion assigned to the job variable, and the command is executed. The reference to the job variable is established via the link name.

2. Substituting job variables and S variables in dialog mode:

```
/sh-jv cmd    ----------------------------------------------------------- (1)
%SHOW-FILE-ATTR
/&(cmd) proc.jv    ------------------------------------------------------ (2)
%         3 :2OSG:$USER1.PROC.JV
%:2OSG: PUBLIC:      1 FILE  RES=        3 FRE=        2 REL=        0 PAGES
/cmd='PRINT-DOCUMENT'  ------------------------------------------------- (3)
/sh-var cmd
CMD = PRINT-DOCUMENT
/&(cmd) proc.jv    ------------------------------------------------------ (4)
%  SCP0810 SPOOLOUT FOR FILE ':2OSG:$USER1.PROC.JV' ACCEPTED. TSN: '1FAL', SPOOL
OUT-NAME: 'SDFTEST', MONJV: '*NONE'
%  SCP1025 PRINT JOB ACCEPTED BY SERVER 'GH5090Y0' WITH TSN '5BXC'
/&(:2osg:cmd) proc.jv   ------------------------------------------------ (5)
%         3 :2OSG:$USER1.PROC.JV
%:2OSG: PUBLIC:      1 FILE  RES=        3 FRE=        2 REL=        0 PAGES
/&(jv(jv-name='CMD')) proc.jv   ---------------------------------------- (6)
%         3 :2OSG:$USER1.PROC.JV
%:2OSG: PUBLIC:      1 FILE  RES=        3 FRE=        2 REL=        0 PAGES
/
```

(1) The job variable `CMD` has the value `SHOW-FILE-ATTR`.

(2) The command name from the job variable `CMD` is substituted i.e. SHOW-FILE-ATTRIBUTES is executed for the file `PROC.JV`.

(3) The S variable `CMD` is created implicitly by being assigned the value `PRINT-DOCUMENT`.

(4) The expression `&(CMD)` in the input is now replaced by the contents of the S variable `CMD`, i.e. the PRINT-DOCUMENT command is executed for the file `PROC.JV`.

(5) The name of the job variable `CMD` is prefixed by the catalog ID in order to retain its contents during substitution.

(6) Job variable substitution can alternatively be achieved using the builtin function JV( ).

3. Submitting the name of the job variable to be substituted as procedure parameter of a non-S procedure:

```
/BEGIN-PROC PAR=*YES(PROC-PAR=(&PARAM1))    ——————————————————————  (1)
    .
    .
    .
/SET-JV-LINK LINK-NAME=PARAM1,JV-NAME=&PARAM1    ———————————————————  (2)
/&(*PARAM1) FILE-NAME=LST.JOB     ——————————————————————————————————  (3)

    .
    .
    .
/END-PROC
```

(1) The job variable to be specified via procedure parameter PARAM1 is to contain the current command to be executed.
As the entry “&(&PARAM1)” is illegal, a link name will have to be assigned.
See Example 4 (3) for the optional use of a nested expression.

(2) The current job variable name is inserted for procedure parameter PARAM1 and assigned the link name PARAM1.

(3) The contents of the declared job variable are substituted for the link name PARAM1. For instance, if the job variable value is `PRINT-DOCUMENT DOCUMENT-FORMAT=*TEXT(LINE-SPACING=*BY-EBCDIC-CONTROL), LAYOUT-CONTROL=*PAR(ROTATION=90, LEFT-MARGIN=10)`, the file LST.JOB is printed as specified. If the job variable value changes to `SHOW-FILE-ATTRIBUTES INFORMATION=*PAR(HISTORY=*YES,SECURITY=*YES)`, the requested file attributes of the file LST.JOB are displayed.

4. Substituting job variables and S variables in an S procedure, the job variable name being submitted as a procedure parameter:

Contents of procedure file DO.JVTEST:

```
/        SET-PROC-OPT  JV-REPLACE=*AFTER-BUILTIN
/        DECL-PAR      JV-1(INIT=*PROMPT)
/        &(JV(JV-NAME=JV-1))   FILE-NAME=LST.JOB
/FEHL:   IF-BLOCK-ERROR
/            WRITE-TEXT  C'** Error &MC **'
/        ELSE
/            WRITE-TEXT  C'** Command&(&(JV-1)) executed **'
/        END-IF
/ENDE:   EXIT-PROC
```

Procedure execution:

```
/sh-jv jv(cmd)  ------------------------------------------------------------ (1)
%SHOW-FILE-ATTR
/cal-proc do.jvtest,log=*yes
%          1  1 /SET-PROC-OPT  JV-REPLACE=*AFTER-BUILTIN
%          2  1 /DECL-PAR      JV-1(INIT=*PROMPT)
%JV-1: cmd   --------------------------------------------------------------- (2)
%          3  1 /SHOW-FILE-ATTR   FILE-NAME=LST.JOB  ------------------------- (3)
%       3 :N:$USER0001.LST.JOB
%:N:    PUBLIC:      1 FILE  RES=        3  FREE=        3  REL=        3 PAGES
%          4  1 /FEHL:
%          4  1 /   IF-BLOCK-ERROR
%          6  1 /ELSE
%          7  1 /WRITE-TEXT  C'** Command SHOW-FILE-ATTR executed **'
** Command SHOW-FILE-ATTR executed **  ------------------------------------ (4)
%          8  1 /END-IF
%          9  1 /ENDE:
%          9  1 /   EXIT-PROC
/
```

(1) `SHOW-FILE-ATTR` is displayed as the contents of job variable *CMD*.

(2) After invoking the procedure `DO.JVTEST`, the procedure parameter `JV-1` is prompted and assigned the value `CMD`. The expression is replaced by the job variable value as ascertained by the builtin function JV(). The desired job variable name is submitted to the builtin function via S variable `JV-1`.

(3) The SHOW-FILE-ATTRIBUTES command is executed.

(4) A nested expression is used at this point:
A job variable is to be substituted, its name being derived from the expression to be substituted for S variable `JV-1`. As a result, the contents of job variable `CMD` are substituted. This is possible only if there is no S variable or builtin function of the same name and if job variable substitution has been specified explicitly in SET-PROCEDURE-OPTION.

4.6 Input compression

SDF offers the possibility of compressing the input of commands and statements in dialog or batch mode.

It should be noted, however, that an abbreviation which is unique today might be ambiguous in a functionally extended future BS2000 version. Users should therefore restrict the use of abbreviations in automated command sequences.

4.6.1 Abbreviation of names

Basically, all names used (keywords) may be abbreviated:

– command/statement names
– operand names
– keyword values

Names can be abbreviated as follows:

– In compound names (partial names linked by a hyphen), portions can be omitted from right to left, together with the respective hyphens.
– Within a partial name or a simple name, characters can be omitted from right to left.
– A leading asterisk does not belong to the name it introduces; it is merely used to distinguish a constant from the variable operand value whose range includes the string of the constant value. The asterisk alone, even if it is unique, does not represent a valid abbreviation.
– As of SDF V4.0A, keyword values in guided dialog and in the syntax representation are always indicated by a leading asterisk. The leading asterisk of a keyword value can be omitted if no alternative variable operand value is possible whose value range contains the name of the keyword value. This abbreviation option can be restricted by means of extensions in subsequent versions. For reasons of compatibility, operand values which were previously written without a leading asterisk are still accepted without the asterisk.
– As of SDF V4.1A, the name or partial name of a keyword value can also contain a period (e.g. *V4.5 or *OSD-V5.0). The period is part of the (partial) name. If the name is abbreviated, the period must not be at the end of it.

For SDF to be able to interpret the abbreviated names correctly, the selected abbreviations must be unique in their immediate syntax environment. However, the syntax file may contain a minimum abbreviation for particular names; in this case, SDF will not accept any shorter input even if it would be unique.

Unique assignment is defined as follows:

– a command name is unique within all valid command names.
  If a partial name of a command is specified in full, this command differs from a second command in which the specified partial name is an abbreviation of the same part of the name. With, for example, the commands START-C-COMPILER and START-COBOL-COMPILER, the input START-C-COMP can only refer to the command START-C-COMPILER.
– a statement name is unique within all valid statements of a loaded program.
– an operand name is unique within all valid operands of the specified command or statement on the same structure level (for an operand name in a lower-ranking structure, only the valid operand names of this structure are considered)
– a keyword value is unique within the set of all possible values for the specified operand.

For example, the input `/MOD-SDF-OPT SYN-F=*NONE,GUI=*MIN` is a possible abbreviation of `/MODIFY-SDF-OPTIONS SYNTAX-FILE=*NONE, GUIDANCE=*MINIMUM`

The user manuals contain “guaranteed” abbreviations (emphasized by means of bold print in the text); these are not necessarily the shortest possible versions, but they retain the basic meaning and will remain unique on a long-term basis.
This cannot be ensured for any of the other abbreviations. Procedures should therefore contain only unabbreviated names, or guaranteed abbreviations, which also greatly enhances the clarity of the procedure.

In addition to the command and statement names, the manuals may use aliases, which are guaranteed in the long term. An alias comprises no more than 8 characters (A...Z), which are derived from the command or statement name. An alias cannot be shortened.
Example: MDSDFO instead of MODIFY-SDF-OPTIONS

The names listed in the manuals are also defined in the syntax files as standard names. These standard names will continue to be accepted even if the command names have been changed, albeit only in the unabbreviated form. For example, if the command name CREATE-FILE were changed to GENERATE-FILE, the entry /CREATE-FILE would still be accepted, but /CR-F would be rejected. For this reason, if procedures are to be completely immune to the renaming of commands, all names should be specified in unabbreviated form.

4.6.2 Default values

Most operands are optional, i.e. they already have a default value which is used for command/statement execution if no explicit specification is entered by the user.
The operand values *UNCHANGED and *CURRENT stand for the existing setting, i.e. the present value is taken over.

Since only operand values containing no default settings need to be specified explicitly, input can often be dramatically reduced.

If operands do not contain any or the desired default values, users can define their own task-specific defaults for entry in the dialog (see section “Task-specific default values” on page 110).

For example, the entry `MOD-SDF-OPT SYN=*N,GUID=*MIN` is a possible abbreviation of:

```
MOD-SDF-OPT SYN=*NONE,GUID=*MIN,LOG=*UNCH,UT=*UNCH, PROC=*UNCH,
CONT=*UNCH, MENU=*UNCH,MODE=*UNCH,DEFAULT-PROG=*UNCH, FUNCTION-
KEYS=*UNCH, INPUT-HISTORY = *UNCH
```

Note

The default value of an operand should not be confused with the operand value *STD. The operand value *STD need not be the default value. The meaning of *STD is given individually in each operand description. *STD can be, for example, a value set at system installation (e.g. SPACE=*STD in the CREATE-FILE command) or a setting dependent upon the task mode (e.g. DIALOG-CONTROL=*STD in the DELETE-FILE command).

4.6.3 Positional operands

Any operand may be specified either as a keyword operand or as a positional operand. When keyword operands are entered, the operand name and the desired value are specified together in the format <operandname>=<operandvalue>.
When positional operands are entered, only the operand value is specified; correct assignment is ensured via its position in the input stream as prescribed by the command/statement definition.
The following should be noted when specifying positional operands:

– Whenever an operand preceding a positional operand is omitted, a comma must be entered instead.

– If an operand is entered as a keyword operand, no positional operands may be entered at the same structure level.

For example, the input `MOD-SDF-OPT *NONE,*MIN` is a possible abbreviation of `MOD-SDF-OPT SYN-F=*NONE,GUID=*MIN`.

It cannot be fully ruled out that an operand position will change in the event of a version changeover. For this reason, only keyword operands should be used in procedures.

4.6.4 Structures

The specification of structures offers the following options for input compression:

STRUCTURE-IMPLICIT notation

Specification of the structure-initiating operand is omitted and the subordinate operand is entered outside the structure parentheses.
The prerequisite for this is generally that the subordinate operand is unique with respect to the entire command/statement or to a higher-ranking structure. Operands for which the STRUCTURE-IMPLICIT notation is guaranteed in the long term are listed explicitly in the corresponding command or statement description.

Example

```
SHOW-FILE-ATTR ACCESS-METHOD=*ISAM
```

is the abbreviated notation for

```
SHOW-FILE-ATTR SEL=*BY-ATTR(ACCESS-METHOD=*ISAM)
```

Note

In many cases in which the STRUCTURE-IMPLICIT notation is not possible, “flat” notation as outlined below can be used.

Flat notation

The structure-initiating operand is specified. The subordinate operand, however, is entered outside the structure parentheses.
The subordinate operand need not be unique with respect to the entire command/statement, but it must not occur in more than one active structure.
Note that a structure is activated not only explicitly but also implicitly (via the default value, if an operand specification is omitted).

Flat notation is not guaranteed in the long term!

Example

```
CRE-FILE FILE1,SUP=*PRIV-DISK,VOL=ABC123,DEV-TYPE=D3475
```

is the abbreviated notation for

```
CRE-FILE FILE1,SUP=*PRIV-DISK(VOL=ABC123,DEV-TYPE=D3475)
```

Notation with NULL-ABBREVIATION=*YES

The subordinate operand is entered within the structure parentheses, but the structure-initiating operand value is reduced to a null string, i.e. omitted. The prerequisite for this is that the structure-initiating operand value has been defined with the attribute NULL-ABBREVIATION=*YES in the syntax file. This attribute can be assigned only once within the set of possible operand values (if two or more structure-initiating operands are admitted; see the “SDF-A” manual [4]).

In the syntax representation, operand values for which the attribute NULL-ABBREVIATION is guaranteed on a long-term basis are enclosed in square brackets (see section “SDF syntax representation” on page 29).

Example:

```
MOD-FILE-ATTR FILE1,PROTECTION=(ACCESS=*READ)
```

is the abbreviated notation for

```
MOD-FILE-ATTR FILE1,PROTECTION=*PARAMETERS(ACCESS=*READ)
```

4.7 Logging

Logging on SYSLST of the execution of an interactive job is specified by means of the LOGGING operand of the SET-LOGON-PARAMETERS (for batch jobs: ENTER-JOB) or MODIFY-JOB-OPTIONS command. If logging has been activated, the scope of SDF input/output logging is set via the SDF parameters LOGGING and MENU-LOGGING. This setting can be changed by explicitly specifying the desired values in the respective operands of the MODIFY-SDF-OPTIONS command.

4.7.1 LOGGING parameter setting

If LOGGING=*INVARIANT-FORM is specified, logging includes:

– all names in the form given in the manuals

– all operands appearing in the input, together with their names and the specified values

– all optional operands implicitly contained in the input, together with their default values

– system-specific command/statement definitions rather than user-specific definitions

– only the end result (accepted by SDF) of a correction dialog; inputs made in guided dialog are chained to form a string.

Operand values defined as SECRET as well as masked inputs are replaced with “P” in the log. INVARIANT-FORM produces a log that can easily be used to reconstruct job execution.

If LOGGING=*ACCEPTED-FORM is specified, logging includes:

– all names in their unabbreviated form; if SDF-A was used to change a name in the syntax file, the new name will be logged and not the one given in the manuals

– all operands appearing in the input, together with their names and the specified values

– only the end result (accepted by SDF) of a correction dialog; inputs made in guided dialog are chained to form a string.

Operand values defined as SECRET as well as masked inputs are replaced with “P” in the log. ACCEPTED-FORM produces a log that can still be used to reconstruct job execution.

If LOGGING=*INPUT-FORM is specified, the input strings in unguided dialog are fully logged and SECRET operands are masked out. In guided dialog, or in a correction dialog, logging is as for ACCEPTED-FORM.

4.7.2 MENU-LOGGING parameter setting

If *YES (rather than *NO) is entered, all output screens (menus and operand forms) are logged on a 1:1 basis. Note that in this case the amount of data records to be logged on SYSLST can become very extensive.

4.7.3 Restrictions

In the following cases the user cannot influence logging:

– Logging has been prohibited in the command definition. Whether and how logging takes place depends on the executing module.

– The command/statement contains operands defined as SECRET (e.g. passwords). If the entry for a secret operand can be uniquely assigned to that operand, “P” is inserted as the operand value in the log. If no unique assignment is possible, only the command/ statement name appears in the log. “No unique assignment” means that the command/ statement name is recognized but subsequent operands contain a syntax error.

– The command/statement name cannot be recognized. In this case, only the errored command/statement name appears in the log because the entry may contain details of secret operands.

– Procedures cannot be logged unless this has been permitted within the procedure. An additional restriction for S procedures is that they are not logged unless the caller also possesses an authorization for read access to the procedure file.

4.8 Selection menu for commands and statements

Commands can be selected via an application domain menu which contains all commands on a particular subject. Within a program the user can select from the statement menu. Another option is using wildcards or abbreviations which are not unique in command or statement names.

4.8.1 Wildcards in command or statement names

Entering a command or statement name with wildcards followed by a question mark produces a selection menu containing all commands or statements which correspond to the specified search pattern. The selection menu is also output if precisely one command or statement is matched. If there is no matching command or statement, an error message will appear.
The selection menu only exists until the command or statement is executed, after which the user is returned to the original input mode. A selection menu with the same contents can be requested again using the same search pattern.

You can use wildcards to find a command or statement without knowing its exact name. You can also request a selection menu based on your own selection criteria. For example, the input ”*job*?” produces all commands whose name contains the string JOB.

Bear in mind the following when using wildcards:
– Any characters which follow the question mark are ignored.
– The wildcard ”?” is not recognized. Instead it will be interpreted as a concluding question mark, i.e. any characters after it will be ignored.
– A search pattern for a command or statement name must not exceed 30 characters in length.
– The wildcard ”/” at the beginning of a command name is interpreted as a command prompt.
– The string ”//” at the beginning of a statement name is interpreted as a statement prompt.
– Abbreviations within a search pattern are not possible. For example, the input ”m-sdf-o*?” does not produce the command MODIFY-SDF-OPTIONS. The input required would be ”m*-sdf-o*?”.

Quitting the selection menu with *EXIT, *EXIT-ALL, *CANCEL or the corresponding function keys in guided dialog effect a return to the application domain menu or statement menu. A temporary dialog is terminated.

4.8.2 Ambiguous abbreviation of command and statement names

Entering an abbreviated command or statement name followed by a question mark produces a selection menu containing all possible matching commands or statements. Any characters after the question mark are ignored.
If the abbreviation matches exactly one command or statement, the operand form is displayed, and characters after the question mark are taken over as operand specifications.
If there is no matching command or statement, an error message is displayed.

Quitting the selection menu with *EXIT, *EXIT-ALL, *CANCEL or the corresponding function keys in guided dialog effects a return to the application domain menu or statement menu. A temporary dialog is terminated.

4.9 Reusing previous inputs

SDF saves syntactically correct commands or statements locally for each task. The INPUT-HISTORY operand in the MODIFY-SDF-OPTIONS command activates/ deactivates or resets the input buffer. It also defines the maximum number of inputs that can be saved. Each time this maximum is reached, the oldest input is deleted. The saved inputs are automatically numbered (input serial number).
Inputs in guided dialog are saved in ACCEPTED form, while those in unguided dialog are saved in INPUT form. The commands or standard statements SHOW-INPUT-HISTORY and RESTORE-SDF-INPUT are not saved. ISP commands are only saved if PASSWORD-PROTECTION has been set to *NO.

The command (or standard statement) SHOW-INPUT-HISTORY shows which inputs are stored.
The RESTORE-SDF-INPUT command redisplays a specified stored input on the screen. The user can then use the displayed command or statement again as input, either as it is or with modifications. This saves having to type the whole command or statement again.

INPUT-HISTORY=*OFF deactivates the input buffer. Stored inputs are retained.
If INPUT-HISTORY=*ON is entered, inputs are saved again.
INPUT-HISTORY=*RESET resets the input buffer, i.e. saved inputs are deleted and subsequent inputs are stored.

Protecting “secret” operands

Values specified for “secret” operands which match neither the default value nor a value defined via SECRET=*NO are stored in the input buffer with “^”.
In unguided dialog when these values are displayed again via RESTORE-SDF-INPUT, the user can do one of the following:

– send off the command/statement unchanged. In this case, SDF displays a blanked input field for each secret operand where the user can enter the desired value.
– delete the “^” and insert the desired value directly before sending off the command/statement.

Values specified for operands which are not “secret” are stored in the input buffer in plaintext. In individual cases these inputs can contain information that the user considers sensitive (e.g. procedure parameters). The following steps will prevent such inputs from being displayed again via SHOW-INPUT-HISTORY or RESTORE-SDF-INPUT:

Before entering sensitive input, the input buffer must be deactivated and then activated again. If the inputs have already been saved, the input buffer can be reset with *RESET, but it must be remembered that this will delete all saved inputs.

Removing protection of “secret” operands

The protection of “secret” operands can only be removed in interactive mode by entering INPUT-HISTORY=*ON(PASSWORD-PROTECTION=*NO). Thereafter, values specified for “secret” operands are stored in the input buffer in plaintext. ISP commands are then also stored in the input buffer.

Under this setting, passwords are displayed on the screen in plaintext with the SHOW-INPUT-HISTORY or RESTORE-SDF-INPUT command, which means that they are visible to unauthorized users. You should therefore ensure that whenever you leave your terminal no unauthorized users can output the contents of the input buffer. If your terminal does not possess any appropriate security mechanisms (e.g. chipcard terminal), you should at least delete the input buffer before you leave.

4.10 Test mode

Test mode can be activated/deactivated by specifying *TEST or *EXECUTE in the MODE operand of the MODIFY-SDF-OPTIONS command. When the test mode has been activated, the subsequent commands are subjected to a syntax check by SDF, but not executed. In addition you can define in the lower-ranking operand CHECK-PRIVILEGES whether the privileges of the task are to be taken into account when the syntax is checked. The only commands executed are MODIFY-SDF-OPTIONS and SHOW-SDF-OPTIONS, which are used to change the SDF settings during testing or to deactivate the test mode, and to obtain information on the current settings.

Note

The chargeable subsystem SDF-P offers debuggers suitable for S procedures (see the “SDF-P” manual [6]). The test mode described here is supported for S procedures subject to the following restrictions:
Inputs are always analyzed by SDF-P first. Statements cannot be tested. SDF-P control flow commands are executed. This may lead to errors since commands setting or declaring S variables, for instance, are not executed in test mode.

Job variables, S variables and procedure parameters are substituted prior to syntax analysis.

In procedure/batch mode, input lines without a leading “/” are regarded as data records and ignored. A data record or a sequence of data records is enclosed between messages CMD0091 and CMD0092, which are used to highlight input records that have not been analyzed.

Specification of a program name in the DEFAULT-PROGRAM-NAME operand of the MODIFY-SDF-OPTIONS command causes the syntax analysis to include statements to a program with an SDF interface. This works only in procedure/batch mode if test mode has been activated.
Input lines with leading “//” are then interpreted as program statements and likewise subjected to a syntax check. This syntax check is performed in accordance with the statements defined for the specified program name in the syntax file (**not** in accordance with the file name used in the START-PROGRAM command).
If statements of several programs are to be evaluated, the DEFAULT-PROGRAM-NAME setting has to be changed accordingly in front of each statement block.
DEFAULT-PROGRAM-NAME=*NONE can be used to restrict syntax analysis to commands again.

Procedure interruption with the $\boxed{\text{K2}}$ key is performed also in test mode. A return with the RESUME-PROCEDURE command is not possible in test mode.
Upon termination of a procedure in test mode, message CMD0093 informs the user that test mode is still active.

Example:

Contents of procedure file “PROC.SELECT”:

```
/BEG-PROC        LOG=*ALL -
/                 ,PAR=*YES(      -
/                        PROC-PAR=(&MODE=*EXEC,-
/                                  &TESTPROGRAM=PERCON,-
/                                  &OUTFILE, &ELEM)-
/                       ,ESC-CHAR=C'&' -
/                        )
/ASS-SYSDTA        TO=*SYSCMD
/MOD-SDF-OPT       MODE=&MODE -
/                 ,DEFAULT-PROG-NAME=&TESTPROGRAM -
/                 ,LOG=*INPUT-FORM
/SHOW-SDF-OPT      INF=*USER
/SHOW-FILE-ATTR    F-NAME=&(JV.INFILE)
/START-PERCON
//SHOW-SDF-OPT
//ASS-INPUT-F      FILE=*DISK-F(NAME=&(JV.INFILE))
//ASS-OUT-F        FILE=*DISK-F(NAME=&OUTFILE)
//SEL-INPUT-REC    COND= ((20,4)=NUMERIC AND (25,5)=ALPHA)
//START-CONV
//END
/SHOW-FILE-ATTR    F-NAME=&OUTFILE
/MOD-JOB-SW        ON=(1)
/START-PROG        FROM-FILE=$LMS
LIB FILE=USER.LIB,USAGE=BOTH
TOCX *
ADDX &OUTFILE>&ELEM
END
/SET-JOB-STEP
/MOD-JOB-SW        OFF=(1)
/MODIFY-SDF-OPT    MODE=*EXEC -
/                 ,DEFAULT-PROG-NAME=*NONE
/END-PROC
```

The procedure PROC.SELECT uses the PERCON utility routine to select data records which satisfy certain criteria (one field may only contain digits, another one must not contain any digits) from the file TEST (= value of the job variable JV.INFILE) and writes these records to a file whose name is specified via procedure parameter &OUTFILE. The result file is then incorporated in the USER.LIB library by means of the LMS utility routine. The name of the library member is queried via procedure parameter &ELEM (the members already existing are displayed beforehand).

Tracer listing:

```
/clp proc.select,proc-par=(mode=*test)  ——————————————————————————————  (1)
%/BEGIN-PROC        LOG=*ALL                 ,PAR=*YES(
     PROC-PAR=(&MODE=*EXEC,&TESTPROGRAM=PERCON,&OUTFILE, &ELEM)
   ,ESC-CHAR=C'&'                          )
%/ASS-SYSDTA        TO=*SYSCMD                  ——————————————————————————  (2)
%/MOD-SDF-OPT      MODE=*TEST                  ,DEFAULT-PROG-NAME=PERCON
          ,LOG=*INPUT-FORM                      ——————————————————————————  (3)
%/SHOW-SDF-OPT      INFORMATION=*USER           ——————————————————————————  (4)
%  USER      : :2OSG:$USER1.SYSSDF.USER.EXAMPLE.1
%              VERSION : UNDEFINED
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *TEST
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : PERCON
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
%/SHOW-FILE-ATTR    F-NAME=TEST       ——————————————————————————————————  (5)
%/START-PERCON
%//SHOW-SDF-OPT                       ——————————————————————————————————  (6)
%//ASS-INPUT-F      FILE=*DISK-F(NAME=TEST)      ————————————————————————  (7)
%//ASS-OUT-F       FILE=*DISK-F(NAME=&OUTFILE)
%&OUTFILE=test4                       ——————————————————————————————————  (8)
%//ASS-OUT-F        FILE=*DISK-F(NAME=TEST4)     ————————————————————————  (9)
%//SEL-INPUT-REC    COND= ((20,4)=NUMERIC AND (25,5)=ALPHA)
%//START-CONV
%//END
%/SHOW-FILE-ATTR    F-NAME=TEST4
%/MOD-JOB-SW        ON=(1)
%/START-PROG        FROM-FILE=$LMS
%  CMD0091 NEXT INPUT(S) IGNORED UNTIL MESSAGE CMD0092   ———————————————  (10)
%LIB FILE=USER.LIB,USAGE=BOTH
%TOCX *
%ADDX &OUTFILE>&ELEM
%END
%  CMD0092 END OF IGNORED INPUT OR INPUTS      ——————————————————————————  (11)
%/SET-JOB-STEP
%/MOD-JOB-SW        OFF=(1)
%/MODIFY-SDF-OPT    MODE=*EXEC        ,DEFAULT-PROG-NAME=*NONE  —————————  (12)
%/END-PROC
```

(1) The PROC.SELECT procedure is called with procedure parameter MODE=*TEST.

(2) System file SYSDTA is assigned system file SYSCMD as input source, i.e. data records are read from the procedure file, too.

(3) The SDF settings are changed: MODE=*TEST causes the test mode to be activated. Testing is to include statements with leading "//". All input is logged as entered.

(4) SHOW-SDF-OPTIONS is executed, i.e. the current SDF settings are displayed.

(5) The value of job variable JV.INFILE ("TEST") has been substituted for &(JV.INFILE).

(6) Statements with leading "//" are syntax-checked but not executed (note that SHOW-SDF-OPTIONS is a statement here).

(7) Job variable replacement has taken place (as for item 5).

(8) Procedure parameter OUTFILE was omitted when the procedure was called and is therefore requested via prompting.

(9) The value "TEST4" is set for &OUTFILE.

(10) The subsequent statements to the LMS routine are ignored because they are not preceded by "//".
In accordance with the declaration, statements with leading "//" are syntax-checked only on the basis of the statements defined for PERCON in the syntax file. If other programs with an SDF interface are used, the declaration must be altered beforehand.

(11) The last data record to be ignored is sensed. The next input line starts with "/" or "//".

(12) MODIFY-SDF-OPTIONS is executed, which in this case means deactivation of test mode.

4.11 Task-specific default values

When entering commands/statements, users can omit operands whose default values meet their requirements. For frequently used commands/statements they are free to define more appropriate operand values within their dialog task. These task-specific default values are used in interactive input instead of the defaults defined in the syntax files.

Defining task-specific default values

Task-specific default values can only be defined within a dialog task through interactive input or in a procedure. To define them, you enter the command/statement with all operand values which are to be defined as defaults. The command or statement name must begin with an exclamation mark (!). Only the syntax of the input is checked, and the input is stored as a definition of task-specific default values. The omission of mandatory operands is ignored.
At each subsequent entry of the command/statement, the task-specific default values are used instead of the original ones.

The following restrictions apply when defining task-specific default values:

- Definition is not possible in batch mode.
- Task-specific default values cannot be defined for secret operands unless the value is to be one of the following:
  - an operand value defined with OUTPUT=*SECRET-PROMPT (generally the operand value *SECRET)
  - an operand value defined with SECRET=*NO.
- They cannot be defined for SDF-P control flow commands (e.g. IF, FOR) and for ISP commands.
- No more than 10 definitions can be stored for one command/statement.
- No more than 100 definitions may be stored in all.

Definition in the dialog

With the input `!<command> <operands>` in command mode, the user can define task-specific default values for the operands of the command <command> that are specified with <operands> . Similarly, with the input `!<statement> <operands>` in program mode, task-specific default values can be defined for the statement <statement> of the loaded program. The user can also switch to temporarily guided dialog to define the default values. In this case, the command/statement name must end with a question mark. Only the operand values that have actually changed are stored as task-specific default values.
As with the normal input of commands/statements, abbreviations are allowed.

Definition in procedures

Task-specific default values can also be defined in procedures, but the definition is only effective for interactive input. To avoid unintentional side effects, only the default values defined in the syntax files are effective in procedure mode.
Procedures enable the user to automate or simplify the definition of task-specific default values. When defined in a logon procedure, the default values are already available at the beginning of the interactive task. Users can prepare procedures for various requirements, which define the desired number of default values and can be called up as necessary.

The definition for a command begins with “/“, the definition for a statement with “//“. Otherwise, defining the values is the same as with interactive input. A command name can be preceded by a label. Definitions are not allowed within the procedure header.

The definition for a statement can also be made in command mode, i.e. the program need not be loaded. For syntax analysis, the program entered as the SDF option DEFAULT-PROGRAM-NAME is used (see the MODIFY-SDF-OPTIONS command). The definition is also stored for use with this program.

Syntax errors during definition of default values trigger error recovery.

Definition at the program interface

The CMDTST and TRCMD macros enable the definition of task-specific default values. With all other SDF macro calls, the definition of default values is rejected as a syntax error.

Evaluation of the definitions

When a command/statement is entered, SDF checks whether there are task-specific default values stored for it and, if so, inserts these values into the entered string. It then analyzes the syntax of the input.

A change of syntax files (e.g. when the version of a subsystem is changed) can mean that stored default values do not match the current syntax, in which case the input is rejected. If this happens, the user must delete the errored definition.

If in the definition lower-ranking operands are specified outside the structure (STRUCTURE-IMPLICIT notation), the structure-initiating operand value does not automatically become the task-specific default value.
For example, the definition `!CREATE-FILE ACCESS=*READ` does not automatically make the higher-ranking operand `PROTECTION=*PARAMETERS` the default value.

Several definitions can be stored for one command/statement. All definitions stored for the command/statement are evaluated in the order in which they were defined.
For one and the same operand, the last-defined default value is used each time.
To add a new default value to the definition of operands of a structure level or to modify an individual default value, it is sufficient to store a new definition with this default value.

Administering task-specific default values

The SHOW-INPUT-DEFAULTS command enables users to find out details of all currently defined task-specific default values. Output is to SYSOUT or SYSLST and can be restricted to specific commands, statements and programs. The definitions can be output with an input serial number, which allows the specific deletion of individual definitions. At program level the standard statement of the same name is available with the same functionality.

The RESET-INPUT-DEFAULTS command allows users to delete task-specific default values. If no operands are specified, the default values of all commands are deleted. Users can also delete the definitions of selected commands/statements, and individual definitions can be singled out for deletion via the input serial number.

Examples

Example 1

```
%  BLS0517 MODULE 'SDAMAIN' LOADED                                     (1)
%  SDA0001 'SDF-A' VERSION '04.1E10' STARTED                           (1)
%//open mode=? ————————————————————————————————————————————————————————  (1)
%  CMD0090 EXPLANATION OF OPERAND ‘MODE‘:
*UPDATE() or *CREATE() or *READ or *INIT() -DEFAULT-: *UPDATE
%//!open mode=*read ———————————————————————————————————————————————————  (2)
%//open mode=? ————————————————————————————————————————————————————————  (3)
%  CMD0090 EXPLANATION OF OPERAND ‘MODE‘:
*UPDATE() or *CREATE() or *READ or *INIT() -DEFAULT-: *READ
%//
```

(1) The SDF-A utility is started. The operand values of the MODE operand of the OPEN-SYNTAX-FILE statement are then queried. *UPDATE is displayed as the default value.

(2) The default value is changed to *READ for this particular task.

(3) When the operand values are queried again, *READ is now displayed as the default value.

Example 2

```
/show-file-attr creation-date=*today ——————————————————————————————————  (1)
%        84 :2OSG:$USER1.DATEI.3
%         3 :2OSG:$USER1.PROC.JV
%:2OSG: PUBLIC:      2 FILES RES=        87 FRE=       21 REL=       18 PAGES

/reset-input-defaults *all ————————————————————————————————————————————  (2)

/call-proc proc.default-1,log=*yes ————————————————————————————————————  (3)
%          1  1 /CMD-DEF-1:
%          1  1 /  !SHOW-FILE-ATTR  INF=*MIN
%          2  1 /!CRE-FILE       SUP=*PRIV(VOL=WORK01,DEV-TYPE=D3490-30)
%          3  1 /!CRE-FILE        SUP=*PUBLIC
%          4  1 /CMD-DEF-2:
%          4  1 /  !PRINT-DOC       LINE-SPACING=*BY-EBCDIC-CONTR
%          5  1 /PROG-DEF-1:
%          5  1 / MOD-SDF-OPT      DEFAULT-PROG=SDF-A
%//!OPEN-SYNTAX-FILE MODE=*READ
%//!SHOW             OBJ=*CMD(NAME=*ALL),ATT-INFO=*NO
%          8  1 /PROG-DEF-2:
%          8  1 / MOD-SDF-OPT      DEFAULT-PROG=SORT
```

```
%//!SORT-RECORDS     KEEP-EQUAL-SEQUENCES=*YES
%         10  1 /END:
%         10  1 /   IF-BLOCK-ERROR
%         12  1 /end-if
%         13  1 /MOD-SDF-OPT      DEFAULT-PROG=*NONE
%         14  1 /SHOW-INPUT-DEFAULTS *CMD

/!SHOW-FILE-ATTR INFORMATION=*MINIMUM
/!CRE-FILE SUPPORT=*PRIVATE-DISK(VOLUME=WORK01,DEVICE-TYPE=D3490-30)
/!CRE-FILE SUPPORT=*PUBLIC-DISK
/!PRINT-DOC LINE-SPACING=*BY-EBCDIC-CONTROL
%         15  1 /WRITE-TEXT 'SDF-A Default-Werte:'
SDF-A Default-Werte:
%         16  1 /SHOW-INPUT-DEFAULTS *STMT(PROG=SDF-A)
//!OPEN-SYNTAX-FILE MODE=*READ
//!SHOW OBJECT=*COMMAND(NAME=*ALL),ATTACHED-INFORMATION=*NO
//!OPEN-SYNTAX-FILE MODE=*READ
//!SHOW OBJECT=*COMMAND(NAME=*ALL),ATTACHED-INFORMATION=*NO
%         17  1 /WRITE-TEXT 'SORT Default-Werte:'
SORT Default-Werte:
%         18  1 /SHOW-INPUT-DEFAULTS *STMT(PROG=SORT)
//!SORT-RECORDS KEEP-EQUAL-SEQUENCES=*YES
//!SORT-RECORDS KEEP-EQUAL-SEQUENCES=*YES
%             1 /EXIT-PROCEDURE ERROR=*NO

/shid *all, input-serial-number=*yes  ———————————————————————————  (4)

/"   8 :" !SHOW-FILE-ATTR INFORMATION=*MINIMUM
/"   9 :" !CRE-FILE SUPPORT=*PRIVATE-DISK(VOLUME=WORK01,DEVICE-TYPE=D3490-30)
/"  10 :" !CRE-FILE SUPPORT=*PUBLIC-DISK
/"  11 :" !PRINT-DOC LINE-SPACING=*BY-EBCDIC-CONTROL
//"  12 :" !OPEN-SYNTAX-FILE MODE=*READ
//"  13 :" !SHOW OBJECT=*COMMAND(NAME=*ALL),ATTACHED-INFORMATION=*NO
//"  14 :" !SORT-RECORDS KEEP-EQUAL-SEQUENCES=*YES
/show-file-attr creation-date=*today  ———————————————————————————  (5)
%S NNN NW              84 :2OSG:$USER1.DATEI.3
%S NNN NW               3 :2OSG:$USER1.PROC.JV
```

(1) SHOW-FILE-ATTRIBUTES displays all files created on the current day. These are output implicitly with INFORMATION=*NAME-AND-SPACE (default value of the command).

(2) RESET-INPUT-DEFAULTS resets all task-specific default values set so far.

(3) The procedure PROC.DEFAULT-1 is called. In the procedure, task-specific default values for commands and for statements of the SDF-A and SORT programs are set (cf. procedure execution). The procedure ends with output of the task-specific default values.
Note:
The programs are not called. The default values of the statements are set in command mode. Before they are set, the corresponding program is set by means of the MODIFY-SDF-OPTION command (operand DEFAULT-PROGRAM-NAME).

(4) SHOW-INPUT-DEFAULTS (alias SHID) then displays all task-specific default values with the input serial number again.

(5) When the SHOW-FILE-ATTRIBUTES command is entered again, all files created on the current day are displayed. These are now output with the task-specific default value INFORMATION=*MINIMUM (cf. default value with the input serial number 8).

5 Examples of command input in interactive mode

The sample session below shows command input in all possible forms of unguided and guided dialog. Among other things, it demonstrates

– which types of prompting SDF offers on the three levels of guided dialog
– how input can be compressed
– how SDF supports the correction of errored entries
– how to switch to temporarily guided dialog
– how to switch guidance levels
– how to call information on the use of SDF.

The sample session was run on a computer with BS2000/OSD-BC V5.0A. The user ID under which the user in the sample session logs on to the system is **USER1**. Language code E was set both at system generation and in the catalog entry for user ID USER1. As a result, English is used as the language of both the SDF dialog and message output (see section “Language for SDF output” on page 46 for system and user ID-specific language settings). The frequently used command SHOW-FILE-ATTRIBUTES is used to demonstrate the various input options. For a detailed description of this command please refer to Volume 3 of the “Commands” manual [1]. Four files are cataloged under user ID USER1:

– the ISAM files TEST.EXAMPLE.1 and TEST.EXAMPLE.2
– the SAM files TEST.EXAMPLE.3 and TEST.EXAMPLE.4

If you wish to familiarize yourself with the various SDF input options, you can duplicate the sample session on your own system, adapting the LOGON specifications (e.g. user ID) and the file names as required.

After that, the function key assignments for the Styleguide mode are set. After the NEXT line, all possible inputs are displayed depending on the guidance mode, and the possible functions and function keys are displayed in a separate line (KEYS:). If the old mode is set, then instead of using the function keys stated, the corresponding old mode keys are to be used. If there is no corresponding function key in the old mode, then the corresponding control statement or the corresponding command must be entered in the NEXT line (see the section “Function keys” on page 65).

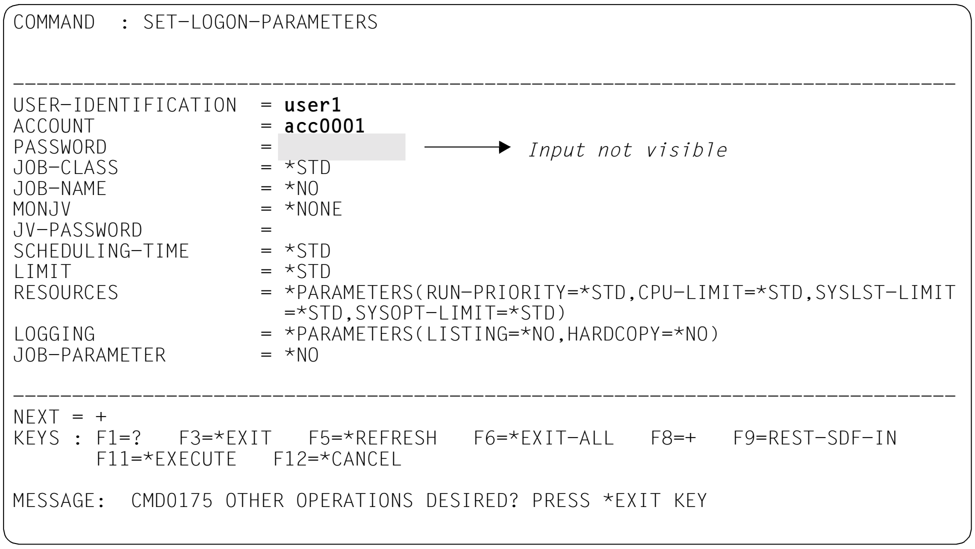
You can terminate guided dialog and the sample session at any time by entering the command MODIFY-SDF-OPTIONS GUIDANCE=*EXPERT (or *NO) in the NEXT line.

5.1 HELP-SDF

After processing of the LOGON command, the user guidance level is preset to GUIDANCE=*EXPERT (expert mode). Once the connection has been set up, the system requests logon:

```
%  JMS0150 INSTALLATION ' S150-40', BS2000 VERSION 'V140', HOST 'D016ZE04':
PLEASE ENTER '/SET-LOGON-PARAMETERS' OR '?'
/?
```

The logon request is answered with “?”. SDF therefore switches to temporarily guided dialog for the SET-LOGON-PARAMETERS command:



```
COMMAND  : SET-LOGON-PARAMETERS


--------------------------------------------------------------------------------
USER-IDENTIFICATION  = user1
ACCOUNT              = acc0001
PASSWORD             =                  ------> Input not visible
JOB-CLASS            = *STD
JOB-NAME             = *NO
MONJV                = *NONE
JV-PASSWORD          =
SCHEDULING-TIME      = *STD
LIMIT                = *STD
RESOURCES            = *PARAMETERS(RUN-PRIORITY=*STD,CPU-LIMIT=*STD,SYSLST-LIMIT
                       =*STD,SYSOPT-LIMIT=*STD)
LOGGING              = *PARAMETERS(LISTING=*NO,HARDCOPY=*NO)
JOB-PARAMETER        = *NO

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL

MESSAGE:  CMD0175 OTHER OPERATIONS DESIRED? PRESS *EXIT KEY
```

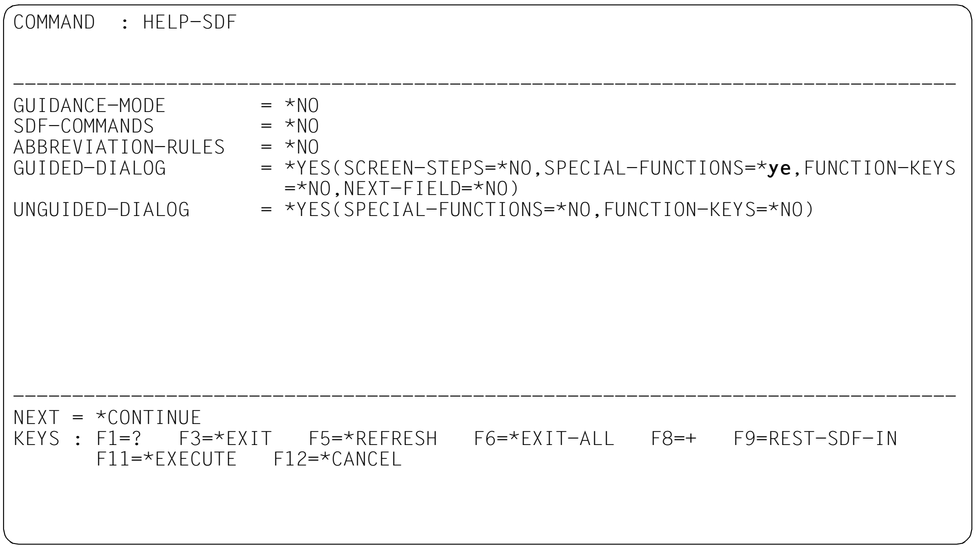
Figure 7: SET-LOGON-PARAMETERS command

The user enters the user ID (`USER1`), the account number (`ACC0001`), the password (`C'AUF54HT$'`, not visible since display of the input field for the “secret” PASSWORD operand is suppressed) and the job name (`TESTSDF`). Since the NEXT line already contains “+” by default, the next page of the operand form can be output after issuing the screen (DUE key). Since no other data needs to be entered for this example, the command is sent immediately (with the F11 key or by entering *EXECUTE in the NEXT line and pressing the DUE key). After processing of the SET-LOGON-PARAMETERS command, user guidance is set to GUIDANCE=*EXPERT (expert mode). The HELP-SDF command without operands outputs general information on SDF. Renewed input of the command with a question mark causes a changeover to temporarily guided dialog.

```
%  JMS0066 JOB 'SDFTEST' ACCEPTED ON 01-10-04 AT 17:14, TSN = 1PY8
/help-sdf
% Introduction
%
% SDF is a convenient command interpreter and dialogue manager
% If the user desires, input can be entered as before.
% But the user can also utilize the advantages of SDF:
%  - Abbreviation mechanism
%  - Block input
%  - Guided dialogue
%  - Command bufferization (history)
%  - Definition of task-specific default values
%
% In addition, a more comprehensible command language has
% has been developed:
% The names of commands, operands and constant operand
% values were chosen so as to indicate clearly their function.
% Similar features (for example, the name of a file) are
% named accordingly similar naming rules (for example, FILE-NAME=
% or FROM-FILE= or TO-FILE=).
% Incorrect input is presented for correction along with
% a message.
% Commands which are part of the old commando language (ISP)
% can be input in the form "<old cmd>?". The new SDF-command
% is then displayed with its new syntax, i.e. a menu driven
% dialog allows to set the appropriate operand values.
% For further information, the user may enter "HELP-SDF?".
% The operand form for the command will then be displayed.
% Further information for an operand may then be obtained
% by setting that operand to "YES".
%
```

```
/help-sdf?
```

When “help-sdf?” is entered, SDF displays the operand form for the HELP-SDF command.



```
COMMAND  : HELP-SDF


--------------------------------------------------------------------------------
GUIDANCE-MODE         = *NO
SDF-COMMANDS          = *NO
ABBREVIATION-RULES    = *NO
GUIDED-DIALOG         = *YES(SCREEN-STEPS=*NO,SPECIAL-FUNCTIONS=*ye,FUNCTION-KEYS
                        =*NO,NEXT-FIELD=*NO)
UNGUIDED-DIALOG       = *YES(SPECIAL-FUNCTIONS=*NO,FUNCTION-KEYS=*NO)









--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Figure 8: Operand form for the HELP-SDF command

The default value *NO of the GUIDED-DIALOG operand is changed to “*ye” (equivalent to *YES) in the *YES structure for the SPECIAL-FUNCTIONS operand, thereby calling up information on the function keys. The specification in the NEXT line controls what happens when the screen is sent off (by pressing the DUE key). The preset entry *CONTINUE stands for *EXECUTE or +, and in this case causes the command to be executed. The following text is displayed:

```
%
% ? as operand value:
%         Supplies help text and options (limit values, etc.)
%         for the operand; in addition, detailed error
%         messages in the case of incorrect input.
% ?? as operand value:
%         calls up a help text, a display of the range of
%         values for the relevant operand and an help text
%         giving information about the data types associated
%         with the operand.
% !  as operand value:
%         Enters the default value for the operand.
% (  following an operand value introducing a structure:
%         Displays the sub-form for the structure associated
%         with the operand value.
% () following an operand value introducing a structure:
%         Suppresses the sub-form and enters the default
%         values for the operands of the structure.
%  -   as the last character in an input line:
%         A continuation line is displayed.
% LZF key :
%         Deletes all characters in the input line, starting
%         at the cursor.
%
```

5.2 Unguided dialog

5.2.1 EXPERT form

After SDF has displayed the information on the function keys, the dialog is resumed as follows:

```
/sh-f-attr test.example.,file-struc=i  ————————————————————————————————— (1)
%       33 :2OSG:$USER1.TEST.EXAMPLE.1  ———————————————————————————————— (2)
%       72 :2OSG:$USER1.TEST.EXAMPLE.2
%:2OSG: PUBLIC:      2 FILES RES=      105 FRE=       26 REL=        9 PAGES
/sh-f-attr test,example.,file-struc=(i,s)  —————————————————————————————— (3)
%  CMD0051 INVALID OPERAND 'INFORMATION'
%  CMD0064 OPERAND VALUE 'EXAMPLE.' DOES NOT MATCH DATA TYPE '*NAME-AND-SPACE
OR *SPACE-SUMMARY OR *ALL-ATTRIBUTES OR *PARAMETERS() OR *STATISTICS OR
*MINIMUM'       ————————————————————————————————————————————————————————— (4)

/sh-f-attr inf=?        ————————————————————————————————————————————————— (5)
%  CMD0090 EXPLANATION OF OPERAND 'INFORMATION':
*NAME-AND-SPACE or *SPACE-SUMMARY or *ALL-ATTRIBUTES or *PARAMETERS() or
*STATISTICS or *MINIMUM -default-: *NAME-AND-SPACE

/sh-f-attr test.example.,file-struc=(i,s)  —————————————————————————————— (6)
%       33 :2OSG:$USER1.TEST.EXAMPLE.1      ————————————————————————————— (7)
%       72 :2OSG:$USER1.TEST.EXAMPLE.2
%       48 :2OSG:$USER1.TEST.EXAMPLE.3
%        3 :2OSG:$USER1.TEST.EXAMPLE.4
%:2OSG: PUBLIC:      4 FILES RES=      156 FRE=       30 REL=        9 PAGES
/mod-sdf-opt guid=n                        —————————————————————————————— (8)
%CMD:
```

(1) The SHOW-FILE-ATTRIBUTES command calls up the default information (size and name) on all ISAM files (“file-struc=i”) whose name begins with “test.example.”.

(2) The system displays the required information.

(3) The SHOW-FILE-ATTRIBUTES command calls up the default information (size and name) on all ISAM and SAM files (“file-struc=(i,s)”) whose name begins with “test.example.”. The FILE-STRUCTURE operand is assigned two values, “(i,s)”, which are enclosed in parentheses (list input). The file name “test,example.” is incorrect (comma instead of period).

(4) SDF interprets “example.” as a value for the INFORMATION operand (input as second positional operand) and rejects it as an incorrect entry.

(5) In the EXPERT form SDF does not conduct a correction dialog and instead requests command input. The user requests help information on the INFORMATION operand by means of entering a question mark instead of the operand value. The parentheses following the word *PARAMETERS indicate that *PARAMETERS initiates a structure.

(6) The incorrect command entered in step 3 is entered correctly.

(7) The system displays the required information.

(8) The MODIFY-SDF-OPTIONS command causes a switch to the NO form of unguided dialog. All the operands of this command are optional and have the default value *UNCHANGED. Consequently only the explicitly specified SDF options, in this case the user guidance form, are modified.

5.2.2 NO form

The dialog is continued as follows:

```
%CMD:sh-f-a test.,file-struc=i  ----------------------------------------- (1)
%  CMD0187 ABBREVIATION OF OPERATION NAME 'SH-F-A' AMBIGUOUS WITH REGARD TO
'SHOW-FILE-ATTRIBUTES,SHOW-FT-ADMISSION-SET'
%CORRECT OPERATIONNAME:sh-f-attr  ---------------------------------------- (2)
%       33 :2OSG:$USER1.TEST.EXAMPLE.1
%       72 :2OSG:$USER1.TEST.EXAMPLE.2
%:2OSG: PUBLIC:      2 FILES RES=      105 FRE=       26 REL=        9 PAGES
%CMD:sh-f-attr test,example.,file-struc=(i,s) ---------------------------- (3)
%  CMD0051 INVALID OPERAND 'INFORMATION'     ----------------------------- (4)
%  CMD0064 OPERAND VALUE 'EXAMPLE.' DOES NOT MATCH DATA TYPE '*NAME-AND-SPACE
OR *SPACE-SUMMARY OR *ALL-ATTRIBUTES OR *PARAMETERS() OR *STATISTICS OR
*MINIMUM'
%ENTER OPERANDS:     ----------------------------------------------------- (5)
%test,example.,file-struc=(i,s)  ----------------------------------------- (6)
test.example.,inf=name     ----------------------------------------------- (7)
%       33 :2OSG:$USER1.TEST.EXAMPLE.1     ------------------------------- (8)
%       72 :2OSG:$USER1.TEST.EXAMPLE.2
%       48 :2OSG:$USER1.TEST.EXAMPLE.3
%        3 :2OSG:$USER1.TEST.EXAMPLE.4
%:2OSG: PUBLIC:      4 FILES RES=      156 FRE=       30 REL=        9 PAGES
%CMD:
```

(1) The SHOW-FILE-ATTRIBUTES command calls up the default information (size and name) on all ISAM files (“file-struc=i”) with a name beginning with “test.example.”. The command is entered in the abbreviated form “sh-f-a”.

(2) SDF requests the user to correct the command name since the abbreviation entered is ambiguous. Once the unambiguous abbreviation “sh-f-attr” has been entered, the SHOW-FILE-ATTRIBUTES command is executed according to the selection criteria previously specified.

(3) The system displays the required information.

(4) The SHOW-FILE-ATTRIBUTES command calls up the default information (size and name) on all ISAM and SAM files (“file-struc=(i,s)”) with a name beginning with “test,example.”. The FILE-STRUCTURE operand is assigned two values -- “(i,s)” -- which are enclosed in parentheses (list input). The file name “test,example.” is incorrect (comma instead of period).

(5) SDF interprets “example.” as a value for the INFORMATION operand (input as second positional operand) and rejects it as an incorrect entry. The parentheses following the word *PARAMETERS indicate that *PARAMETERS initiates a structure.

(6) SDF invites the user to correct the operands.

(7) SDF offers the entered operands for correction.

(8) The operands are corrected. Both the file name and the INFORMATION operand must be corrected. It is also possible to perform the corrections by switching temporarily to guided dialog.

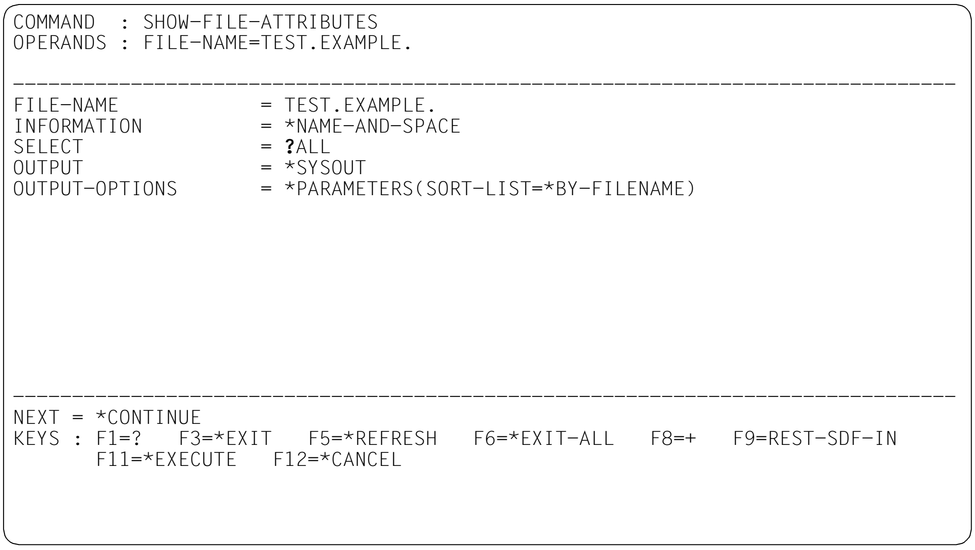
(9) The system displays the required information.

5.3 Temporarily guided dialog

To input a command (or statement) it is possible to switch temporarily from unguided to guided dialog. This is accomplished by entering a question mark after the command name.

```
%CMD:sh-f-attr? test.example.
```

When “sh-f-attr? test.example.” is entered, SDF displays the operand form for the SHOW-FILE-ATTRIBUTES command. This form is preset with the file name entered for the FILE-NAME operand and the default values of the optional operands INFORMATION, SELECT and OUTPUT.

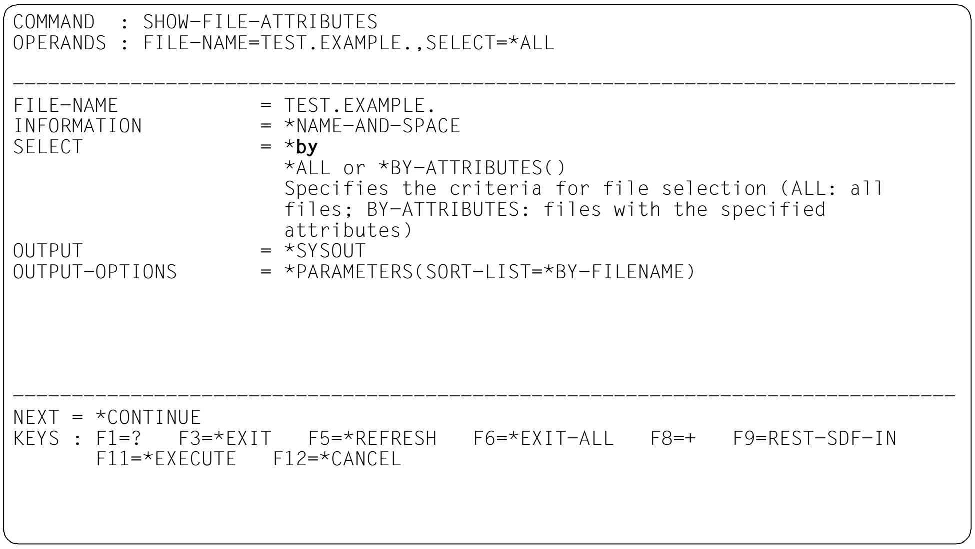


```
COMMAND  : SHOW-FILE-ATTRIBUTES
OPERANDS : FILE-NAME=TEST.EXAMPLE.

-------------------------------------------------------------------------------
FILE-NAME              = TEST.EXAMPLE.
INFORMATION            = *NAME-AND-SPACE
SELECT                 = ?ALL
OUTPUT                 = *SYSOUT
OUTPUT-OPTIONS         = *PARAMETERS(SORT-LIST=*BY-FILENAME)










-------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Figure 9: Operand form for the SHOW-FILE-ATTRIBUTES command

Additional information on the SELECT operand is called up by entering “?”. The rest of the line, “ALL”, need in this case not be deleted. The NEXT line is preset with *CONTINUE (stands for *EXECUTE or +) and in this case causes SDF to display the operand form for the SHOW-FILE-ATTRIBUTES command with an explanation of the SELECT operand.

SDF displays additional information on the SELECT operand. The parentheses following the *BY-ATTRIBUTES value indicate that *BY-ATTRIBUTES initiates a structure.

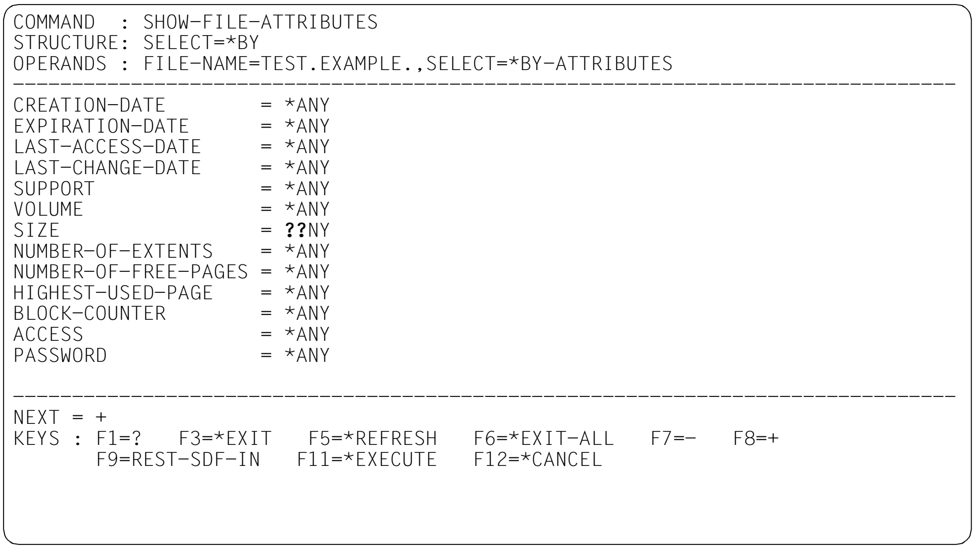


```
COMMAND  : SHOW-FILE-ATTRIBUTES
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*ALL

--------------------------------------------------------------------------------
FILE-NAME             = TEST.EXAMPLE.
INFORMATION           = *NAME-AND-SPACE
SELECT                = *by
                        *ALL or *BY-ATTRIBUTES()
                        Specifies the criteria for file selection (ALL: all
                        files; BY-ATTRIBUTES: files with the specified
                        attributes)
OUTPUT                = *SYSOUT
OUTPUT-OPTIONS        = *PARAMETERS(SORT-LIST=*BY-FILENAME)





--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Figure 10: Operand form for the SHOW-FILE-ATTRIBUTES command

The default value *ALL which is preset for the SELECT operand is overwritten with “*by” (*BY-ATTRIBUTES). Sending off the screen causes SDF to display a subform for the BY-ATTRIBUTES structure.

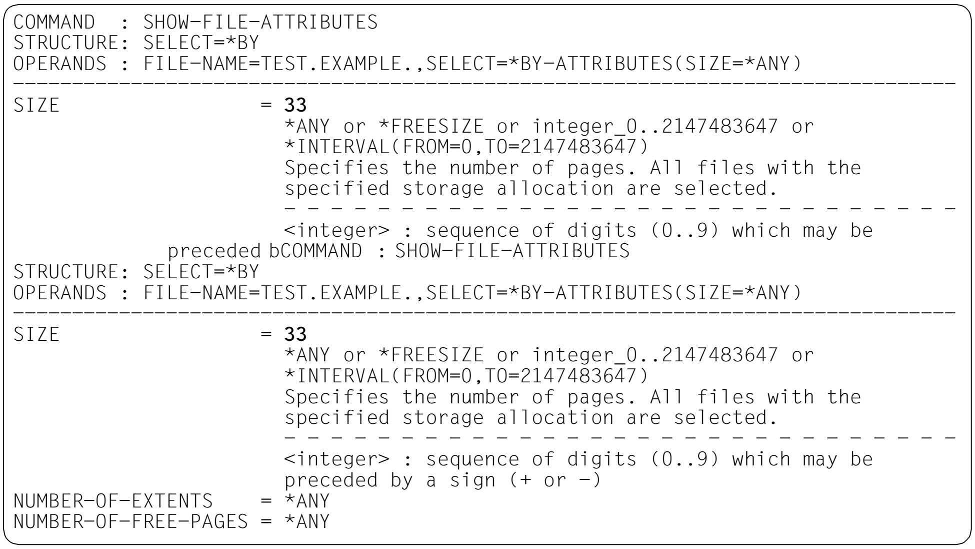
SDF displays the subform for the *BY-ATTRIBUTES structure.



```
COMMAND  : SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES
--------------------------------------------------------------------------------
CREATION-DATE         = *ANY
EXPIRATION-DATE       = *ANY
LAST-ACCESS-DATE      = *ANY
LAST-CHANGE-DATE      = *ANY
SUPPORT               = *ANY
VOLUME                = *ANY
SIZE                  = ??NY
NUMBER-OF-EXTENTS     = *ANY
NUMBER-OF-FREE-PAGES  = *ANY
HIGHEST-USED-PAGE     = *ANY
BLOCK-COUNTER         = *ANY
ACCESS                = *ANY
PASSWORD              = *ANY

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL
```

Figure 11: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

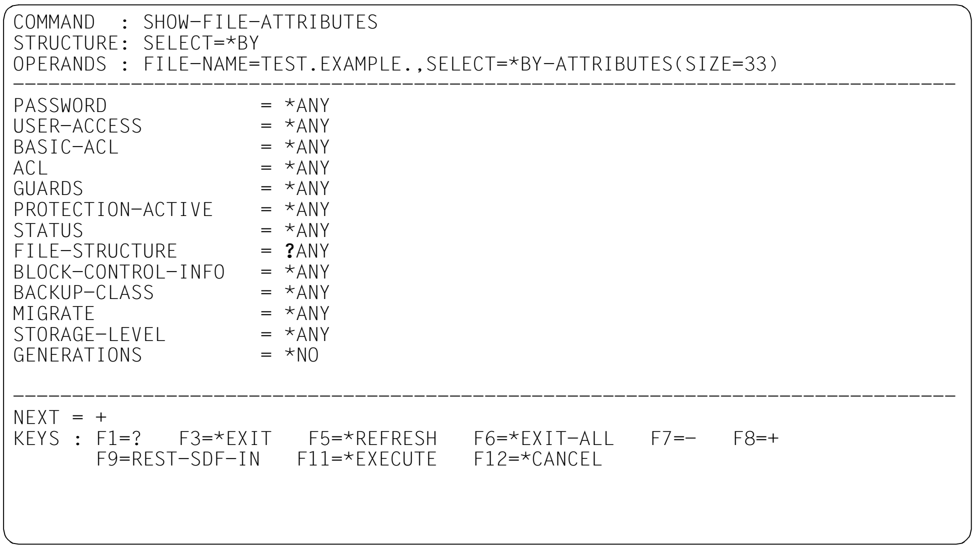
Entering “??” calls up additional information on the SIZE operand. The rest of the line, “Y”, need in this case not be deleted. Sending off the form causes SDF to output further information on the SIZE operand. In addition to possible operand values and the help text (if “?” is entered), information on data types permitted as input is displayed.



```
COMMAND  : SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES(SIZE=*ANY)
--------------------------------------------------------------------------------
SIZE                  = 33
                        *ANY or *FREESIZE or integer_0..2147483647 or
                        *INTERVAL(FROM=0,TO=2147483647)
                        Specifies the number of pages. All files with the
                        specified storage allocation are selected.
                        - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        <integer> : sequence of digits (0..9) which may be
             preceded bCOMMAND  : SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES(SIZE=*ANY)
--------------------------------------------------------------------------------
SIZE                  = 33
                        *ANY or *FREESIZE or integer_0..2147483647 or
                        *INTERVAL(FROM=0,TO=2147483647)
                        Specifies the number of pages. All files with the
                        specified storage allocation are selected.
                        - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        <integer> : sequence of digits (0..9) which may be
                        preceded by a sign (+ or -)
NUMBER-OF-EXTENTS     = *ANY
NUMBER-OF-FREE-PAGES  = *ANY
```

Figure 12: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

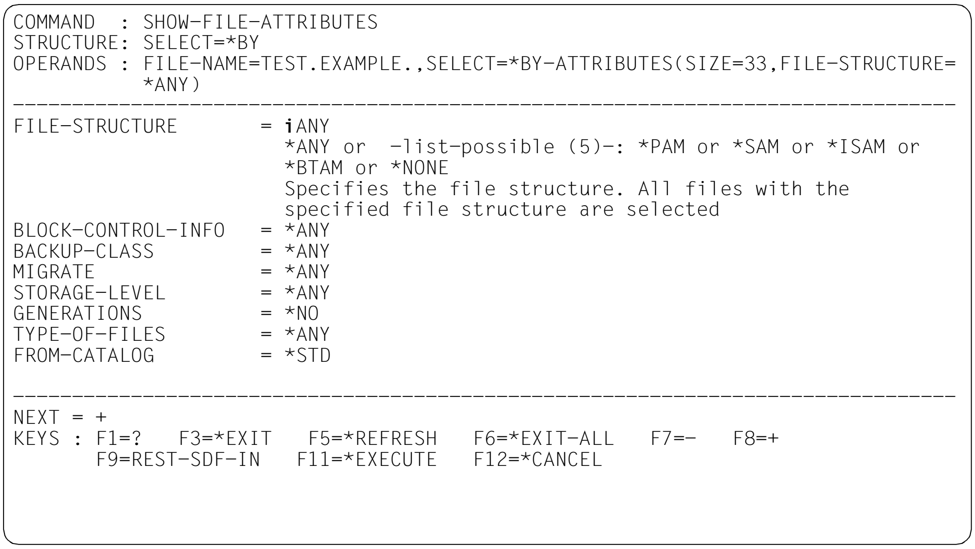
The operand value ANY predefined for the SIZE operand is overwritten with “33” (selecting files which occupy precisely 33 PAM pages). As the NEXT line is already preset with “+”, sending off the screen ($\boxed{\text{DUE}}$ key) causes the next page of the operand form to be displayed.



```
COMMAND  : SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES(SIZE=33)
--------------------------------------------------------------------------------
PASSWORD             = *ANY
USER-ACCESS          = *ANY
BASIC-ACL            = *ANY
ACL                  = *ANY
GUARDS               = *ANY
PROTECTION-ACTIVE    = *ANY
STATUS               = *ANY
FILE-STRUCTURE       = ?ANY
BLOCK-CONTROL-INFO   = *ANY
BACKUP-CLASS         = *ANY
MIGRATE              = *ANY
STORAGE-LEVEL        = *ANY
GENERATIONS          = *NO

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL
```

Figure 13: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

Entering “?” calls up additional information on the FILE-STRUCTURE operand. The remainder of the line, “ANY”, need not be deleted in this case. Sending off the form causes SDF to display further information on the FILE-STRUCTURE operand.

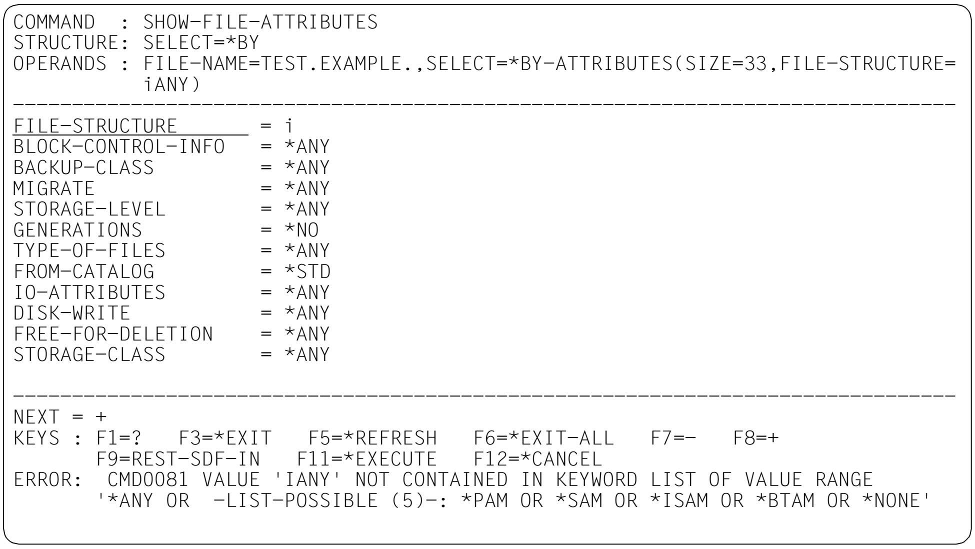


```
COMMAND  : SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES(SIZE=33,FILE-STRUCTURE=
           *ANY)
--------------------------------------------------------------------------------
FILE-STRUCTURE       = iANY
                       *ANY or  -list-possible (5)-: *PAM or *SAM or *ISAM or
                       *BTAM or *NONE
                       Specifies the file structure. All files with the
                       specified file structure are selected
BLOCK-CONTROL-INFO   = *ANY
BACKUP-CLASS         = *ANY
MIGRATE              = *ANY
STORAGE-LEVEL        = *ANY
GENERATIONS          = *NO
TYPE-OF-FILES        = *ANY
FROM-CATALOG         = *STD

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL
```

Figure 14: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

ISAM is the access method selected. The entry “i” is an unambiguous abbreviation for *ISAM. The leading asterisk can be omitted because only keywords are permitted as alternative operand values. The notation without an asterisk is not guaranteed in the long term and should therefore not be used exclusively in the dialog.
By mistake the rest of the line, “ANY”, is not deleted. The form is sent off.

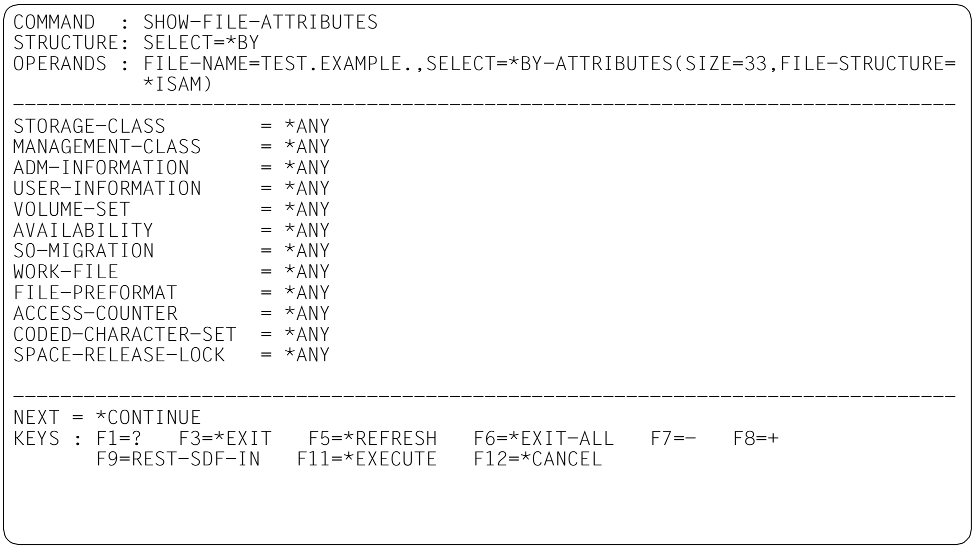
SDF also interprets the remainder of the line, “ANY”, as part of the operand value for FILE-STRUCTURE. It rejects the value “iANY” as an invalid entry and requests the user to correct it (the faulty operand is highlighted, in the example it is underlined).



```
COMMAND  : SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES(SIZE=33,FILE-STRUCTURE=
           iANY)
--------------------------------------------------------------------------------
FILE-STRUCTURE        = i
BLOCK-CONTROL-INFO    = *ANY
BACKUP-CLASS          = *ANY
MIGRATE               = *ANY
STORAGE-LEVEL         = *ANY
GENERATIONS           = *NO
TYPE-OF-FILES         = *ANY
FROM-CATALOG          = *STD
IO-ATTRIBUTES         = *ANY
DISK-WRITE            = *ANY
FREE-FOR-DELETION     = *ANY
STORAGE-CLASS         = *ANY

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL
ERROR:  CMD0081 VALUE 'IANY' NOT CONTAINED IN KEYWORD LIST OF VALUE RANGE
       '*ANY OR  -LIST-POSSIBLE (5)-: *PAM OR *SAM OR *ISAM OR *BTAM OR *NONE'
```

Figure 15: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

The operand value for FILE-STRUCTURE is corrected by deleting the characters “ANY”. When the form has been sent off, the next page in the form is displayed (default entry "+" in the NEXT line):



```
COMMAND  : SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES(SIZE=33,FILE-STRUCTURE=
           *ISAM)
--------------------------------------------------------------------------------
STORAGE-CLASS         = *ANY
MANAGEMENT-CLASS      = *ANY
ADM-INFORMATION       = *ANY
USER-INFORMATION      = *ANY
VOLUME-SET            = *ANY
AVAILABILITY          = *ANY
S0-MIGRATION          = *ANY
WORK-FILE             = *ANY
FILE-PREFORMAT        = *ANY
ACCESS-COUNTER        = *ANY
CODED-CHARACTER-SET   = *ANY
SPACE-RELEASE-LOCK    = *ANY

--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL
```

Figure 16: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

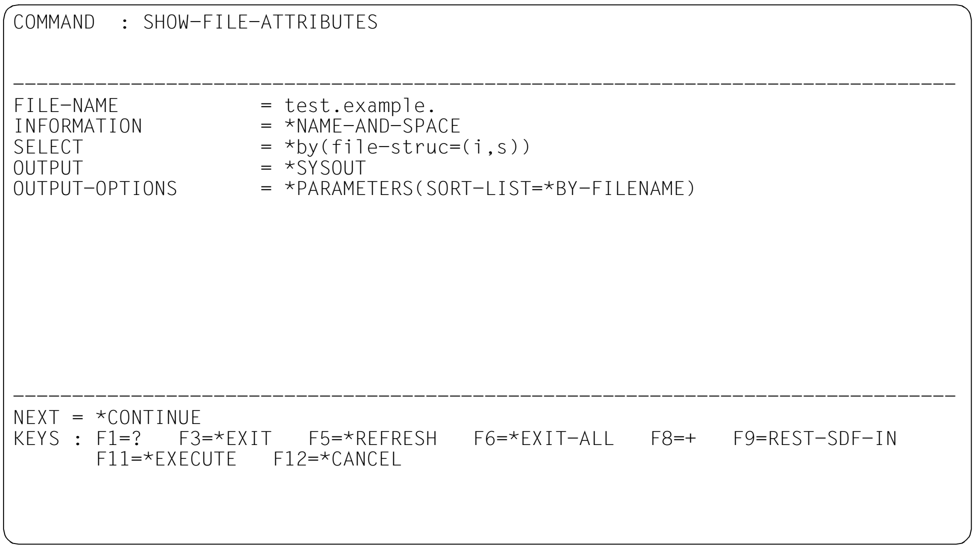
Sending off the form causes the command to be executed. The system displays the required information.

```
%        33 :2OSG:$USER1.TEST.EXAMPLE.1
%:2OSG: PUBLIC:       1 FILE  RES=        33 FRE=       24 REL=        9 PAGES
```

The dialog is continued as follows:

```
%CMD:fstat?
```

To input the FSTATUS command (ISP format), the user switches temporarily to guided dialog (question mark after the command name). The commands FSTATUS (ISP format) and SHOW-FILE-ATTRIBUTES (SDF format) are identical in function. Since FSTATUS was entered without operands, SDF displays the operand form for the SHOW-FILE-ATTRIBUTES command. This form is preset with the default values of the optional operands FILE-NAME, INFORMATION, SELECT and OUTPUT.



```
COMMAND  : SHOW-FILE-ATTRIBUTES


-----------------------------------------------------------------------------
FILE-NAME              = test.example.
INFORMATION            = *NAME-AND-SPACE
SELECT                 = *by(file-struc=(i,s))
OUTPUT                 = *SYSOUT
OUTPUT-OPTIONS         = *PARAMETERS(SORT-LIST=*BY-FILENAME)









-----------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Figure 17: Operand form for the SHOW-FILE-ATTRIBUTES command

The default value *ALL of the FILE-NAME operand is overwritten with “test.example.”, the default value *ALL of the SELECT operand with “by” (BY-ATTRIBUTES). Display of a subform for the associated structure is suppressed by the direct entry of a lower-ranking operand structure ”(file-struc=i)”. The entry “file-struc=i” assigns the FILE-STRUCTURE operand the value *ISAM. Sending off the form causes the command to be executed. The system displays the required information.

```
%        33 :2OSG:$USER1.TEST.EXAMPLE.1
%        72 :2OSG:$USER1.TEST.EXAMPLE.2
%:2OSG: PUBLIC:      2 FILES RES=       105 FRE=       26 REL=        9 PAGES
```

The dialog is continued as follows:

```
%CMD:sh-f-attr test,example.,select=(file-struc=i)      ——————————————————  (9)
%  CMD0051 INVALID OPERAND 'INFORMATION'         ——————————————————————————— (10)
%  CMD0064 OPERAND VALUE 'EXAMPLE.' DOES NOT MATCH DATA TYPE '*NAME-AND-SPACE
OR *SPACE-SUMMARY OR *ALL-ATTRIBUTES OR *PARAMETERS() OR *STATISTICS OR
*MINIMUM'
%ENTER OPERANDS:     —————————————————————————————————————————————————————— (11)
%test,example.,select=(file-struc=i)   ————————————————————————————————————— (12)
? —————————————————————————————————————————————————————————————————————————— (13)
```

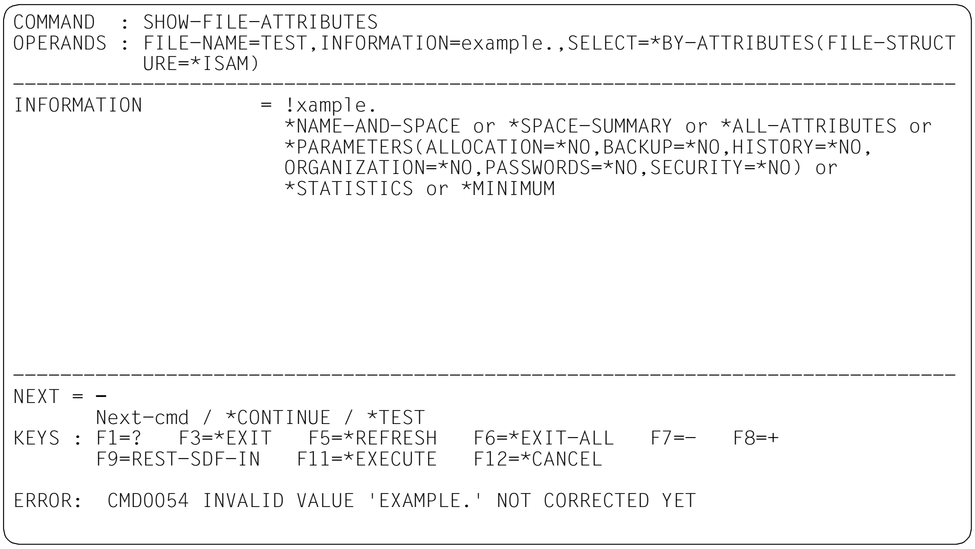
(1) The SHOW-FILE-ATTRIBUTES command calls up the default information (size and name) on all ISAM files (”select=(file-struc=i)” is equivalent to `SELECT=*BY-ATTRIBUTES(FILE-STRUCTURE =*ISAM)`) whose name begins with “test,example.”. By mistake a comma is inserted in the partially qualified file name instead of a period.

(2) SDF interprets “example.” as a value for the INFORMATION operand (input as second positional operand) and rejects it as an incorrect entry.

(3) SDF invites the user to correct the operands.

(4) SDF offers the entered operands for correction.

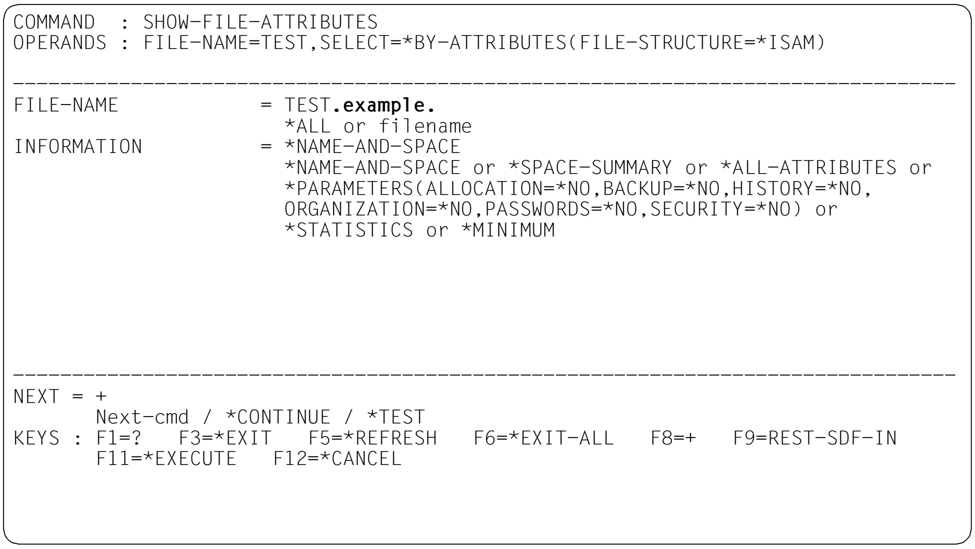
(5) Entering the question mark causes SDF to switch to temporarily guided dialog while the corrections are performed.



```
COMMAND  : SHOW-FILE-ATTRIBUTES
OPERANDS : FILE-NAME=TEST,INFORMATION=example.,SELECT=*BY-ATTRIBUTES(FILE-STRUCT
           URE=*ISAM)
--------------------------------------------------------------------------------
INFORMATION          = !xample.
                         *NAME-AND-SPACE or *SPACE-SUMMARY or *ALL-ATTRIBUTES or
                         *PARAMETERS(ALLOCATION=*NO,BACKUP=*NO,HISTORY=*NO,
                         ORGANIZATION=*NO,PASSWORDS=*NO,SECURITY=*NO) or
                         *STATISTICS or *MINIMUM









--------------------------------------------------------------------------------
NEXT = -
      Next-cmd / *CONTINUE / *TEST
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
      F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL

ERROR:  CMD0054 INVALID VALUE 'EXAMPLE.' NOT CORRECTED YET
```

Figure 18: Operand form for the SHOW-FILE-ATTRIBUTES command

Entering “!” assigns the INFORMATION operand its default value (NAME-AND-SPACE). The rest of the line, “XAMPLE.”, need in this case not be deleted. In order to correct the value of the first operand (FILE-NAME), the user has to scroll backwards in the operand form. This is achieved by F7 (corresponds to entering “–” in the NEXT line. The rest of the “CONTINUE” line must be deleted with the LZF key, for example). Sending off the form by means of the DUE key causes SDF to display the beginning of the operand form.



```
COMMAND  : SHOW-FILE-ATTRIBUTES
OPERANDS : FILE-NAME=TEST,SELECT=*BY-ATTRIBUTES(FILE-STRUCTURE=*ISAM)

--------------------------------------------------------------------------------
FILE-NAME             = TEST.example.
                        *ALL or filename
INFORMATION           = *NAME-AND-SPACE
                        *NAME-AND-SPACE or *SPACE-SUMMARY or *ALL-ATTRIBUTES or
                        *PARAMETERS(ALLOCATION=*NO,BACKUP=*NO,HISTORY=*NO,
                        ORGANIZATION=*NO,PASSWORDS=*NO,SECURITY=*NO) or
                        *STATISTICS or *MINIMUM







--------------------------------------------------------------------------------
NEXT = +
       Next-cmd / *CONTINUE / *TEST
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Figure 19: Operand form for the SHOW-FILE-ATTRIBUTES command

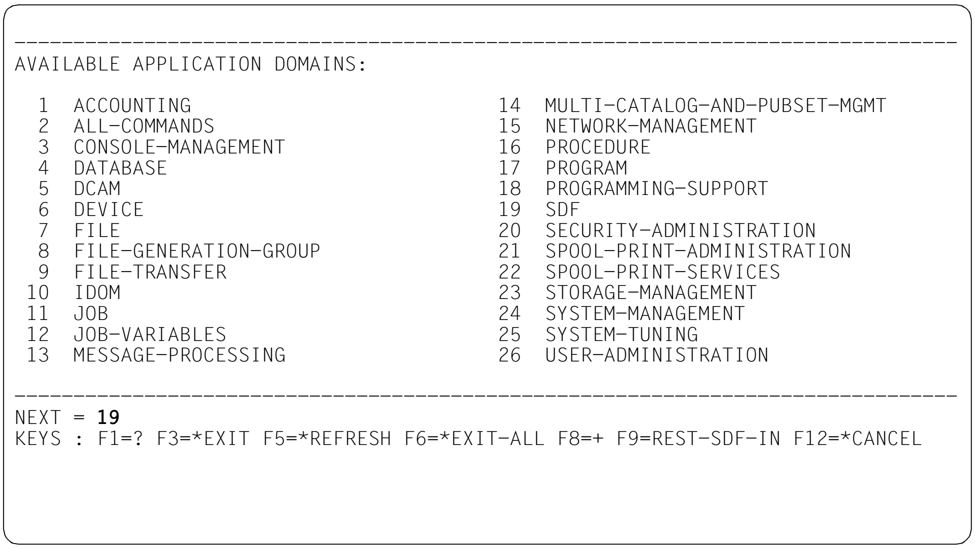
The value of the FILE-NAME operand is corrected. Sending off the form by means of the F11 key (equivalent to entering *EXECUTE in the NEXT line) causes the command to be executed.

```
%        33 :2OSG:$USER1.TEST.EXAMPLE.1
%        72 :2OSG:$USER1.TEST.EXAMPLE.2
%:2OSG: PUBLIC:      2 FILES RES=      105 FRE=       26 REL=         9 PAGES
```

The dialog is continued as follows:

```
%CMD:?
```

Entering a question mark instead of a command causes SDF to display the application domain menu.

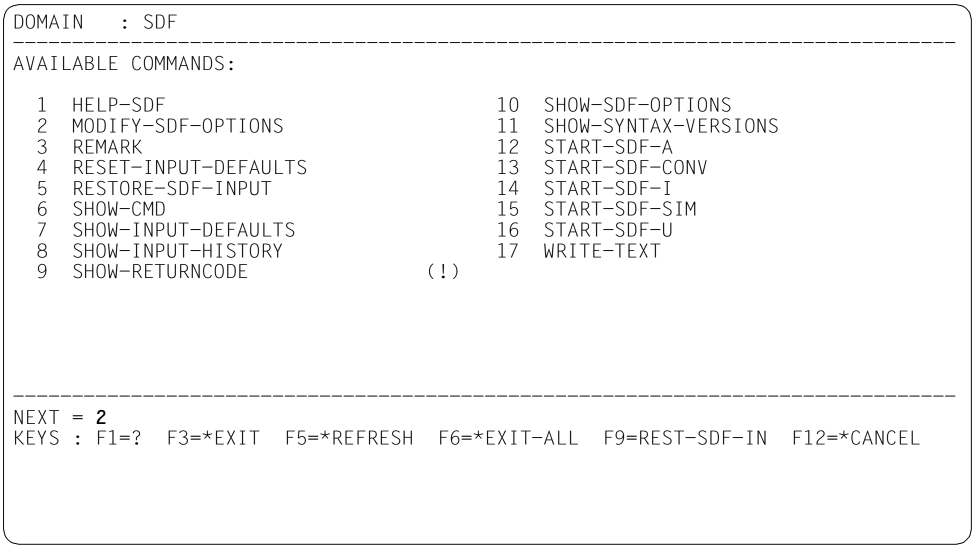


```
-------------------------------------------------------------------------------
AVAILABLE APPLICATION DOMAINS:

  1  ACCOUNTING                              14  MULTI-CATALOG-AND-PUBSET-MGMT
  2  ALL-COMMANDS                            15  NETWORK-MANAGEMENT
  3  CONSOLE-MANAGEMENT                      16  PROCEDURE
  4  DATABASE                                17  PROGRAM
  5  DCAM                                    18  PROGRAMMING-SUPPORT
  6  DEVICE                                  19  SDF
  7  FILE                                    20  SECURITY-ADMINISTRATION
  8  FILE-GENERATION-GROUP                   21  SPOOL-PRINT-ADMINISTRATION
  9  FILE-TRANSFER                           22  SPOOL-PRINT-SERVICES
 10  IDOM                                    23  STORAGE-MANAGEMENT
 11  JOB                                     24  SYSTEM-MANAGEMENT
 12  JOB-VARIABLES                           25  SYSTEM-TUNING
 13  MESSAGE-PROCESSING                      26  USER-ADMINISTRATION

-------------------------------------------------------------------------------
NEXT = 19
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F8=+ F9=REST-SDF-IN F12=*CANCEL
```

Figure 20: Application domain menu

Entering “16” initiates display of the command menu for the application domain SDF. This could also be done by entering “(sdf)” instead of “19”.

SDF displays the command menu for the application domain SDF.

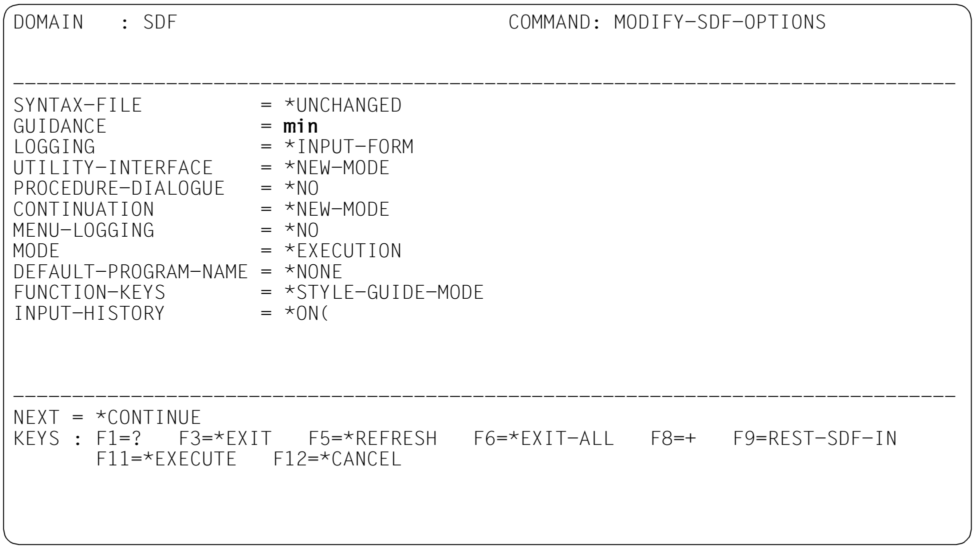


```
DOMAIN   : SDF
-------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  HELP-SDF                                10  SHOW-SDF-OPTIONS
  2  MODIFY-SDF-OPTIONS                      11  SHOW-SYNTAX-VERSIONS
  3  REMARK                                  12  START-SDF-A
  4  RESET-INPUT-DEFAULTS                    13  START-SDF-CONV
  5  RESTORE-SDF-INPUT                       14  START-SDF-I
  6  SHOW-CMD                                15  START-SDF-SIM
  7  SHOW-INPUT-DEFAULTS                     16  START-SDF-U
  8  SHOW-INPUT-HISTORY                      17  WRITE-TEXT
  9  SHOW-RETURNCODE               (!)





-------------------------------------------------------------------------------
NEXT = 2
KEYS : F1=?  F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F9=REST-SDF-IN  F12=*CANCEL
```

Figure 21: Command menu for the application domain SDF

Entering “2” selects the MODIFY-SDF-OPTIONS command.

SDF displays the operand form for the MODIFY-SDF-OPTIONS command.

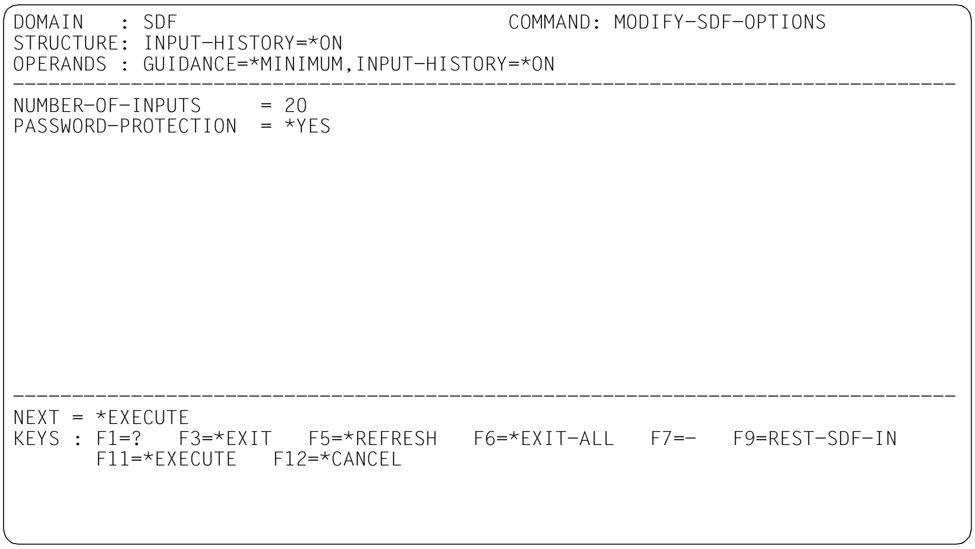


```
DOMAIN   : SDF                            COMMAND: MODIFY-SDF-OPTIONS


--------------------------------------------------------------------------------
SYNTAX-FILE          = *UNCHANGED
GUIDANCE             = min
LOGGING              = *INPUT-FORM
UTILITY-INTERFACE    = *NEW-MODE
PROCEDURE-DIALOGUE   = *NO
CONTINUATION         = *NEW-MODE
MENU-LOGGING         = *NO
MODE                 = *EXECUTION
DEFAULT-PROGRAM-NAME = *NONE
FUNCTION-KEYS        = *STYLE-GUIDE-MODE
INPUT-HISTORY        = *ON(



--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Figure 22: Operand form for the MODIFY-SDF-OPTIONS command

The default value *NO of the GUIDANCE operand is changed to *MINIMUM. This is effected by entering “min”.

The specification in the NEXT line controls what happens when the screen is sent off ($\boxed{\text{DUE}}$ key). The preset entry *CONTINUE stands for *EXECUTE if the current form does not contain a structure-initiating operand value. In this case the INPUT-HISTORY operand is preset to “*ON(”, which means that when $\boxed{\text{DUE}}$ is pressed the subform of the *ON structure is displayed. The user can execute the command without switching to the operand subform by pressing the $\boxed{\text{F11}}$ key or entering *EXECUTE in the NEXT line.



```
DOMAIN   : SDF                                  COMMAND: MODIFY-SDF-OPTIONS
STRUCTURE: INPUT-HISTORY=*ON
OPERANDS : GUIDANCE=*MINIMUM,INPUT-HISTORY=*ON
--------------------------------------------------------------------------------
NUMBER-OF-INPUTS     = 20
PASSWORD-PROTECTION  = *YES












--------------------------------------------------------------------------------
NEXT = *EXECUTE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

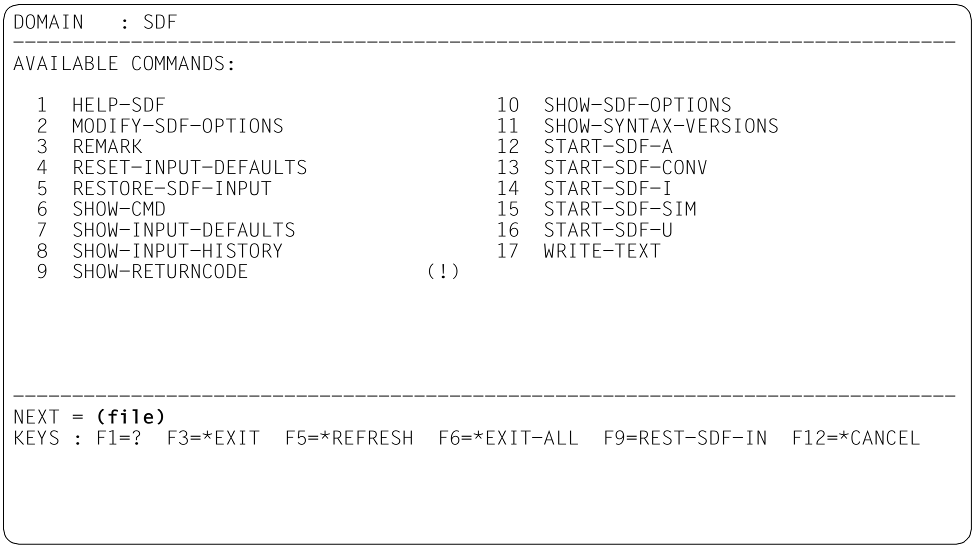
Figure 23: Operand subform for the *ON structure of the MODIFY-SDF-OPTIONS command

Sending off the form causes a switch to dialog with minimum user guidance.

5.4 Guided dialog

5.4.1 Minimum user guidance

The application domain SDF has been selected in temporarily guided dialog. This setting is retained after execution of the MODIFY-SDF-OPTIONS command. Consequently, SDF resumes the dialog by displaying the command menu for the application domain SDF.



```
DOMAIN    : SDF
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  HELP-SDF                              10  SHOW-SDF-OPTIONS
  2  MODIFY-SDF-OPTIONS                    11  SHOW-SYNTAX-VERSIONS
  3  REMARK                                12  START-SDF-A
  4  RESET-INPUT-DEFAULTS                  13  START-SDF-CONV
  5  RESTORE-SDF-INPUT                     14  START-SDF-I
  6  SHOW-CMD                              15  START-SDF-SIM
  7  SHOW-INPUT-DEFAULTS                   16  START-SDF-U
  8  SHOW-INPUT-HISTORY                    17  WRITE-TEXT
  9  SHOW-RETURNCODE               (!)





--------------------------------------------------------------------------------
NEXT = (file)
KEYS : F1=?  F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F9=REST-SDF-IN  F12=*CANCEL
```

Figure 24: Command menu for the application domain SDF

Entering “(file)” in the NEXT line changes the application domain setting. From now on the application domain FILE is set.

SDF displays the command menu for the application domain FILE.

```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  ADD-ALIAS-CATALOG-ENTRY            14  CREATE-TAPE-SET
  2  ADD-FILE-LINK                      15  DELETE-ALTERNATE-INDEX
  3  ADD-ISAM-POOL-LINK                 16  DELETE-FILE
  4  ADD-PASSWORD                       17  DELETE-ISAM-POOL
  5  CHANGE-FILE-LINK                   18  DELETE-SYSTEM-FILE
  6  CHECK-FILE-CONSISTENCY             19  DELETE-TAPE-SET
  7  CHECK-IMPORT-DISK-FILE             20  EDIT-FILE-ATTRIBUTES
  8  CONCATENATE-DISK-FILES             21  EDIT-FILE-LINK
  9  COPY-FILE                          22  EXPORT-FILE
 10  COPY-SYSTEM-FILE                   23  EXTEND-TAPE-SET
 11  CREATE-ALTERNATE-INDEX             24  HOLD-ALIAS-SUBSTITUTION       (!)
 12  CREATE-FILE                        25  IMPORT-FILE
 13  CREATE-ISAM-POOL                   26  LOAD-ALIAS-CATALOG

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F8=+ F9=REST-SDF-IN F12=*CANCEL
```

Figure 25: Command menu for the application domain FILE

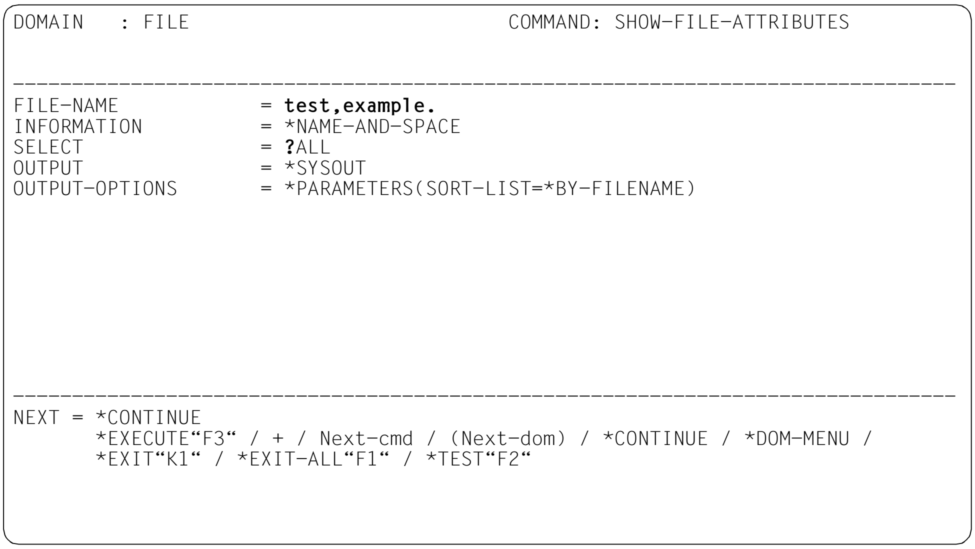
As the NEXT line is already preset with “+”, sending off the screen ($\boxed{\text{DUE}}$ key) causes the next page of the command menu to be output.

```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 27  LOCK-FILE-LINK                     40  SET-FILE-NAME-PREFIX
 28  MODIFY-ACS-OPTIONS                 41  SHOW-ACS-OPTIONS
 29  MODIFY-ALIAS-CATALOG-ENTRY         42  SHOW-ACS-SYSTEM-FILES
 30  MODIFY-FILE-ATTRIBUTES             43  SHOW-ALIAS-CATALOG-ENTRY
 31  PURGE-ALIAS-CATALOG          (!)   44  SHOW-CE-LOCK
 32  REMOVE-ALIAS-CATALOG-ENTRY         45  SHOW-FILE
 33  REMOVE-FILE-ALLOCATION-LOCKS       46  SHOW-FILE-ATTRIBUTES
 34  REMOVE-FILE-LINK                   47  SHOW-FILE-LINK
 35  REMOVE-ISAM-POOL-LINK              48  SHOW-FILE-LOCKS
 36  REMOVE-PASSWORD                    49  SHOW-FILE-NAME-PREFIX         (!)
 37  REPAIR-DISK-FILES                  50  SHOW-INDEX-ATTRIBUTES
 38  REPAIR-FILE-LOCKS                  51  SHOW-ISAM-POOL-ATTRIBUTES
 39  RESUME-ALIAS-SUBSTITUTION    (!)   52  SHOW-ISAM-POOL-LINK

--------------------------------------------------------------------------------
NEXT = 46
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F12=*CANCEL
```

Figure 26: Command menu for the application domain FILE

“46” causes the operand form for the SHOW-FILE-ATTRIBUTES command to be displayed.

SDF outputs the operand form for the SHOW-FILE-ATTRIBUTES command. The form is preset with the default values of the optional operands FILE-NAME, INFORMATION, SELECT and OUTPUT.

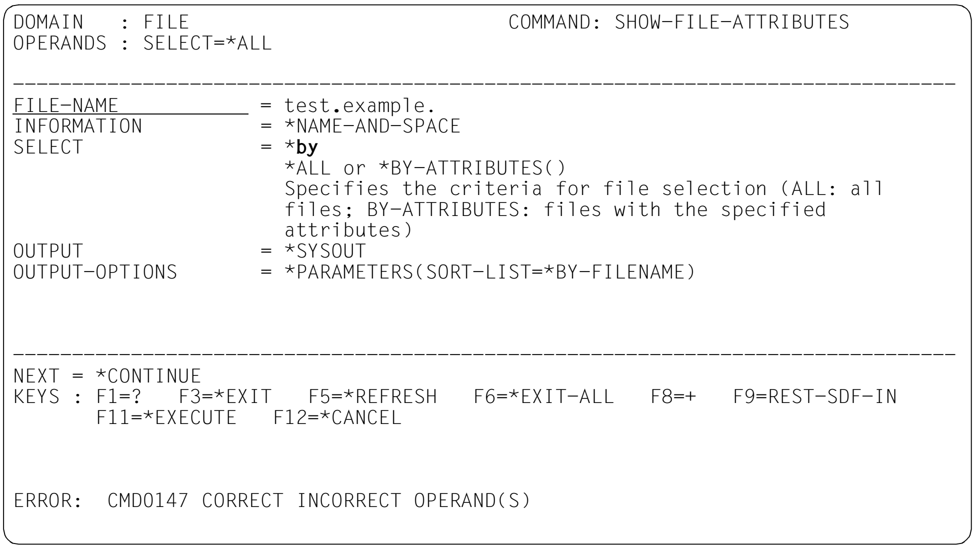


```
DOMAIN   : FILE                              COMMAND: SHOW-FILE-ATTRIBUTES


--------------------------------------------------------------------------------
FILE-NAME             = test,example.
INFORMATION           = *NAME-AND-SPACE
SELECT                = ?ALL
OUTPUT                = *SYSOUT
OUTPUT-OPTIONS        = *PARAMETERS(SORT-LIST=*BY-FILENAME)









--------------------------------------------------------------------------------
NEXT = *CONTINUE
       *EXECUTE“F3“ / + / Next-cmd / (Next-dom) / *CONTINUE / *DOM-MENU /
       *EXIT“K1“ / *EXIT-ALL“F1“ / *TEST“F2“
```

Figure 27: Operand form for the SHOW-FILE-ATTRIBUTES command

By mistake, the value of the FILE-NAME operand is given as “test,example.” (comma instead of period). Additional information on the SELECT operand is called up by entering “?”. The rest of the line, “ALL”, need in this case not be deleted. The specification in the NEXT line controls what happens when the screen is sent off ($\boxed{\text{DUE}}$ key). The default value *CONTINUE stands for *EXECUTE or +, and in this case causes SDF to display the operand form for the SHOW-FILE-ATTRIBUTES command with an explanation of the SELECT operand.

SDF displays the operand form for the SHOW-FILE-ATTRIBUTES command again. The form includes a request to correct the value of the FILE-NAME operand and contains additional information on the SELECT operand. The parentheses following the *BY-ATTRIBUTES value indicate that *BY-ATTRIBUTES initiates a structure.

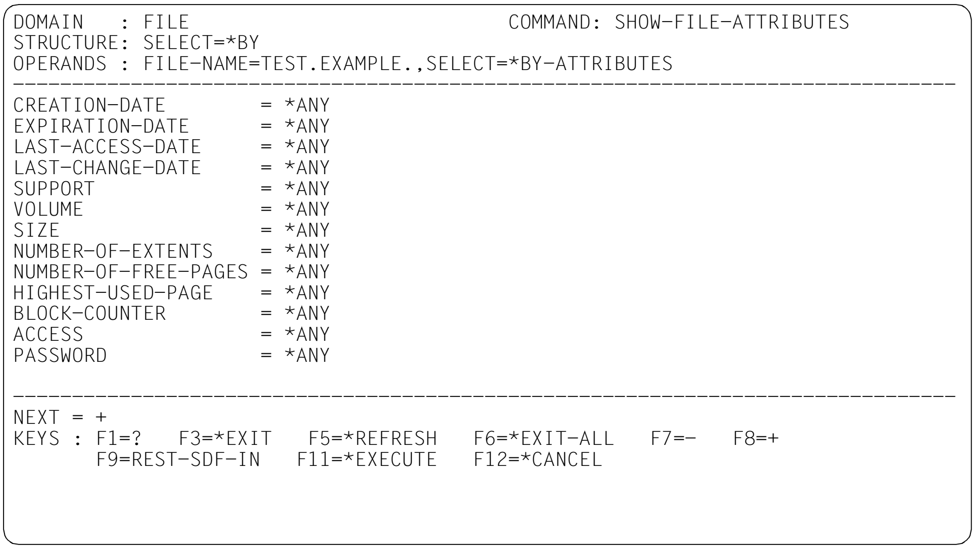


```
DOMAIN   : FILE                           COMMAND: SHOW-FILE-ATTRIBUTES
OPERANDS : SELECT=*ALL

--------------------------------------------------------------------------------
FILE-NAME            = test.example.
INFORMATION          = *NAME-AND-SPACE
SELECT               = *by
                       *ALL or *BY-ATTRIBUTES()
                       Specifies the criteria for file selection (ALL: all
                       files; BY-ATTRIBUTES: files with the specified
                       attributes)
OUTPUT               = *SYSOUT
OUTPUT-OPTIONS       = *PARAMETERS(SORT-LIST=*BY-FILENAME)



--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL



ERROR:  CMD0147 CORRECT INCORRECT OPERAND(S)
```

Figure 28: Operand form for the SHOW-FILE-ATTRIBUTES command

The FILE-NAME operand value is corrected. The default value *ALL of the SELECT operand is overwritten with “*by” (BY-ATTRIBUTES). Sending off the form causes SDF to display a subform for the *BY-ATTRIBUTES structure.

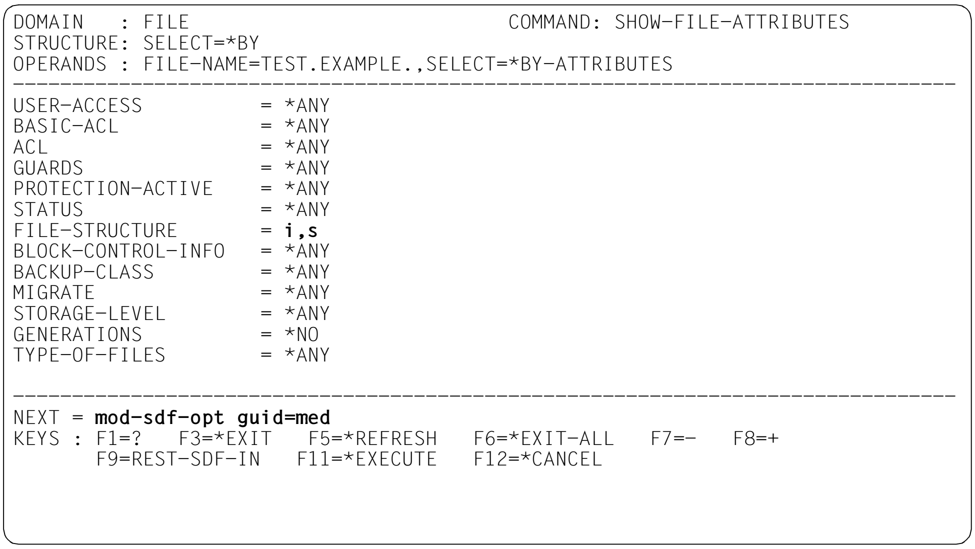
SDF displays the subform for the *BY-ATTRIBUTES structure.



```
DOMAIN   : FILE                           COMMAND: SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES
--------------------------------------------------------------------------------
CREATION-DATE        = *ANY
EXPIRATION-DATE      = *ANY
LAST-ACCESS-DATE     = *ANY
LAST-CHANGE-DATE     = *ANY
SUPPORT              = *ANY
VOLUME               = *ANY
SIZE                 = *ANY
NUMBER-OF-EXTENTS    = *ANY
NUMBER-OF-FREE-PAGES = *ANY
HIGHEST-USED-PAGE    = *ANY
BLOCK-COUNTER        = *ANY
ACCESS               = *ANY
PASSWORD             = *ANY

--------------------------------------------------------------------------------
NEXT = +
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL
```

Figure 29: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

As the NEXT line is already preset with “+”, sending off the screen (DUE key) causes the next page of the operand form to be output.



```
DOMAIN    : FILE                          COMMAND: SHOW-FILE-ATTRIBUTES
STRUCTURE: SELECT=*BY
OPERANDS : FILE-NAME=TEST.EXAMPLE.,SELECT=*BY-ATTRIBUTES
--------------------------------------------------------------------------------
USER-ACCESS          = *ANY
BASIC-ACL            = *ANY
ACL                  = *ANY
GUARDS               = *ANY
PROTECTION-ACTIVE    = *ANY
STATUS               = *ANY
FILE-STRUCTURE       = i,s
BLOCK-CONTROL-INFO   = *ANY
BACKUP-CLASS         = *ANY
MIGRATE              = *ANY
STORAGE-LEVEL        = *ANY
GENERATIONS          = *NO
TYPE-OF-FILES        = *ANY

--------------------------------------------------------------------------------
NEXT = mod-sdf-opt guid=med
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F11=*EXECUTE   F12=*CANCEL
```

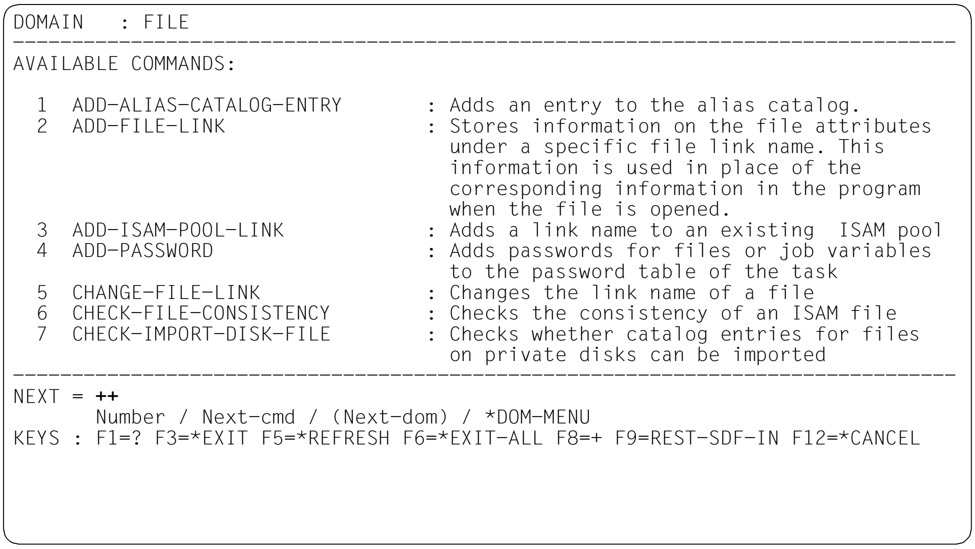
Figure 30: Operand subform for the *BY-ATTRIBUTES structure of the SHOW-FILE-ATTRIBUTES command

Entering “i,s” assigns the FILE-STRUCTURE operand the values ISAM and SAM. The parentheses normally required for list entries may be omitted here. Any “Next-cmd” command may be entered in the NEXT line, regardless of the application domain, and it will be executed after execution of the SHOW-FILE-ATTRIBUTES command. The “Next-cmd” entered is the MODIFY-SDF-OPTIONS command (“mod-sdf-opt guid=med”), which causes a switch to dialog with medium user guidance. Sending off the form first causes the SHOW-FILE-ATTRIBUTES command to be executed. The system displays the required information. Then the user guidance level is changed. The dialog is resumed by pressing the transmit (DUE) key.

```
%        33 :2OSG:$USER1.TEST.EXAMPLE.1
%        72 :2OSG:$USER1.TEST.EXAMPLE.2
%        48 :2OSG:$USER1.TEST.EXAMPLE.3
%         3 :2OSG:$USER1.TEST.EXAMPLE.4
%:2OSG: PUBLIC:      4 FILES RES=      156 FRE=       30 REL=        9 PAGES
%PLEASE ACKNOWLEDGE
```

5.4.2 Medium user guidance

When the transmit (DUE) key is pressed, SDF again displays the command menu for the application domain FILE. Under medium user guidance this comprises a number of pages. The first page is displayed. Unlike the menu displayed by SDF under minimum user guidance, this menu includes explanations of the commands (help texts). In all higher guidance levels the possible control statements are also displayed after the NEXT line in the Styleguide mode set here.

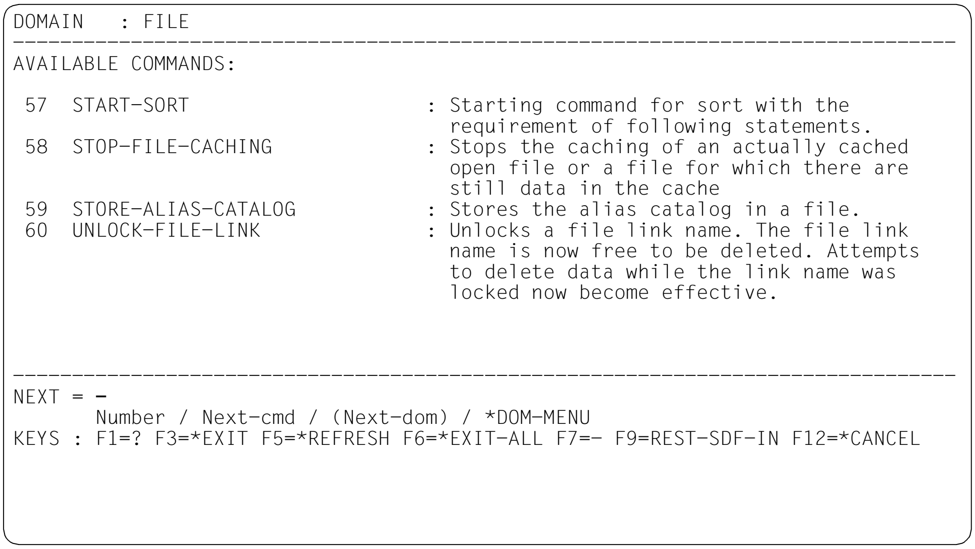


```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  ADD-ALIAS-CATALOG-ENTRY       : Adds an entry to the alias catalog.
  2  ADD-FILE-LINK                 : Stores information on the file attributes
                                     under a specific file link name. This
                                     information is used in place of the
                                     corresponding information in the program
                                     when the file is opened.
  3  ADD-ISAM-POOL-LINK            : Adds a link name to an existing  ISAM pool
  4  ADD-PASSWORD                  : Adds passwords for files or job variables
                                     to the password table of the task
  5  CHANGE-FILE-LINK              : Changes the link name of a file
  6  CHECK-FILE-CONSISTENCY        : Checks the consistency of an ISAM file
  7  CHECK-IMPORT-DISK-FILE        : Checks whether catalog entries for files
                                     on private disks can be imported
--------------------------------------------------------------------------------
NEXT = ++
       Number / Next-cmd / (Next-dom) / *DOM-MENU
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F8=+ F9=REST-SDF-IN F12=*CANCEL
```

Figure 31: Command menu for the application domain FILE

Entering “++” in the NEXT line causes SDF to scroll to the end of the menu.

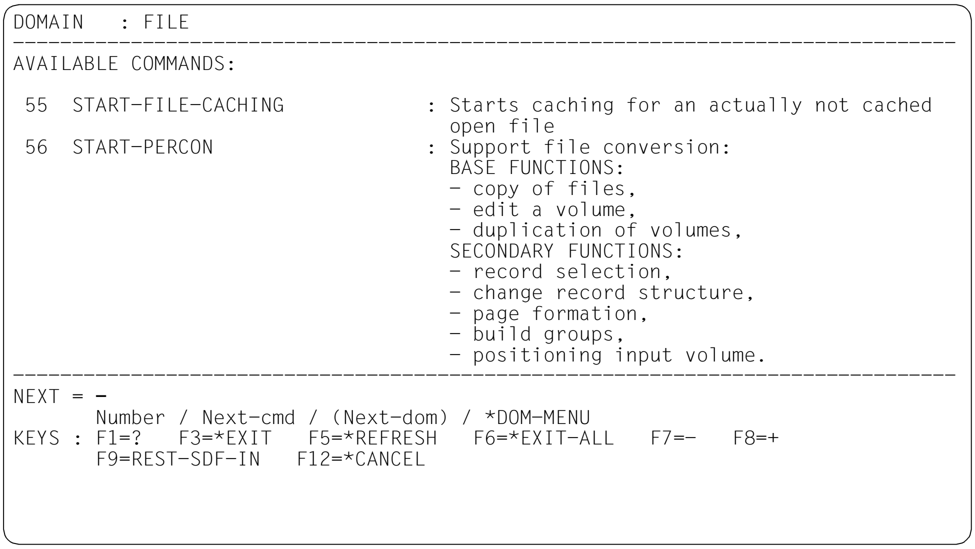
SDF displays the last page of the command menu for the application domain FILE.



```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 57  START-SORT                       : Starting command for sort with the
                                        requirement of following statements.
 58  STOP-FILE-CACHING                : Stops the caching of an actually cached
                                        open file or a file for which there are
                                        still data in the cache
 59  STORE-ALIAS-CATALOG              : Stores the alias catalog in a file.
 60  UNLOCK-FILE-LINK                 : Unlocks a file link name. The file link
                                        name is now free to be deleted. Attempts
                                        to delete data while the link name was
                                        locked now become effective.



--------------------------------------------------------------------------------
NEXT = -
       Number / Next-cmd / (Next-dom) / *DOM-MENU
KEYS : F1=? F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F7=- F9=REST-SDF-IN F12=*CANCEL
```

Figure 32: Command menu for the application domain FILE

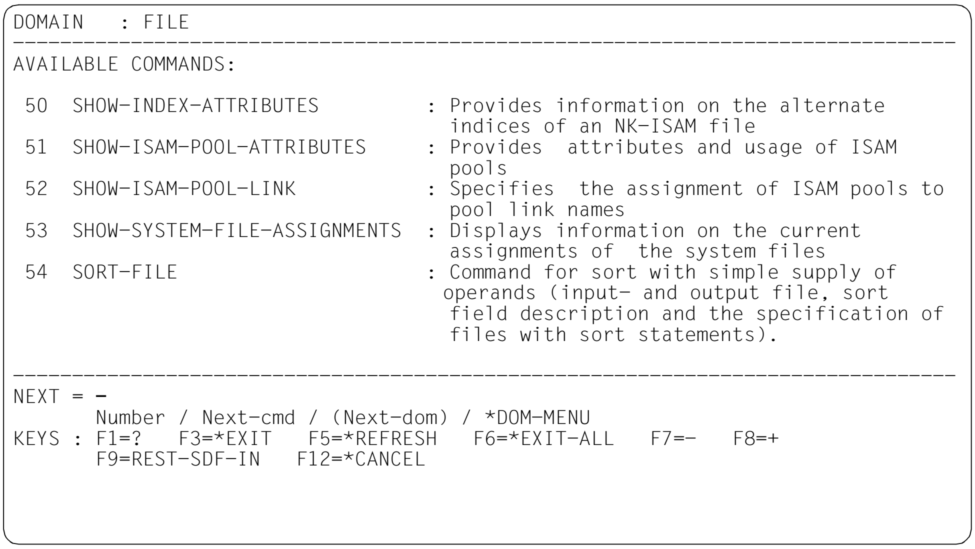
Entering “–” (alternative: F7 key) causes SDF to scroll back one page in the menu.



```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 55  START-FILE-CACHING               : Starts caching for an actually not cached
                                        open file
 56  START-PERCON                     : Support file conversion:
                                        BASE FUNCTIONS:
                                        - copy of files,
                                        - edit a volume,
                                        - duplication of volumes,
                                        SECONDARY FUNCTIONS:
                                        - record selection,
                                        - change record structure,
                                        - page formation,
                                        - build groups,
                                        - positioning input volume.
--------------------------------------------------------------------------------
NEXT = -
       Number / Next-cmd / (Next-dom) / *DOM-MENU
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F12=*CANCEL
```

Figure 33: Command menu for the application domain FILE

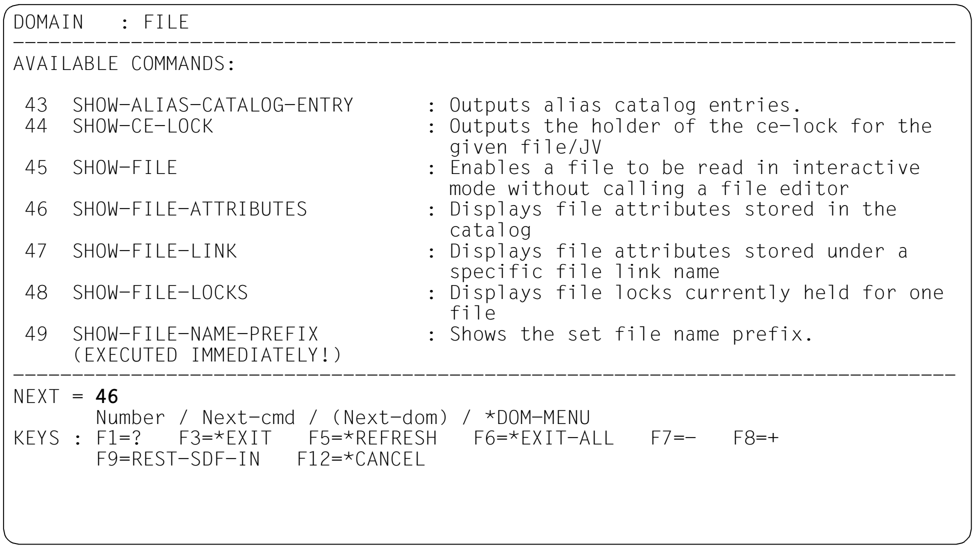
Entering “–” (alternative: F7 key) causes SDF to scroll back one page in the menu.



```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 50  SHOW-INDEX-ATTRIBUTES         : Provides information on the alternate
                                     indices of an NK-ISAM file
 51  SHOW-ISAM-POOL-ATTRIBUTES     : Provides  attributes and usage of ISAM
                                     pools
 52  SHOW-ISAM-POOL-LINK           : Specifies  the assignment of ISAM pools to
                                     pool link names
 53  SHOW-SYSTEM-FILE-ASSIGNMENTS  : Displays information on the current
                                     assignments of  the system files
 54  SORT-FILE                     : Command for sort with simple supply of
                                    operands (input- and output file, sort
                                     field description and the specification of
                                     files with sort statements).

--------------------------------------------------------------------------------
NEXT = -
       Number / Next-cmd / (Next-dom) / *DOM-MENU
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F12=*CANCEL
```

Figure 34: Command menu for the application domain FILE

Entering “–”(alternative: F7 key) causes SDF to scroll back one page in the menu.

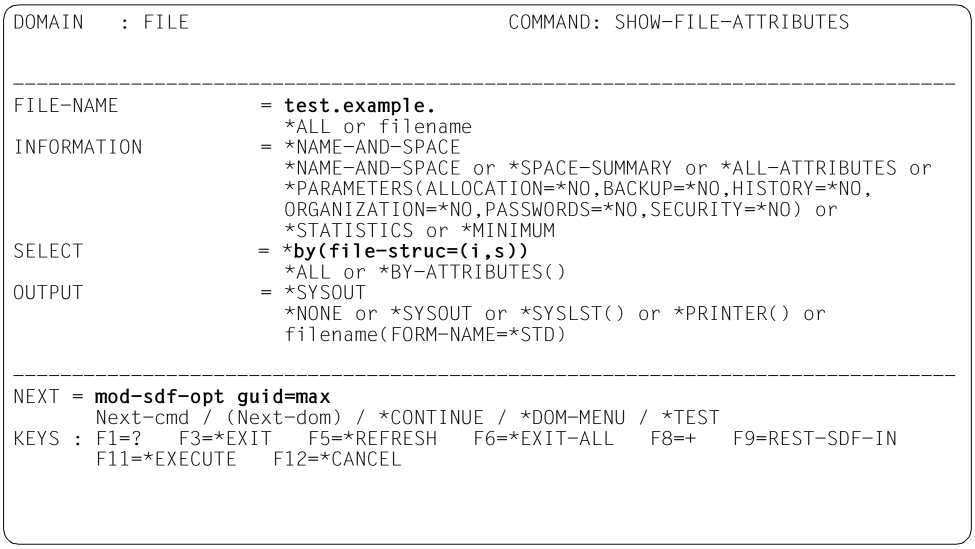


```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 43  SHOW-ALIAS-CATALOG-ENTRY      : Outputs alias catalog entries.
 44  SHOW-CE-LOCK                  : Outputs the holder of the ce-lock for the
                                     given file/JV
 45  SHOW-FILE                     : Enables a file to be read in interactive
                                     mode without calling a file editor
 46  SHOW-FILE-ATTRIBUTES          : Displays file attributes stored in the
                                     catalog
 47  SHOW-FILE-LINK                : Displays file attributes stored under a
                                     specific file link name
 48  SHOW-FILE-LOCKS               : Displays file locks currently held for one
                                     file
 49  SHOW-FILE-NAME-PREFIX         : Shows the set file name prefix.
     (EXECUTED IMMEDIATELY!)
--------------------------------------------------------------------------------
NEXT = 46
       Number / Next-cmd / (Next-dom) / *DOM-MENU
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F7=-   F8=+
       F9=REST-SDF-IN   F12=*CANCEL
```

Figure 35: Command menu for the application domain FILE

Entering “46” calls up the operand form for the SHOW-FILE-ATTRIBUTES command.

SDF displays the operand form for the SHOW-FILE-ATTRIBUTES command. Unlike the form for minimum user guidance, this form includes data on the permissible operand values. The parentheses following the *BY-ATTRIBUTES value indicate that *BY-ATTRIBUTES initiates a structure with its own form.



```
DOMAIN   : FILE                          COMMAND: SHOW-FILE-ATTRIBUTES


--------------------------------------------------------------------------------
FILE-NAME            = test.example.
                       *ALL or filename
INFORMATION          = *NAME-AND-SPACE
                       *NAME-AND-SPACE or *SPACE-SUMMARY or *ALL-ATTRIBUTES or
                       *PARAMETERS(ALLOCATION=*NO,BACKUP=*NO,HISTORY=*NO,
                       ORGANIZATION=*NO,PASSWORDS=*NO,SECURITY=*NO) or
                       *STATISTICS or *MINIMUM
SELECT               = *by(file-struc=(i,s))
                       *ALL or *BY-ATTRIBUTES()
OUTPUT               = *SYSOUT
                       *NONE or *SYSOUT or *SYSLST() or *PRINTER() or
                       filename(FORM-NAME=*STD)

--------------------------------------------------------------------------------
NEXT = mod-sdf-opt guid=max
       Next-cmd / (Next-dom) / *CONTINUE / *DOM-MENU / *TEST
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

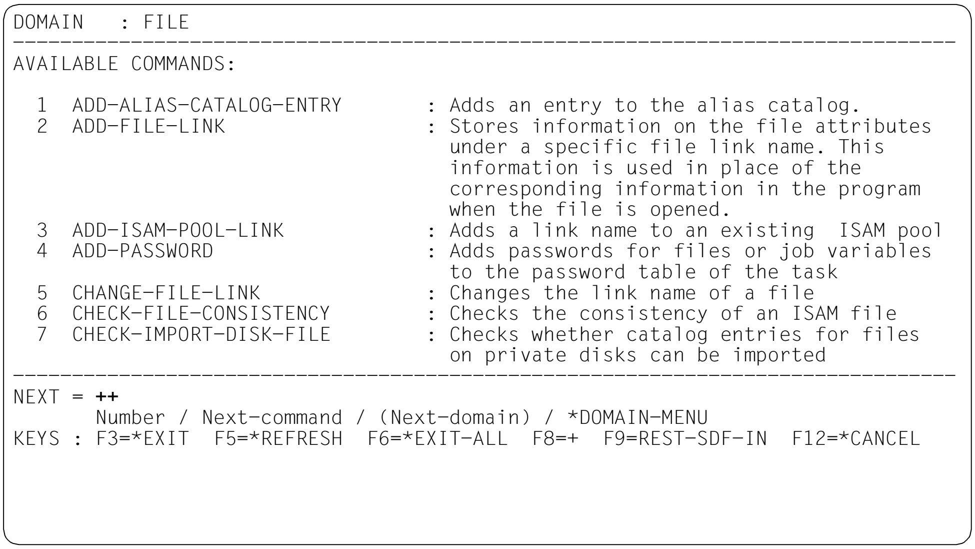
Figure 36: Operand form for the SHOW-FILE-ATTRIBUTES command

“test.example.” is entered as the FILE-NAME operand value. The default value ALL of the SELECT operand is overwritten with “by” (BY-ATTRIBUTES). Display of a subform for the associated structure is suppressed by the parentheses. Entering “file-struc=(i,s)” assigns the FILE-STRUCTURE operand the values *ISAM and *SAM. It is essential here to enclose the two values in parentheses: “(i,s)”. Any “Next-cmd” command may be entered in the NEXT line, regardless of the application domain, and it will be executed after execution of the SHOW-FILE-ATTRIBUTES command. The “Next-cmd” entered here is the MODIFY-SDF-OPTIONS command (“mod-sdf-opt guid=max”). This causes a switch to dialog with maximum user guidance. Sending off the form first causes the SHOW-FILE-ATTRIBUTES command to be executed. The system displays the required information. Then the user guidance level is changed. The dialog is resumed by pressing the transmit ($\boxed{\text{DUE}}$) key.

```
%        33 :2OSG:$USER1.TEST.EXAMPLE.1
%        72 :2OSG:$USER1.TEST.EXAMPLE.2
%        48 :2OSG:$USER1.TEST.EXAMPLE.3
%         3 :2OSG:$USER1.TEST.EXAMPLE.4
%:2OSG: PUBLIC:      4 FILES RES=      156 FRE=       30 REL=        9 PAGES
%PLEASE ACKNOWLEDGE
```

5.4.3 Maximum user guidance

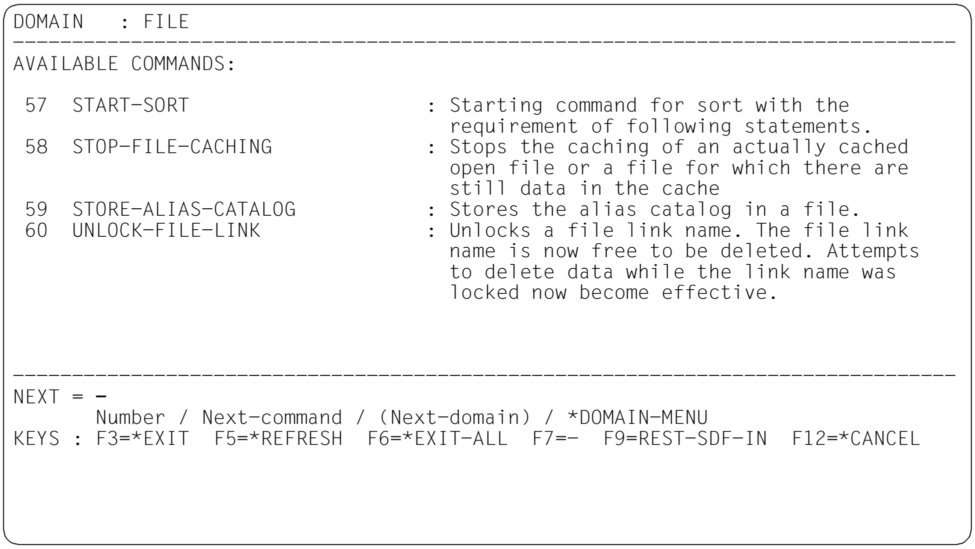
When the transmit (DUE) key is pressed, SDF again displays the command menu for the application domain FILE, which differs from the menu for medium guidance in minor details only.



```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  ADD-ALIAS-CATALOG-ENTRY       : Adds an entry to the alias catalog.
  2  ADD-FILE-LINK                 : Stores information on the file attributes
                                     under a specific file link name. This
                                     information is used in place of the
                                     corresponding information in the program
                                     when the file is opened.
  3  ADD-ISAM-POOL-LINK            : Adds a link name to an existing  ISAM pool
  4  ADD-PASSWORD                  : Adds passwords for files or job variables
                                     to the password table of the task
  5  CHANGE-FILE-LINK              : Changes the link name of a file
  6  CHECK-FILE-CONSISTENCY        : Checks the consistency of an ISAM file
  7  CHECK-IMPORT-DISK-FILE        : Checks whether catalog entries for files
                                     on private disks can be imported
--------------------------------------------------------------------------------
NEXT = ++
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F8=+  F9=REST-SDF-IN  F12=*CANCEL
```

Figure 37: Command menu for the application domain FILE

Entering “++” in the NEXT line causes SDF to scroll to the end of the menu. SDF displays the last page of the command menu for the application domain FILE.

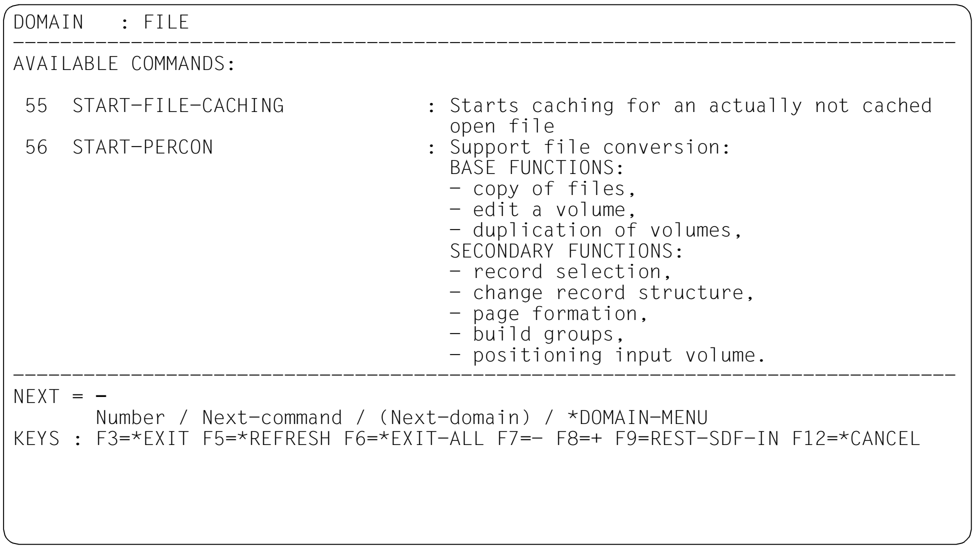


```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 57  START-SORT                    : Starting command for sort with the
                                     requirement of following statements.
 58  STOP-FILE-CACHING             : Stops the caching of an actually cached
                                     open file or a file for which there are
                                     still data in the cache
 59  STORE-ALIAS-CATALOG           : Stores the alias catalog in a file.
 60  UNLOCK-FILE-LINK              : Unlocks a file link name. The file link
                                     name is now free to be deleted. Attempts
                                     to delete data while the link name was
                                     locked now become effective.




--------------------------------------------------------------------------------
NEXT = -
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F7=-  F9=REST-SDF-IN  F12=*CANCEL
```

Figure 38: Command menu for the application domain FILE

Entering “–” (alternative: F7 key) causes SDF to scroll back one screen in the command menu.

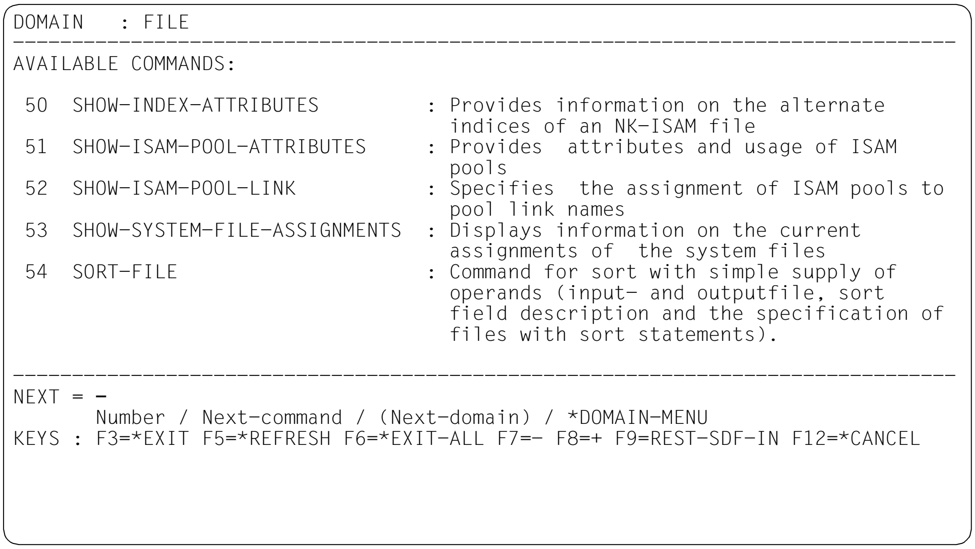
SDF displays the last page but one of the command menu for the application domain FILE.



```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 55  START-FILE-CACHING            : Starts caching for an actually not cached
                                     open file
 56  START-PERCON                  : Support file conversion:
                                     BASE FUNCTIONS:
                                     - copy of files,
                                     - edit a volume,
                                     - duplication of volumes,
                                     SECONDARY FUNCTIONS:
                                     - record selection,
                                     - change record structure,
                                     - page formation,
                                     - build groups,
                                     - positioning input volume.
--------------------------------------------------------------------------------
NEXT = -
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F7=- F8=+ F9=REST-SDF-IN F12=*CANCEL
```

Figure 39: Command menu for the application domain FILE

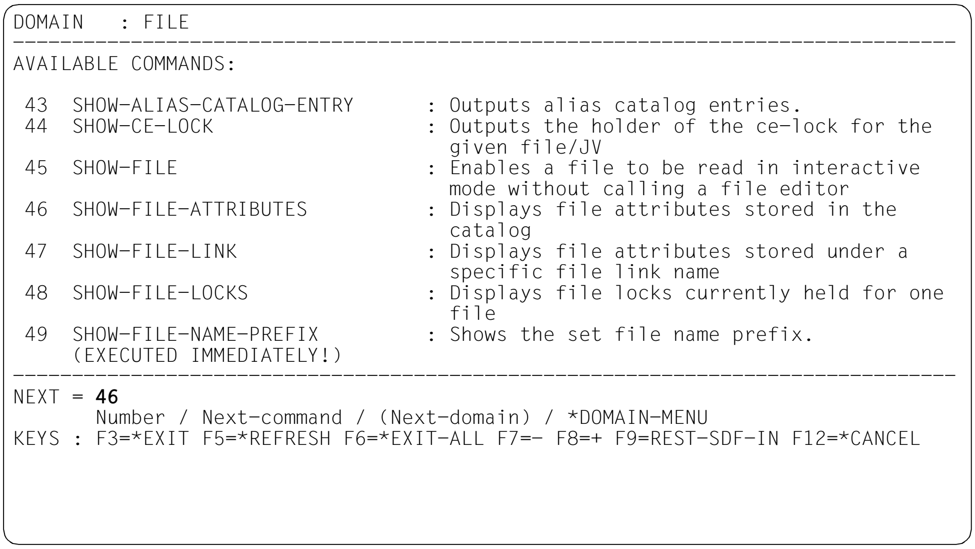
Entering “–” (alternative: F7 key) causes SDF to scroll back one screen in the command menu. SDF displays the desired page of the command menu for the application domain FILE.



```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 50  SHOW-INDEX-ATTRIBUTES         : Provides information on the alternate
                                     indices of an NK-ISAM file
 51  SHOW-ISAM-POOL-ATTRIBUTES     : Provides  attributes and usage of ISAM
                                     pools
 52  SHOW-ISAM-POOL-LINK           : Specifies  the assignment of ISAM pools to
                                     pool link names
 53  SHOW-SYSTEM-FILE-ASSIGNMENTS  : Displays information on the current
                                     assignments of  the system files
 54  SORT-FILE                     : Command for sort with simple supply of
                                     operands (input- and outputfile, sort
                                     field description and the specification of
                                     files with sort statements).

--------------------------------------------------------------------------------
NEXT = -
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F7=- F8=+ F9=REST-SDF-IN F12=*CANCEL
```

Figure 40: Command menu for the application domain FILE

Entering “–” (alternative: F7 key) causes SDF to scroll back one screen in the command menu.



```
DOMAIN   : FILE
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

 43  SHOW-ALIAS-CATALOG-ENTRY      : Outputs alias catalog entries.
 44  SHOW-CE-LOCK                  : Outputs the holder of the ce-lock for the
                                     given file/JV
 45  SHOW-FILE                     : Enables a file to be read in interactive
                                     mode without calling a file editor
 46  SHOW-FILE-ATTRIBUTES          : Displays file attributes stored in the
                                     catalog
 47  SHOW-FILE-LINK                : Displays file attributes stored under a
                                     specific file link name
 48  SHOW-FILE-LOCKS               : Displays file locks currently held for one
                                     file
 49  SHOW-FILE-NAME-PREFIX         : Shows the set file name prefix.
     (EXECUTED IMMEDIATELY!)
--------------------------------------------------------------------------------
NEXT = 46
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F7=- F8=+ F9=REST-SDF-IN F12=*CANCEL
```

Figure 41: Command menu for the application domain FILE

Entering “46” calls up the operand form for the SHOW-FILE-ATTRIBUTES command.

SDF displays the operand form for the SHOW-FILE-ATTRIBUTES command. Unlike the form for medium user guidance, this form includes additional information on the permissible operand values, e.g. minimum and maximum length, and explanations of the operands (help texts). Since under maximum guidance the *PARAMETERS structure also has its own subform, the operands of this structure are not listed in the form. The sets of parentheses following the values *PARAMETERS and *BY-ATTRIBUTES indicate that these values initiate a structure. As the form for maximum guidance is more comprehensive, it takes up two pages.

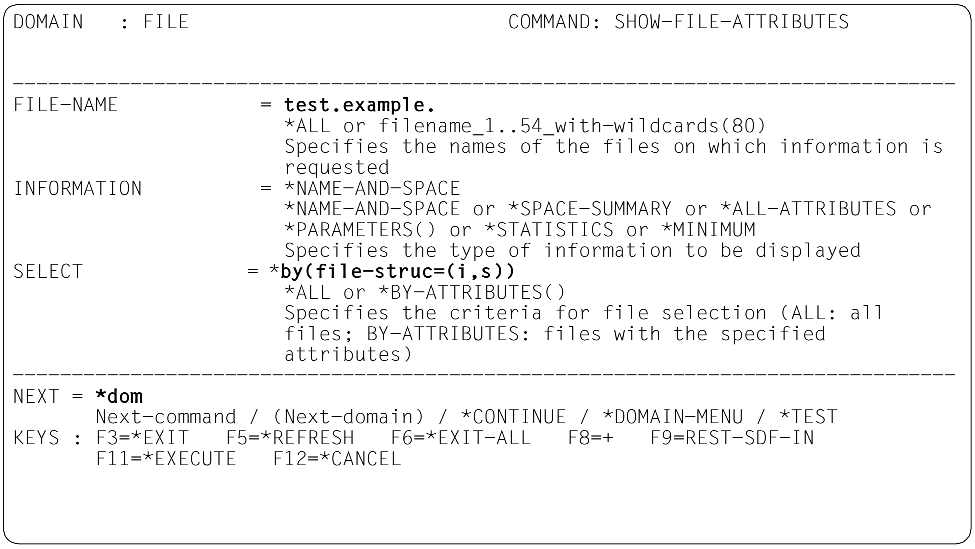
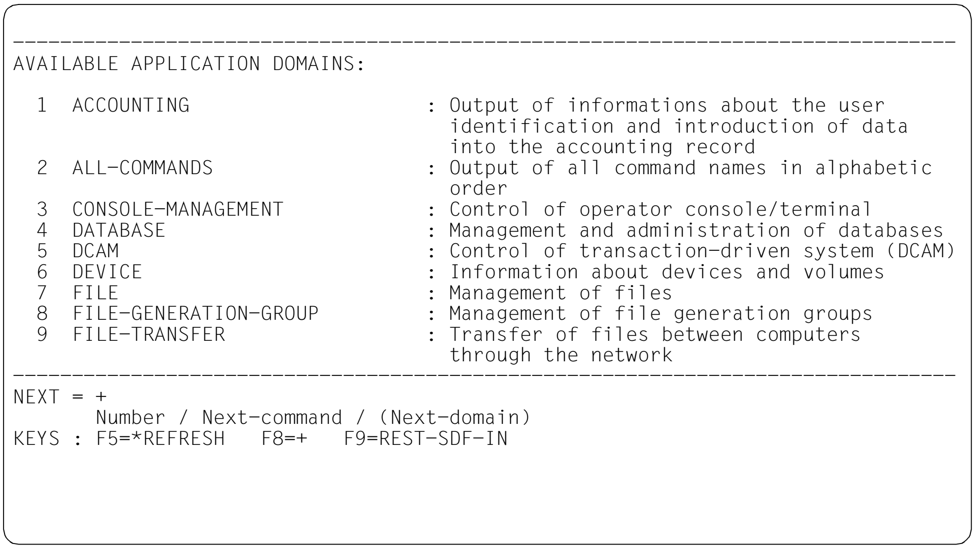

```
DOMAIN   : FILE                           COMMAND: SHOW-FILE-ATTRIBUTES


--------------------------------------------------------------------------------
FILE-NAME            = test.example.
                       *ALL or filename_1..54_with-wildcards(80)
                       Specifies the names of the files on which information is
                       requested
INFORMATION          = *NAME-AND-SPACE
                       *NAME-AND-SPACE or *SPACE-SUMMARY or *ALL-ATTRIBUTES or
                       *PARAMETERS() or *STATISTICS or *MINIMUM
                       Specifies the type of information to be displayed
SELECT              = *by(file-struc=(i,s))
                       *ALL or *BY-ATTRIBUTES()
                       Specifies the criteria for file selection (ALL: all
                       files; BY-ATTRIBUTES: files with the specified
                       attributes)
--------------------------------------------------------------------------------
NEXT = *dom
       Next-command / (Next-domain) / *CONTINUE / *DOMAIN-MENU / *TEST
KEYS : F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Figure 42: Operand form for the SHOW-FILE-ATTRIBUTES command

“test.example.” is entered as the FILE-NAME operand value. The default value *ALL of the SELECT operand is overwritten with “by” (BY-ATTRIBUTES). Display of a subform for the associated structure is suppressed by the parentheses. Entering “file-struc=(i,s)” assigns the FILE-STRUCTURE operand the values ISAM and SAM. The application domain menu may be called up by entering *DOMAIN-MENU in the NEXT line; to do this, it is sufficient to enter “*dom” and then press the LZF key. “*dom” is an unambiguous abbreviation. The $\boxed{\text{LZF}}$ key (from the German for “delete character string”) deletes the characters “TINUE” remaining from the default setting *CONTINUE. “*domTINUE” would be interpreted by SDF as an incorrect entry. Sending off the form initiates execution of the SHOW-FILE-ATTRIBUTES command. Sending off the form first causes the command to be executed; the system displays the required information. When the transmit ($\boxed{\text{DUE}}$) key is pressed, SDF displays the application domain menu.

```
%        33 :2OSG:$USER1.TEST.EXAMPLE.1
%        72 :2OSG:$USER1.TEST.EXAMPLE.2
%        48 :2OSG:$USER1.TEST.EXAMPLE.3
%         3 :2OSG:$USER1.TEST.EXAMPLE.4
%:2OSG: PUBLIC:      4 FILES RES=      156 FRE=       30 REL=        9 PAGES
%PLEASE ACKNOWLEDGE
```

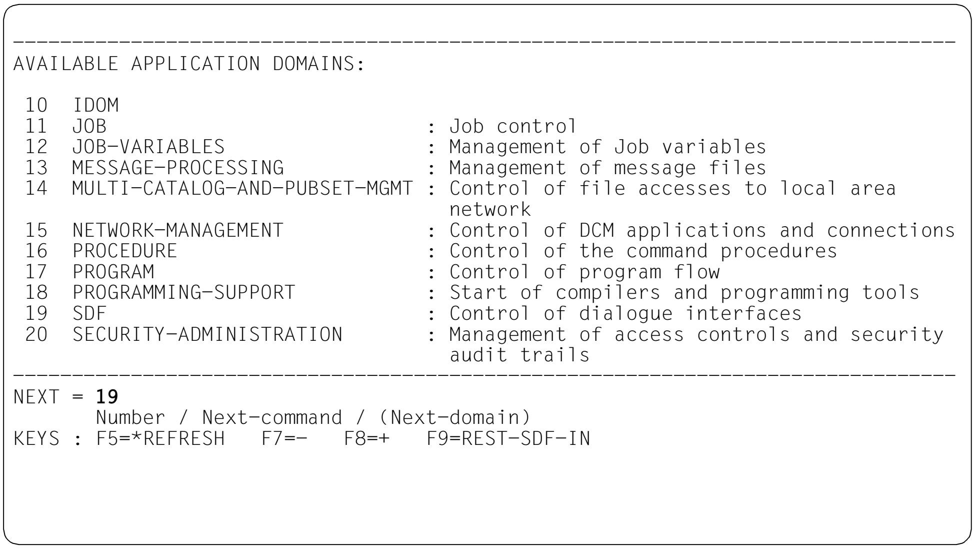
SDF displays the application domain menu, which differs from the menu for medium user guidance in minor details only. Unlike the menu for minimum guidance, it includes explanations of the application domains (help texts).



```
--------------------------------------------------------------------------------
AVAILABLE APPLICATION DOMAINS:

  1  ACCOUNTING                      : Output of informations about the user
                                       identification and introduction of data
                                       into the accounting record
  2  ALL-COMMANDS                    : Output of all command names in alphabetic
                                       order
  3  CONSOLE-MANAGEMENT              : Control of operator console/terminal
  4  DATABASE                        : Management and administration of databases
  5  DCAM                            : Control of transaction-driven system (DCAM)
  6  DEVICE                          : Information about devices and volumes
  7  FILE                            : Management of files
  8  FILE-GENERATION-GROUP           : Management of file generation groups
  9  FILE-TRANSFER                   : Transfer of files between computers
                                       through the network
--------------------------------------------------------------------------------
NEXT = +
       Number / Next-command / (Next-domain)
KEYS : F5=*REFRESH   F8=+   F9=REST-SDF-IN
```

Figure 43: Application domain menu

As the NEXT line is already preset with “+”, sending off the screen (DUE key) causes the next page of the application domain menu to be output.

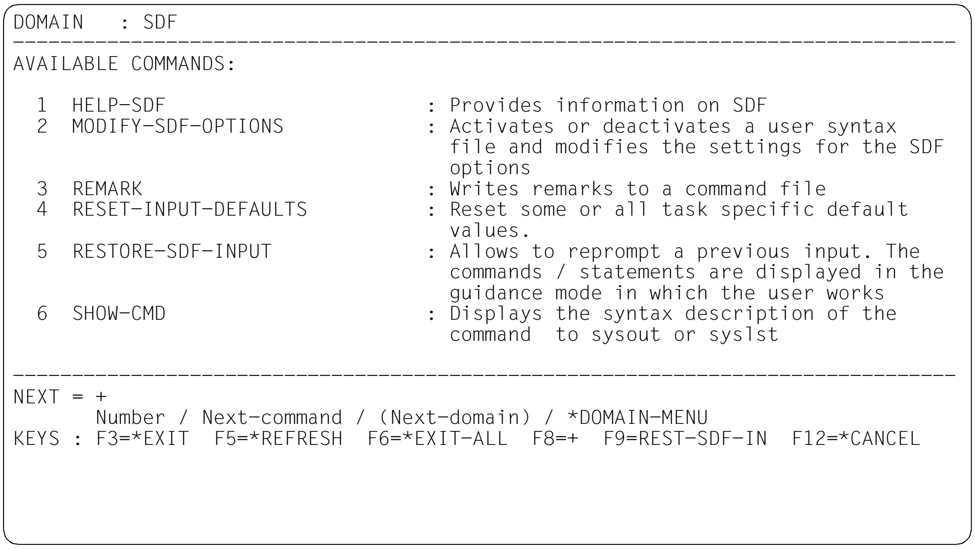


```
--------------------------------------------------------------------------------
AVAILABLE APPLICATION DOMAINS:

 10  IDOM
 11  JOB                           : Job control
 12  JOB-VARIABLES                 : Management of Job variables
 13  MESSAGE-PROCESSING            : Management of message files
 14  MULTI-CATALOG-AND-PUBSET-MGMT : Control of file accesses to local area
                                     network
 15  NETWORK-MANAGEMENT            : Control of DCM applications and connections
 16  PROCEDURE                     : Control of the command procedures
 17  PROGRAM                       : Control of program flow
 18  PROGRAMMING-SUPPORT           : Start of compilers and programming tools
 19  SDF                           : Control of dialogue interfaces
 20  SECURITY-ADMINISTRATION       : Management of access controls and security
                                     audit trails
--------------------------------------------------------------------------------
NEXT = 19
       Number / Next-command / (Next-domain)
KEYS : F5=*REFRESH   F7=-   F8=+   F9=REST-SDF-IN
```

Figure 44: Application domain menu

Entering “19” selects the application domain SDF.

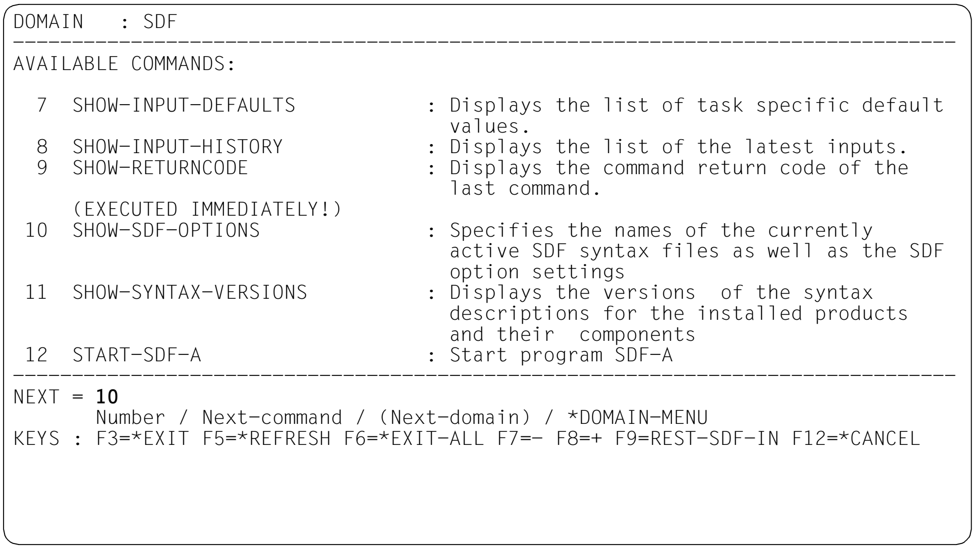
SDF displays the command menu for the application domain SDF:



```
DOMAIN   : SDF
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  HELP-SDF                      : Provides information on SDF
  2  MODIFY-SDF-OPTIONS            : Activates or deactivates a user syntax
                                     file and modifies the settings for the SDF
                                     options
  3  REMARK                        : Writes remarks to a command file
  4  RESET-INPUT-DEFAULTS          : Reset some or all task specific default
                                     values.
  5  RESTORE-SDF-INPUT             : Allows to reprompt a previous input. The
                                     commands / statements are displayed in the
                                     guidance mode in which the user works
  6  SHOW-CMD                      : Displays the syntax description of the
                                     command  to sysout or syslst

--------------------------------------------------------------------------------
NEXT = +
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F8=+  F9=REST-SDF-IN  F12=*CANCEL
```

Figure 45: Command menu for the application domain SDF

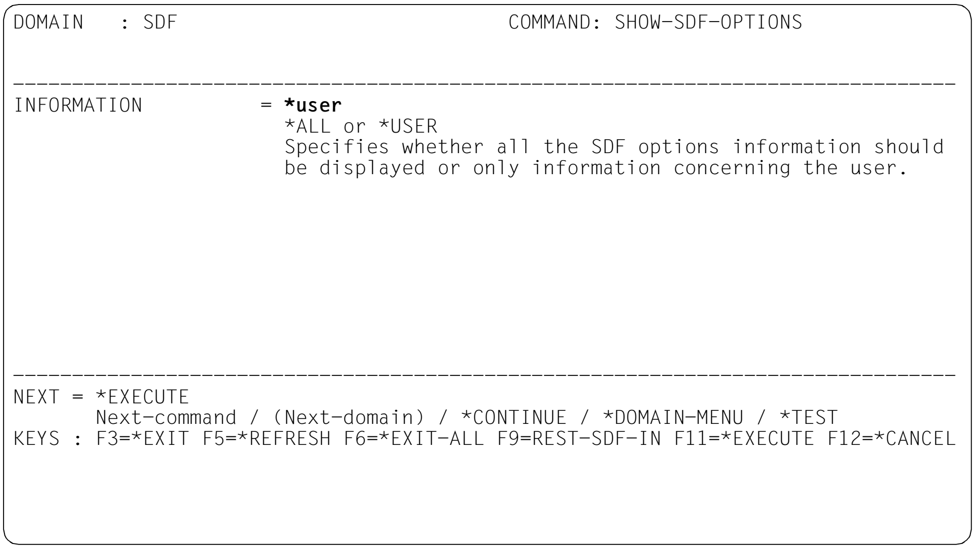
Entering “+” (or F8 key) causes SDF to scroll to the next screen in the command menu.



```
DOMAIN   : SDF
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  7  SHOW-INPUT-DEFAULTS          : Displays the list of task specific default
                                    values.
  8  SHOW-INPUT-HISTORY           : Displays the list of the latest inputs.
  9  SHOW-RETURNCODE              : Displays the command return code of the
                                    last command.
     (EXECUTED IMMEDIATELY!)
 10  SHOW-SDF-OPTIONS             : Specifies the names of the currently
                                    active SDF syntax files as well as the SDF
                                    option settings
 11  SHOW-SYNTAX-VERSIONS         : Displays the versions  of the syntax
                                    descriptions for the installed products
                                    and their  components
 12  START-SDF-A                  : Start program SDF-A
--------------------------------------------------------------------------------
NEXT = 10
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F7=- F8=+ F9=REST-SDF-IN F12=*CANCEL
```

Figure 46: Command menu for the application domain SDF

Entering “10” requests the operand form for the SHOW-SDF-OPTIONS command.



```
DOMAIN   : SDF                          COMMAND: SHOW-SDF-OPTIONS


--------------------------------------------------------------------------------
INFORMATION          = *user
                       *ALL or *USER
                       Specifies whether all the SDF options information should
                       be displayed or only information concerning the user.









--------------------------------------------------------------------------------
NEXT = *EXECUTE
       Next-command / (Next-domain) / *CONTINUE / *DOMAIN-MENU / *TEST
KEYS : F3=*EXIT F5=*REFRESH F6=*EXIT-ALL F9=REST-SDF-IN F11=*EXECUTE F12=*CANCEL
```

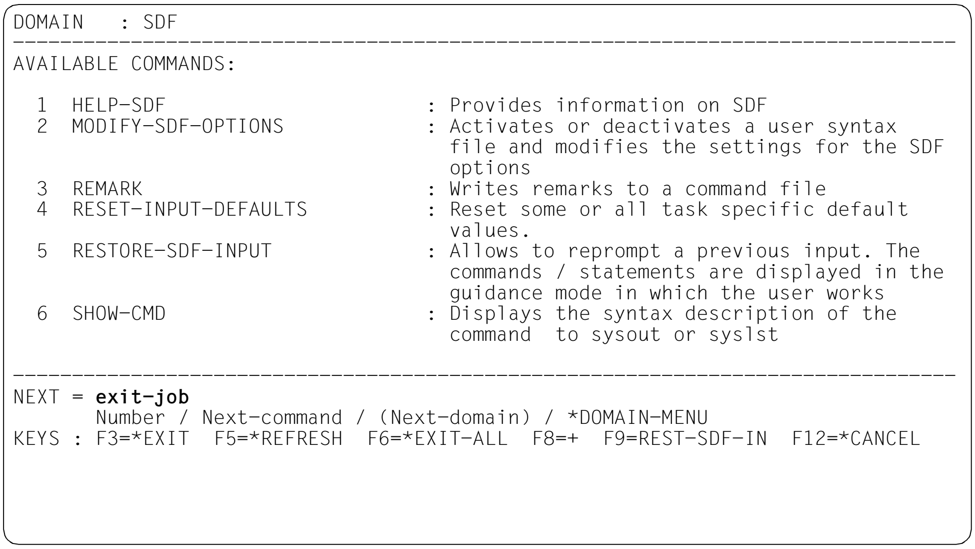
Figure 47: Operand form for the SHOW-SDF-OPTIONS command

The default value *ALL of the INFORMATION operand is overwritten with ”*user”.

When the screen is sent off with $\boxed{\text{DUE}}$, SDF outputs information on all SDF options that you yourself can change with MODIFY-SDF-OPTIONS:

```
%  USER      : *NONE
%CURRENT SDF OPTIONS :
%  GUIDANCE          : *MAXIMUM
%  LOGGING           : *INPUT-FORM
%  CONTINUATION      : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING      : *NO
%  MODE              : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS     : *STYLE-GUIDE-MODE
%  INPUT-HISTORY     : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
%PLEASE ACKNOWLEDGE
```

When the $\boxed{\text{DUE}}$ key is pressed, SDF again displays the command menu for the application domain SDF:



```
DOMAIN   : SDF
--------------------------------------------------------------------------------
AVAILABLE COMMANDS:

  1  HELP-SDF                       : Provides information on SDF
  2  MODIFY-SDF-OPTIONS             : Activates or deactivates a user syntax
                                      file and modifies the settings for the SDF
                                      options
  3  REMARK                         : Writes remarks to a command file
  4  RESET-INPUT-DEFAULTS           : Reset some or all task specific default
                                      values.
  5  RESTORE-SDF-INPUT              : Allows to reprompt a previous input. The
                                      commands / statements are displayed in the
                                      guidance mode in which the user works
  6  SHOW-CMD                       : Displays the syntax description of the
                                      command  to sysout or syslst

--------------------------------------------------------------------------------
NEXT = exit-job
       Number / Next-command / (Next-domain) / *DOMAIN-MENU
KEYS : F3=*EXIT  F5=*REFRESH  F6=*EXIT-ALL  F8=+  F9=REST-SDF-IN  F12=*CANCEL
```

Figure 48: Command menu for the application domain SDF

Instead of selecting a command from the menu by entering a number in the NEXT line, it is also possible to enter any “Next-command” desired. Sending off the menu causes this command to be executed. The EXIT-JOB command is entered in the NEXT line. Sending off the form terminates the session.

6 SDF syntax files

SDF is delivered together with syntax files containing:

- information on the syntax of commands and statements
- information on how these commands are implemented in BS2000, e.g.
  - the names of the system modules which handle command execution
  - the conventions for parameter transfer to the executive system modules
- information on privileges that the calling user must possess
- general and command-specific conventions for dialog guidance
- explanatory texts relating to the commands and their operands (help texts).

SDF extracts the information it requires for processing a command input from the activated syntax files.

Similarly, SDF supports the input of statements to any programs which are defined in a syntax file, e.g. to SDF-A or SDF-U.

The software product SDF-A (**S**ystem **D**ialog **F**acility-**A**dministration) and the utility routine SDF-U (**S**ystem **D**ialog **F**acility-**U**tility) can be used to process syntax files.
Syntax files can be merged by means of the utility routine SDF-I.

There are three types of syntax file:

- system syntax file
- group syntax file
- user syntax file.

It is possible to activate more than one syntax file for a given task: one basic system syntax file and several subsystem syntax files. In addition, one group and one user syntax file may be activated for a given task. For this eventuality, there is a fixed file hierarchy defining how SDF handles a number of syntax files.

6.1 System syntax file

SDF cannot be used unless a basic system syntax file is present. The basic system syntax file is activated automatically when the system is loaded. As of SDF V3.0, further system syntax files - the subsystem syntax files - may be present in addition to the basic system syntax file.

Only one basic system syntax file can be active in the entire system at any given time.

Entries on system syntax files are maintained in the SDF parameter file. The system syntax files are managed by the systems support staff by means of the MODIFY-SDF-PARAMETERS command. The SDF-PAR utility routine allows the systems support staff to create an SDF parameter file.

Basic system syntax file

The basic syntax file contains the definitions of the application areas, the standard statements and the SDF-specific commands (from the SDF application area) as well as the system-wide global information settings. The SYSSDF.SDF.045 syntax file is activated for this purpose by default. Systems support can make modifications with the SDF-A or SDF-U utility routines (e.g. to restrict functions). Modifications apply to all users of the system. Exchanging the basic system syntax file during the session takes immediate effect for all existing and future tasks.

Subsystem syntax file

There can be also several subsystem syntax files activated in addition to the basic syntax file. A subsystem syntax file contains the definitions of commands and programs that belong to a subsystem managed by DSSM or even to any installation unit (including BS2CP). Subsystem syntax files can be modified using SDF-A. Any such modifications, e.g. limitations on functional scope, apply for all users of the system.
Exchanging a subsystem syntax file during the session takes immediate effect for all existing and future tasks.

There are two ways to activate subsystem syntax files:

- Tthe name of the subsystem syntax file is defined in the subsystem declaration. In this case, the subsystem syntax file is automatically activated when the subsystem is loaded and deactivated when the subsystem is unloaded.
- The name of the subsystem syntax file is entered in the SDF parameter file. In this case, the subsystem syntax file is automatically activated at system initialization. If the pubset on which this file is stored is not available when the system is initialized, then it can only be activated after importing the pubset.
  It can be modified (deactivated or exchanged) via the SDF parameter file only. In this case, the subsystem syntax file is available independently of the availability of the corresponding subsystem.

If a command or statement definition is defined in more than one active subsystem syntax files, then the following cases arise:

- Only one version of the subsystem can be active at a time. In this case the command or statement definition is used that is found in the syntax file of the version of the subsystem last activated.
- Several versions of the subsystem can be active at a time (coexistence). The following cases arise:
  - The command or statement definition from the syntax file of the corresponding subsystem version is used for a task that is assigned to a specific subsystem version. The syntax analysis of a START-<utility> command is always performed before the program is loaded, i.e. before connecting the task to a subsystem version of the program called. This is why the syntax file of the first version of the subsystem activated is evaluated.
  - The command or statement definition contained in the first version of the subsystem activated is used for tasks that are not assigned to any subsystem version.

6.2 Group syntax file

A group syntax file may be active, but need not be. Systems support assigns it to the user ID. The assignment is made via the user entry. The *PROFILE-ID* output field of the SHOW-USER-ATTRIBUTES command contains the name of a profile ID. It is via this profile ID that a user ID or a group of user IDs can be assigned a group syntax file. The profile ID and group syntax file assignment is managed in the SDF parameter file. The respective entries are made by systems support staff by means of the MODIFY-SDF-PARAMETERS command.
When such an assignment has been made, the group syntax file is automatically activated after logon processing and remains active until the end of the task. Only one group syntax file may be active per task at any one time. Any change in the assignment is effective for future tasks only.

A group syntax file may contain extensions, limitations and other modifications relative to the system syntax file. It is thus possible for systems support staff to modify the range of functions available to specific users. If, in the system syntax file, they have imposed a system-global restriction on the full range of functions provided by Fujitsu Siemens Computers, they can lift this restriction for a particular user ID via a group syntax file. Conversely, in a group syntax file they can impose a restriction for a specific user ID on functions offered on a system-global basis.

Group syntax files must normally be generated by the user by means of the software product SDF-A; for certain software products they can be supplied by Fujitsu Siemens Computers.

If new or modified commands or statements are delivered with a new product version, then all group syntax files that contain this command or statement definition must be replaced or modified accordingly.

6.3 User syntax file

A user syntax file may be activated, but need not be. For a task started under the identification <userid>, the user syntax file cataloged under the name $<userid>.SDF.USER.SYNTAX is automatically activated following processing of the SET-LOGON-PARAMETERS command.

During a task the user may activate/deactivate one or more user syntax files with the MODIFY-SDF-OPTIONS command. A user syntax file may contain extensions, limitations and other modifications relative both to the system syntax files and, in certain cases, to the group syntax file. Possible extensions and other functional modifications are restricted to program statements and to commands which are implemented via procedures. Limitations on functional scope defined in a system or group syntax file for BS2000 commands (implemented via system modules) cannot be lifted in a user syntax file.

User syntax files must be generated by the user with the aid of the software product SDF-A, but can also be supplied with certain software products.

If new or modified commands or statements are delivered with a new product version, then all user syntax files that contain this command or statement definition must be replaced or modified accordingly.

6.4 Privileges in syntax files

As of BS2000/OSD-BC V1.0, definitions of privileges in syntax files are evaluated. Privileges can be associated with application domains, programs, commands, statements, operands and operand values. Users have access only to those objects of a syntax file whose privilege matches the one defined for their own user ID. Privileges in syntax files can thus be used to restrict the functional scope for particular users, making a restriction via a specific group syntax file superfluous.

Unless otherwise specified, all user IDs with the exception of the system IDs are assigned the privilege STD-PROCESSING. If the chargeable software product SECOS is present in the system, privileged functions can be made available to non-privileged user IDs by assigning them the required privilege.

The BS2000 concept of privileges is described in the “SECOS” manual [10]. The assignment of privileges in syntax files is described in the “SDF-A” manual [4].

6.5 File hierarchy

Which command and statement inputs are possible depends on the contents of the syntax files assigned. The activated syntax files are searched in the order user syntax file ⟶ group syntax file ⟶ system syntax file.

The definitions contained in the user syntax file have priority over those in the group and system syntax files. If a definition is contained in several user syntax files, the definition in the last activated user syntax file has priority.

The definitions of BS2000 commands (implemented via system modules) contained in the user syntax files must be accommodated in full by the group or system syntax file. If such a command is defined in all three files, the definition in the user syntax file must be accommodated in full by the group syntax file.

The definitions contained in the group syntax file have priority over those in the system syntax file. Definitions not contained in the group syntax file are taken from the system syntax file. If the file hierarchy was disabled during assignment of a group syntax file, the available command/statement set is based exclusively on the group syntax file.

If a definition is contained in several system or subsystem syntax files, the definition in the last activated subsystem syntax file has priority.

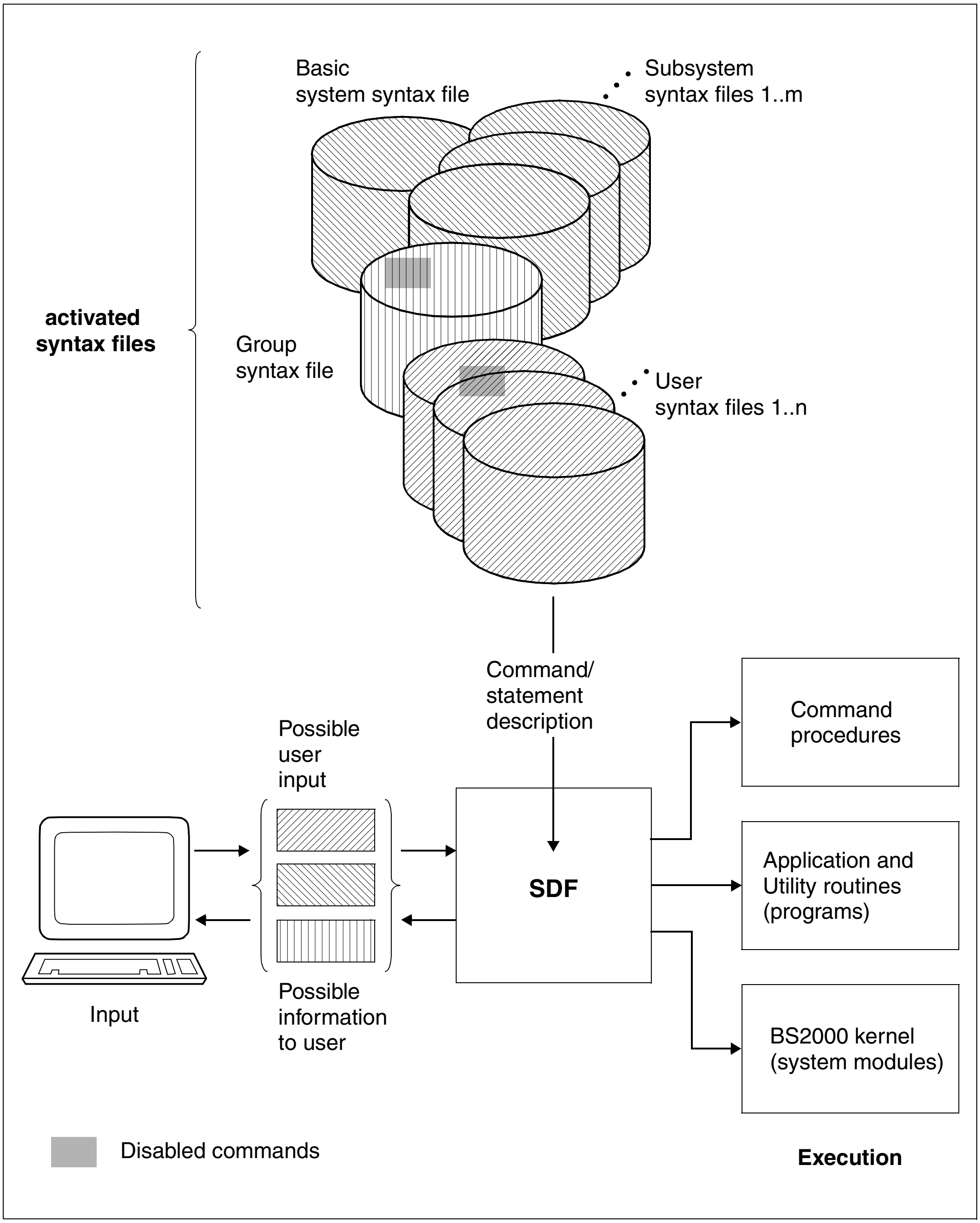


Figure 49: How SDF works when different syntax file types are activated

6.6 Processing syntax files

Syntax file processing (creation and/or modification) is possible only with the utility routines SDF-A, SDF-U or SDF-I, and on the logical level only. The internal structure of the file remains transparent to the user. Editing of syntax files by means other than those mentioned above will therefore lead to error messages when manipulated command/ statement definitions are addressed.

With the software product SDF-A it is possible to create new syntax files and to modify existing ones.

SDF-A offers the following capabilities:

- definition of user-specific commands implemented via command procedures, and modification of such definitions
- definition of user-specific programs and program statements, and modification of such definitions
- disabling of application domains, commands, programs, statements, operands and operand values
- assignment of privileges for application domains, commands, programs, statements, operands and operand values
- modification of default values
- renaming of application domains, of command, statement and operand names and of operand values which are keywords
- modification or replacement of help texts for guided dialog, e.g. replacing German help texts by texts in the appropriate local language
- modification of the assignment of commands to application domains and definition of new application domains
- modification of default values of SDF parameters, e.g. for guided dialog
- display of syntax file contents

The operations performed on a syntax file with SDF-A take effect as soon as the file is activated.

With the utility routine SDF-U, systems support can manage syntax files.
SDF-U provides a small subset of the functions of SDF-A:

- modification of default values of SDF options, e.g. for guided dialog
- modification of definitions of commands which are implemented via procedures
- display of syntax file contents
- copying of syntax file contents
- deletion of syntax file objects

The utility routine SDF-I can be used to obtain a system/group syntax file from a number of software unit syntax files. Note that only syntax files of the same type can be processed.

SDF-I offers the following functions:

- Output of information from a syntax file. The following items are shown:
  - type of the syntax file
  - format of the syntax file
  - presettings of SDF options (global information)
  - version (global information)
  - commands contained and product names of the software units

  The SDF-I statement SHOW-SYNTAX-FILE is described in the section SDF-I statements (see page 232).
- Converting a syntax file to the new format. As of SDF V2.0A, the internal logical structure of syntax files has been changed. SDF V2.0A supports both the new and the old format (the latter, however, with reduced performance). Syntax files that were created with an SDF-A version earlier than V2.0 can be converted from old to new format by means of the SDF-I statement CONVERT-SYNTAX-FILE (see page 232).
- Merging various software unit syntax files into one syntax file. This function serves to create a new syntax file or to add a software unit syntax file for a newly configured software product to an existing syntax file.
- Unmerging a syntax file. This function serves to incorporate the software unit syntax file for a new version of an already established software product into the existing syntax file. In this process, the constituents that were added through the old software unit syntax file are unmerged.

7 SDF user interface

This section lists all SDF-specific commands, standard statements and macro calls available to the user for task-specific SDF settings, user syntax file management, and utilization of SDF for user-written programs.

7.1 Commands of the SDF domain

All commands the user needs for controlling and managing the task-specific SDF interface are contained in the SDF application domain (see also Volumes 1-5 of the “Commands” manual [1]). The following are described below the table:

– HELP-SDF
– MODIFY-SDF-OPTIONS
– REMARK
– RESET-INPUT-DEFAULTS
– RESTORE-SDF-INPUT
– SHOW-CMD
– SHOW-INPUT-DEFAULTS
– SHOW-INPUT-HISTORY
– SHOW-RETURNCODE
– SHOW-SDF-OPTIONS
– SHOW-SYNTAX-VERSIONS
– WRITE-TEXT

Each command description contains the command name with a brief explanatory text, the application domain, the privileges with which the command can be executed, a functional description, syntax notation, description of the operands, command return codes, representation of outputs to SYSOUT or in S variables (optional for SHOW commands), examples and notes.

The commands for system-wide management and control of SDF are reserved for systems support staff and can be found in the manuals “Commands, volumes 1-5” [1] and “SDF Management” [5].

HELP-SDF
Provide information on SDF

Domain:	SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The HELP-SDF command displays information regarding SDF.

The information available is subdivided into sections that can be accessed via the individual operands of the command.

When the user enters HELP-SDF without operands or only with the preset operand values, a brief introduction to SDF is displayed (for a description see “HELP-SDF with preset operand values” on page 167).

Entering /HELP-SDF? displays the operand form for the command (temporarily guided dialog), and the information required can be selected.

Format

```
HELP-SDF                                                          Alias: HPSDF

 GUIDANCE-MODE = *NO / *YES
,SDF-COMMANDS = *NO / *YES
,ABBREVIATION-RULES = *NO / *YES
,GUIDED-DIALOG = *YES (...)
    *YES(...)
      |   SCREEN-STEPS = *NO / *YES
      |  ,SPECIAL-FUNCTIONS = *NO / *YES
      |  ,FUNCTION-KEYS = *NO / *YES
      |  ,NEXT-FIELD = *NO / *YES
,UNGUIDED-DIALOG = *YES (...) / *NO
    *YES(...)
      |   SPECIAL-FUNCTIONS = *NO / *YES
      |  ,FUNCTION-KEYS = *NO / *YES
```

Operands

GUIDANCE-MODE = *NO / *YES
Type of dialog guidance (output see page 168).

SDF-COMMANDS = *NO / *YES
SDF-specific commands (output see page 169).

ABBREVIATION-RULES = *NO / *YES
Abbreviation rules for the command syntax (output see page 170).

GUIDED-DIALOG = *YES(...) / *NO
Use of guided dialog (output see page 171).

GUIDED-DIALOG = *YES(...)
The description of guided dialog is subdivided into 4 blocks of information which can be accessed individually as follows:

> **SCREEN-STEPS = *NO / *YES**
> Sequence and contents of the menus (application domain menu, command menu, operand form, subform; output see page 171).
>
> **SPECIAL-FUNCTIONS = *NO / *YES**
> Special functions within the menu (output see page 171).

FUNCTION-KEYS = *NO / *YES
Function keys for menu control (output see page 172).

NEXT-FIELD = *NO / *YES
Input in the NEXT line for menu control (output see page 173).

UNGUIDED-DIALOG = *NO / *YES
Use of unguided dialog (output see page 173).

UNGUIDED-DIALOG = *YES(...)
The description of unguided dialog is subdivided into 2 blocks of information which can be accessed individually as follows:

SPECIAL-FUNCTIONS = *NO / *YES
Special functions within the unguided dialog menu (output see page 174).

FUNCTION-KEYS = *NO / *YES
Function keys for controlling the unguided dialog menu (output see page 175).

Return codes

(SC2)	SC1	Maincode	Meaning / Guaranteed messages
	0	CMD0001	Command executed
1	32	CMD0500	Syntax description in current syntax file invalid
1	64	CMD0810	Error during execution. Guaranteed messages: CMD0500, CMD0810

Information displayed with HELP-SDF:

HELP-SDF with preset operand values

```
%
% Introduction
%
% SDF is a convenient command interpreter and dialogue manager
% If the user desires, input can be entered as before.
% But the user can also utilize the advantages of SDF:
%  - Abbreviation mechanism
%  - Block input
%  - Guided dialogue
%  - Command bufferization (history)
%  - Definition of task-specific default values
%
% In addition, a more comprehensible command language has
% has been developed:
% The names of commands, operands and constant operand
% values were chosen so as to indicate clearly their function.
% Similar features (for example, the name of a file) are
% named accordingly similar naming rules (for example, FILE-NAME=
% or FROM-FILE= or TO-FILE=).
% Incorrect input is presented for correction along with
% a message.
% Commands which are part of the old commando language (ISP)
%PLEASE ACKNOWLEDGE
% can be input in the form "<old cmd>?". The new SDF-command
% is then displayed with its new syntax, i.e. a menu driven
% dialog allows to set the appropriate operand values.
% For further information, the user may enter "HELP-SDF?".
% The operand form for the command will then be displayed.
% Further information for an operand may then be obtained
% by setting that operand to "YES".
%
```

*HELP-SDF with GUIDANCE-MODE=*YES*

```
%
% Types of dialogue:
%
% 1. Guided dialogue
% Guided dialogue : supplies menus in the following order :
% Domain menu, Command menu, Operand form menu, sub-form menu.
% The extent of information offered in a menu can be controlled via
% the GUIDANCE operand of the MODIFY-SDF-OPTIONS command :
% - GUIDANCE=*MAXIMUM  all operand values with options (limit
%                     values etc.); help texts for commands
%                      and operands
% - GUIDANCE=*MEDIUM   all operand values;
%                     help texts only for commands
% - GUIDANCE=*MINIMUM  only default values for operands;
%                     no options; no help texts.
%
%
% 2. Unguided dialogue
% Unguided dialogue : no menus. There are 2 types of unguided
% dialogue, which can be controlled via the GUIDANCE operand
% of the MODIFY-SDF-OPTIONS cmd :
% - GUIDANCE=*NO:   the system requests command input with
%                 "%CMD:"; syntax error dialogue : correction
%                 of incorrect input without repetition
%                 of the entire command, detailed error
%                 messages; grouped command input :
%                 several commands, separated by logical
%                 end-of-line characters (LZE), can be issued
%                  at one time.
% - GUIDANCE=*EXPERT: the system requests command input with "/",
%                   no syntax error dialogue, detailed error
%                   messages; grouped command input.ch.
%
```

*HELP-SDF with SDF-COMMANDS=*YES*

```
%
% SDF-specific commands:
%
% - HELP-SDF
%     Displays information about SDF.
% - MODIFY-SDF-OPTIONS
%     Changes the task-specific SDF options.
% - REMARK
%     Inserts remarks into the command files.
% - RESET-INPUT-DEFAULTS
%     Deletes task-specific default values.
% - RESTORE-SDF-INPUT
%     Reprompts a previous input.
% - SHOW-CMD
%    Displays the syntax description of a command to SYSOUT/SYSLST.
% - SHOW-SDF-OPTIONS
%    Displays the task-specific information for the SDF options.
% - SHOW-INPUT-DEFAULTS
%     Outputs the task-specific default values.
% -  SHOW-INPUT-HISTORY
%     Displays the list of the latest inputs.
% - SHOW-SYNTAX-VERSIONS
%    Displays information about the contents of the active
%     syntax files.
% - WRITE-TEXT
%     Writes text to SYSOUT/SYSLST.
```

*HELP-SDF with ABBREVIATION-RULES=*YES*

```
%
% Abbreviation rules for the command syntax:
%
% -  Command name, operand name and constant operand values
%   (not basic data types) may be abbreviated as follows:
%   - Characters may be dropped from right to left
%     e.g.: TRANSFER-FILE abbreviated to TRANS
%   - Characters may be dropped from right to left
%     within the name elements.
%     e.g.: TRANSFER-FILE abbreviated to T-F
%          Restriction : Abbreviations must be unambiguous;
%          e.g.: CREATE as an abbreviation is ambiguous
%          between CREATE-FILE,CREATE-FILE-GENERATION,
%          CREATE-FILE-GROUP and CREATE-JV.
%
% - Operand values may also be specified without the
%   corresponding operand names (positional operands).
%   Here the following must be observed:
%   - A comma must be given for each omitted operand
%     preceding the positional operand.
%   - If, in the series of operands for a command,
%     an operand is specified completely (operand name and
%     operand value), subsequent operands may no longer be
%     specified as positional operands.
%
% - Guided dialogue: Operands for a structure need not be
%   in a sub-form, but may instead be entered in the operand
%   form. They must be enclosed in parentheses and must directly
%   follow the operand value introducing the structure.
%
% - Guided dialogue: In the NEXT line of the menu, complete
%   commands can also be entered. If a command name ends with
%   a question mark (<command>?), the operand form for the
%   specified command appears. It lists the operand values that
%   have already been specified (temporary guided dialogue).
%   If the specified command has no operands, the command help
%   text is displayed.
%
% - Ambiguity menus: entering a command name containing wildcards
%   and ending with a question mark causes SDF to display a menu
%   listing the commands which match the wildcard pattern.
%
```

*HELP-SDF with GUIDED-DIALOG=*YES(SCREEN-STEPS=*YES)*

```
%
% Guided dialogue: Order and content of the menus
%
% Domain menu    : Lists all domains (commands are assigned
%                 to particular domains according to their
%                 function). A domain can be selected by
%                 number or name. The domain *ALL-COMMANDS lists all
%                 the commands.
% Command menu   : Lists all commands of the selected domain.
%                 A command can be selected by number or name
%                 A command followed by ! has no operands.
% Operand form   : Lists the operands of the selected command.
%                 The operand values desired are to be
%                 specified.
% Sub-form       : Lists the operands of a structure
%                 If an operand value introducing a structure
%                 is specified in an operand form, a sub-form
%                 for the corresponding structure is
%                 displayed.
% Command menu   : If a command has been executed or aborted,
%                 the current command menu appears again.
%
```

*HELP-SDF with GUIDED-DIALOG=*YES(SPECIAL-FUNCTIONS=*YES)*

```
%
% ? as operand value:
%          Supplies help text and options (limit values, etc.)
%          for the operand; in addition, detailed error
%          messages in the case of incorrect input.
% ?? as operand value:
%          calls up a help text, a display of the range of
%          values for the relevant operand and an help text
%          giving information about the data types associated
%          with the operand.
% !  as operand value:
%          Enters the default value for the operand.
% (  following an operand value introducing a structure:
%          Displays the sub-form for the structure associated
%          with the operand value.
% () following an operand value introducing a structure:
%          Suppresses the sub-form and enters the default
%          values for the operands of the structure.
%  -   as the last character in an input line:
%          A continuation line is displayed.
% LZF key :
%         Deletes all characters in the input line, starting
%           at the cursor.
```

*HELP-SDF with GUIDED-DIALOG=*YES(FUNCTION-KEYS=*YES)*

```
%
% Guided dialogue: Function keys
%
% The effect of the function keys depends on the FUNCTION-KEYS
% option (MODIFY-SDF-OPTIONS command/statement).
% 2 modes are possible : the *OLD-MODE and the *STYLE-GUIDE-MODE
% which offers a richer functionality.
%
% *OLD-MODE
%
% K1  EXIT function
%     Equivalent to *EXIT in NEXT line.
%     Equivalent to F3 in *STYLE-GUIDE-MODE (see below).
% K2  INTERRUPT function.
%     Interrupts the program or procedure which is currently
%     running or interrupt of command output.
% K3  REFRESH function
%     Screen refresh function.
%     Equivalent to *REFRESH in NEXT line.
%     Equivalent to F5 in *STYLE-GUIDE-MODE.
% F1  EXIT-ALL function
%     Equivalent to *EXIT-ALL in NEXT line.
%     Equivalent to F6 in *STYLE-GUIDE-MODE (see below).
% F2  TEST function
%     Checks inputs for syntax errors. Equivalent to *TEST in the
%     NEXT line.
% F3  EXECUTE function
%     Executes the current operation. Equivalent to *EXECUTE in
%     the NEXT line.
%     Equivalent to F11 in *STYLE-GUIDE-MODE (see below).
%
% *STYLE-GUIDE-MODE
%
% K2  INTERRUPT function.
%     Interrupts the program or procedure which is currently
%     running or interrupt of command output.
% F1  HELP function.
%     Causes the display of a guided screen.
%     Equivalent to entering "?" in the NEXT line.
% F3  EXIT function.
%     Exits the current menu or form and switches to the
%     higher-ranking menu.
%     Equivalent to *EXIT in NEXT line.
%     Equivalent to K1 in *OLD-MODE.
% F5  REFRESH
%     Screen refresh function.
%     Equivalent to *REFRESH in NEXT line.
%     Equivalent to K3 in *OLD-MODE.
% F6  EXIT-ALL function
%     Exits the current menu or form and switches
%     to the highest-ranking menu.
%     Equivalent to *EXIT-ALL in NEXT line.
%     Equivalent to F1 in *OLD-MODE.
% F7  BACKWARD paging function
%     Pages backward in the menu or form.
%     Equivalent to "-" in NEXT line.
% F8  FORWARD paging function
%     Pages forward in the menu or form.
%     Equivalent to "+" in NEXT line.
% F9  REST-SDF-IN function.
%     Causes the last command / statement to be restored.
%     Equivalent to RESTORE-SDF-INPUT.
% F11 EXECUTE function
%     Executes the current operation. Equivalent to *EXECUTE in
%     the NEXT line.
%     Equivalent to F3 in *OLD-MODE.
% F12 Cancel function.
%     Exits the current menu or form and switches
%PLEASE ACKNOWLEDGE
%     to the higher-ranking menu or form.
%     Equivalent to *CANCEL in NEXT line.
%
```

*HELP-SDF with GUIDED-DIALOG=*YES(NEXT-FIELD=*YES)*

```
%
% Guided dialogue: NEXT line
%
% +,++,-,--  : Pages forward and back in the menu.
%              + and - equivalent to F8 and F7 respectively
%              in *STYLE-GUIDE-MODE.
% *EXECUTE   : Executes the current command. Equivalent
%              to the F3 key in *OLD-MODE and F11 in
%              *STYLE-GUIDE-MODE.
% *CONTINUE  : Combination of + or *EXECUTE depending of the
%              value given in the actual questionnaire.
% *TEST      : Checks input for syntax errors. Equivalent
%              to the F2 key in *OLD-MODE.
% *EXIT      : exits the current menu or form and switches
%              to the higher-ranking menu. Equivalent
%              to the K1 key in *OLD-MODE and F3 key in
%              *STYLE-GUIDE-MODE.
% *EXIT-ALL  : exits the current menu or form and switches
%              to the highest-ranking menu. Equivalent
%PLEASE ACKNOWLEDGE
%              to the F1 key in *OLD-MODE and F6 key in
%              *STYLE-GUIDE-MODE.
% *CANCEL    : exits the current menu or form and switches
%              to the higher-ranking menu or form. Equivalent
%              to the F12 key in *STYLE-GUIDE-MODE.
% *REFRESH   : Repeats the last menu displayed. Equivalent
%              to the K3 key in *OLD-MODE and F5 in
%              *STYLE-GUIDE-MODE.
% *DOM-MENU  : Switches to the domain menu.
% (<domain>) : Displays the command menu of the domain <domain>.
% <command>? : Executes the current operation and displays
%              the operand form for the command <command>.
%              Operand values that have already been specified
%              are entered in the form.
% <command>  : Executes the current operation and then,
%              in addition, the specified command <command>.
% !<command> : Executes the current operation and defines the
%             specified operand values as task-specific
%             default values of the command. When used in
%             combination with <command>? allows the default
%             values to be entered in the operand form.
% *DOWN(<operand>) : Displays the sub-form for the operand
%              value <operand>.
%PLEASE ACKNOWLEDGE
% *UP       : Switches from the sub-form back to the
%              operand form.
% ?         : Increases the guidance level.
% ??        : Further increases the guidance level.
%
```

*HELP-SDF with UNGUIDED-DIALOG=*YES(SPECIAL-FUNCTIONS=*YES)*

```
%
% Unguided dialogue: Special functions
%
%  ?  as operand value :
%      Supplies help text and options
%      (limit values, etc.)
% ?  as command :
%      Displays the domain menu.
%      (temporary guided dialogue)
% ?? as operand value :
%      calls up a help text, a display of the range of
% values for the relevant operand and an help text
% giving information about the data types associated
% with the operand.
% <command>? as command :
% Displays the operand form for the command
% <command> (temporary guided dialogue).
% Operand values that have already been
% specified are filled in.
% If there are no operands, the help text of the
% command is displayed.
% !<command> as command :
% Defines the specified operand values as task-specific
% default values of the command. When used in
% combination with <command>? allows the default
% values to be entered in temporarily unguided dialog.
% ^ or *SECRET as operand value (only for passwords):
% Causes the input field for the password to
% be kept blank.
% LZF key :
% Deletes all characters in the input line, starting
% at the cursor.
% LZE key : enables blocked input. Several commands can be
% given separated by LZEs and executed at the same time
% by a terminating DUE
%
```

*HELP-SDF with UNGUIDED-DIALOG=*YES(FUNCTION-KEY=*YES)*

```
%
% Unguided dialogue: Function keys
%
% The effect of the function keys depends on the FUNCTION-KEYS
% option (MODIFY-SDF-OPTIONS command/statement).
% 2 modes are possible : the *OLD-MODE and the *STYLE-GUIDE-MODE
% which offers a richer functionality.
%
% *OLD-MODE
%
% K1  EXIT function.
%     Terminates the program which is currently running.
%     A message is issued asking the user whether or not
%     he wants to terminate.
%     Equivalent to F3 in *STYLE-GUIDE-MODE.
% K2  INTERRUPT function.
%     Interrupts the program or procedure which is currently
%     running or interrupts the command output.
% F1  EXIT-ALL function.
%     Terminates the program which is currently running.
%    A message is issued asking the user whether or not
%    he wants to terminate.
%    Equivalent to F6 in *STYLE-GUIDE-MODE.
%
% *STYLE-GUIDE-MODE
%
% K2  INTERRUPT function.
%    Interrupts the program or procedure which is currently
%    running or interrupt of command output.
% F1  HELP function.
%    Causes the display of a guided screen.
%    Equivalent to entering "?".
%
%
% F3  EXIT function.
%    Terminates the program which is currently running.
%    A message is issued asking the user whether or not
%    he wants to terminate.
%    Equivalent to K1 in *OLD-MODE.
% F6  EXIT-ALL function.
%    Terminates the program which is currently running.
%    A message is issued asking the user whether or not
%    he wants to terminate.
%    Equivalent to F1 in *OLD-MODE.
% F9  RESTORE-SDF-INPUT function.
%    Causes the last command / statement to be restored.
%    Equivalent to RESTORE-SDF-INPUT.
% F12 CANCEL function.
%    Terminates the program which is currently running.
%    A message is issued asking the user whether or not
%    he wants to terminate.
%
```

MODIFY-SDF-OPTIONS
Activate user syntax file and modify SDF options

Domain:	SDF
Privileges:	STD-PROCESSING TSOS HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

With the MODIFY-SDF-OPTIONS command, the user can

- activate or deactivate user syntax files (user syntax files are generated using the SDF-A utility routine; see the “SDF-A” manual [4])

  Note

  As of version V2.0A, SDF uses a *modified* syntax file format. For performance reasons, it is advisable to use new-format syntax files only. User syntax files are converted to the new format either by processing with SDF-A (as of V2.0A) or by means of the CONVERT-SYNTAX-FILE statement of the SDF-I utility routine.
  A new-format syntax file cannot be converted back to the old format.

- change the settings of SDF options for specific tasks:
  – type of dialog guidance (GUIDANCE operand)
  – extent of logging (LOGGING)
  – input interface for utilities (UTILITY-INTERFACE operand)
  – syntax error dialog in procedures (PROCEDURE-DIALOGUE operand)
  – position of the continuation character (CONTINUATION operand)
  – syntax checking of commands (MODE operand)
  – syntax checking of statements (DEFAULT-PROGRAM-NAME operand)
  – selection of function key assignment (FUNCTION-KEYS operand)
  – storage of input (INPUT-HISTORY operand)

The settings of SDF options are taken from the global information of the activated system or group syntax file at the beginning of the task. If the user activates a user syntax file, the settings of SDF options are modified according to the global information contained in that file. When the OMNIS utility routine is used, however, the GUIDANCE=*EXPERT mode is set at the beginning of the task and if the user syntax file is switched.

In (temporarily) guided dialog the operand form of the MODIFY-SDF-OPTIONS command contains as preset values the current settings, which can also be queried using the SHOW-SDF-OPTIONS command. The *UNCHANGED operand states that the current setting is not to be modified.

In programs with an SDF interface, MODIFY-SDF-OPTIONS is available as a standard statement with the same syntax and functionality but without the operands MODE, DEFAULT-PROGRAM-NAME and COMMAND-STATISTICS.

Privileged functions

Systems support (privilege TSOS) can use the CMD-STATISTICS operand to activate and deactivate the statistical recording of all commands issued in the system. All commands contained in the system or subsystem syntax files are taken into account.

The statistics are recorded in the SYS.SDF.CMD.STATISTICS file. If the statistics routine is active (CMD-STATISTICS=*YES), SDF counts every command issued by running a counter for each command and each alias.
If the statistics routine is deactivated (CMD-STATISTICS=*NO), SDF evaluates the count and writes the edited statistics to the above-named file. Any existing statistics file is overwritten. SDF then resets the command counter to zero.

The edited statistics file contains:

– a header showing the date and time when the statistics routine was activated and deactivated.
– for each subsystem a title containing the name of the subsystem
– for each command of a subsystem the command name and the count.

Format

```
MODIFY-SDF-OPTIONS                                                    Alias: MDSDFO

SYNTAX-FILE = *UNCHANGED / *ADD(...) / *REMOVE(...) / *NONE
   *ADD(...)
     | ADD-NAME = *STD / list-poss(2000): *STD / <filename 1..54>
   *REMOVE(...)
     | REMOVE-NAME = *LAST / *ALL / *BY-SELECTION / list-poss(2000): <filename 1..54> / *STD
,GUIDANCE = *UNCHANGED / *EXPERT / *NO / *MAXIMUM / *MEDIUM / *MINIMUM
,LOGGING = *UNCHANGED / *INPUT-FORM / *ACCEPTED-FORM / *INVARIANT-FORM
,UTILITY-INTERFACE = *UNCHANGED / *OLD-MODE / *NEW-MODE
,PROCEDURE-DIALOGUE = *UNCHANGED / *YES / *NO
,CONTINUATION = *UNCHANGED / *OLD-MODE / *NEW-MODE
,MENU-LOGGING = *UNCHANGED / *NO / *YES
,CMD-STATISTICS = *UNCHANGED / *NO / *YES
,MODE = *UNCHANGED / *EXECUTION / *TEST(...)
   *TEST(...)
     | CHECK-PRIVILEGES = *UNCHANGED / *NO / *YES
,DEFAULT-PROGRAM-NAME = *UNCHANGED / *NONE / <structured-name 1..30>
,FUNCTION-KEYS = *UNCHANGED / *STYLE-GUIDE-MODE / *BY-TERMINAL-TYPE / *OLD-MODE
,INPUT-HISTORY = *UNCHANGED / *ON(...) / *OFF / *RESET
   *ON(...)
     | NUMBER-OF-INPUTS = *UNCHANGED / <integer 1..100>
     | ,PASSWORD-PROTECTION = *UNCHANGED / *YES / *NO
```

Operands

SYNTAX-FILE = *UNCHANGED / *ADD(...) / *REMOVE(...)
Indicates whether a user syntax file is to be activated or deactivated. Two or more user syntax files can be activated at the same time.
A user syntax file to be activated must be accessible and shareable (if the job submitter is not the owner of the syntax file). If the syntax file is protected by basic ACL or GUARDS, the job submitter must have at least execute authorization (see operands USER-ACCESS, BASIC-ACL and GUARDS in the CREATE-FILE and MODIFY-FILE-ATTRIBUTES commands respectively).

If the file is protected by an execute password, the password must be contained in the password table for the job (see ADD-PASSWORD command).

SYNTAX-FILE = *UNCHANGED
The previous declaration still applies.

SYNTAX-FILE = *ADD(...)
One or more additional user syntax files are to be activated.

ADD-NAME = list-poss(2000): *STD / <filename 1..54>
Specifies the user syntax file to be activated.
A list of two or more syntax files can also be specified.

ADD-NAME = *STD
The user syntax file with the standard file name **SDF.USER.SYNTAX** is activated if it exists in the user ID.

ADD-NAME = <filename 1..54>
The specified user syntax file is activated.

SYNTAX-FILE = *REMOVE(...)
One or more user syntax files are to be deactivated.

REMOVE-NAME = *LAST / *ALL / list-poss(2000): <filename 1..54> / *STD
Specifies the user syntax file to be deactivated.
A list of several syntax files can also be specified.

REMOVE-NAME = *LAST
The last activated user syntax file is deactivated.

REMOVE-NAME = *ALL
All activated user syntax files are deactivated.

REMOVE-NAME = *BY-SELECTION
All activated user syntax files are displayed in a selection mask. Those which are to be deactivated can be marked in a mark column.

REMOVE-NAME = <filename 1..54>
The specified user syntax file is deactivated.

REMOVE-NAME = *STD
The user syntax file with the standard file name **SDF.USER.SYNTAX** is deactivated.

SYNTAX-FILE = *NONE
No user syntax file has been activated for the job.
Currently active user syntax files, if any, are deactivated.

GUIDANCE =
Specifies the type of dialog guidance.

GUIDANCE = *UNCHANGED
The previous declaration still applies.

GUIDANCE = *EXPERT
The EXPERT form of unguided dialog is set:
The system requests command input with "/" and statement input with "//". No syntax error dialog is carried out. Blocked input of commands/statements is possible, i.e. several commands/statements, separated by logical end-of-line characters, can be issued at the same time.

GUIDANCE = *NO
The NO form of unguided dialog is set:
Depending on the language set (English (E) or German (D)), the system requests command input with "%CMD:" or "%KDO:" and statement input with "%STMT:" or "%ANW:".
A syntax error dialog allows input errors to be corrected without having to repeat the entire command/ statement. Blocked input of commands/statements is possible, i.e. several commands/statements, separated by logical-end-of-line characters, can be issued at the same time.

GUIDANCE = *MAXIMUM
Guided dialog, maximum help level, is set:
All alternative operand values with suffixes, and all help texts for the displayed application domains, commands, statements and operands are displayed.

GUIDANCE = *MEDIUM
Guided dialog, medium help level, is set:
All alternative operand values without suffixes, and all help texts for the displayed application domains, commands and statements are displayed.

GUIDANCE = *MINIMUM
Guided dialog, minimum help level, is set:
Only default values for operands without suffixes are displayed.

LOGGING =
Specifies how inputs are to be logged. Input for operands defined as "secret" is never listed in a log.

LOGGING = *UNCHANGED
The previous declaration still applies.

LOGGING = *INPUT-FORM
In unguided dialog, input strings are logged exactly as entered. In guided dialog or following an error dialog, logging is performed as with *ACCEPTED-FORM.

LOGGING = *ACCEPTED-FORM
The following is logged:
- all names in their extended form
- all specified operands, each with its name and the specified value
- the final representation resulting from a correction dialog, if any.

Inputs of the guided dialog are concatenated to form a single string.

LOGGING = *INVARIANT-FORM
The following is logged:
- all names with the default names defined in the syntax file
- all specified operands, each with its name and the specified value
- all optional operands implicitly contained in the input, each with its name and its default value
- the final representation resulting from a correction dialog, if any.

Inputs of the guided dialog are concatenated to form a single string.

UTILITY-INTERFACE = <u>*UNCHANGED</u> / *OLD-MODE / *NEW-MODE
Controls the input interface of utility routines which offer an old and a new (SDF) interface in parallel.

PROCEDURE-DIALOGUE = <u>*UNCHANGED</u> / *YES / *NO
Specifies whether the user is to be requested to correct errored procedure commands during the execution of a procedure (syntax error dialog in procedures). At the same time the operand controls the help request ("?" in the input) within procedures.

CONTINUATION = <u>*UNCHANGED</u> / *OLD-MODE / *NEW-MODE
Determines the column in which the command continuation character is to be specified in procedure and ENTER files. If *OLD-MODE is set, the continuation character must be in column 72 precisely. If *NEW-MODE is set, it may be entered in any column from column 2 through 72.

MENU-LOGGING = <u>*UNCHANGED</u> / *NO / *YES
Specifies whether menus of the guided dialog are to be logged in their entirety. This operand is restricted to diagnostic purposes.

CMD-STATISTICS = <u>*UNCHANGED</u> / *NO / *YES
Only systems support can specify this operand (privilege TSOS):
Determines whether statistics on the issued commands are to be created from the system syntax file.

MODE = *UNCHANGED / *EXECUTION / *TEST(...)
Specifies whether test mode is activated or deactivated.

MODE = *EXECUTION
Test mode is deactivated.

MODE = *TEST(...)
Test mode is activated. Commands issued after the MODIFY-SDF-OPTIONS command are subjected to a syntax check but not executed. The MODIFY-SDF-OPTIONS and SHOW-SDF-OPTIONS commands are *always* executed.
The lower-ranking operand CHECK-PRIVILEGES determines whether the user has the privileges necessary for the input.
The treatment of statements in test mode can be defined in the DEFAULT-PROGRAM-NAME operand.
In *S procedures*, SDF-P control flow commands are executed also. This may cause errors, since commands declaring or setting S variables are not executed in test mode. The chargeable subsystem SDF-P offers a special debugger for S procedures (see the section dealing with debugging aids in the “SDF-P” manual [6]).

CHECK-PRIVILEGES = *UNCHANGED / *NO / *YES
Specifies whether the user’s privileges are checked in addition to the syntax check.

CHECK-PRIVILEGES = *NO
The user’s privileges are not checked. This setting may be necessary, for example, if the procedure to be tested is created for other users.

CHECK-PRIVILEGES = *YES
As well as the syntax check, SDF checks whether the user has the necessary privileges for the input.

DEFAULT-PROGRAM-NAME = *UNCHANGED / *NONE / <structured-name 1..30>
Determines whether in test mode (see the MODE operand) SDF program statements (beginning with //) are to undergo a syntax check. This check is performed on the basis of the syntax of the statements defined in the syntax file for the specified program name. The check is independent of the program call, since LOAD-/START-EXECUTABLE-PROGRAM (and LOAD-/START-PROGRAM) are not executed in test mode.

FUNCTION-KEYS = *UNCHANGED / *STYLE-GUIDE-MODE / *BY-TERMINAL-TYPE / *OLD-MODE
Determines the assignment of the function keys.

FUNCTION-KEYS = *STYLE-GUIDE-MODE
The function keys are assigned according to the SNI Styleguide.

K2	Interrupt function
F1	Help function
F3	Exit function
F5	Refresh function (only in guided dialogs)
F6	Exit All function
F7	Page back (only in guided dialogs)
F8	Page forward (only in guided dialogs)
F9	Execute RESTORE-SDF-INPUT INPUT=*LAST
F11	Execute function (only in guided dialogs)
F12	Cancel function

FUNCTION-KEYS = *BY-TERMINAL-TYPE
The function key assignment depends on the terminal type. If the terminal type supports the wider-ranging functionality of the Fujitsu Siemens Styleguide, SDF chooses the setting *STYLE-GUIDE-MODE. Otherwise, SDF chooses the setting *OLD-MODE.

Note

To choose a setting, SDF analyzes the type with which the terminal was generated in the system. If the generated terminal type is different from the actual terminal type, there is no guarantee that the setting will match the functional scope actually supported. In the case of a terminal emulation, the recognized terminal type can depend on both the generation and the value of an environment variable. For more detailed information, see the description of the emulation program used.

FUNCTION-KEYS = *OLD-MODE
The function keys are assigned in Old mode, which is supported by all terminal types.

K1	Exit function
K2	Interrupt function
K3	Refresh function (only in guided dialogs)
F1	Exit All function
F2	Test function (only in guided dialogs)
F3	Execute function (only in guided dialogs)

INPUT-HISTORY = *UNCHANGED / *ON(...) / *OFF / *RESET
Determines whether the input buffer is activated, deactivated or reset.

INPUT-HISTORY = *ON(...)
The input buffer is activated. SDF saves all syntactically correct inputs (commands and statements) to the input buffer. The commands or standard statements RESTORE-SDF-INPUT and SHOW-INPUT-HISTORY are not saved. Whether or not ISP commands are saved depends on what is specified in the PASSWORD-PROTECTION operand.
The SHOW-INPUT-HISTORY command (or standard statement) enables the user to display the saved inputs, while the RESTORE-SDF-INPUT command can be used to redisplay a specific input for entering again, either as it is or in modified form.

Note

Values specified for “secret” operands which match neither the default value nor a value defined via SECRET=*NO are stored in the input buffer with “^” or in plaintext, depending on the PASSWORD-PROTECTION operand.
Values specified for operands which are not “secret” are stored in the input buffer in plain text. In individual cases these inputs can contain sensitive information (e.g. procedure parameters).

The following steps will prevent such inputs from being displayed again via SHOW-INPUT-HISTORY or RESTORE-SDF-INPUT:
Before entering sensitive input, the input buffer must be deactivated and then activated again. If the inputs have already been saved, the input buffer can be reset with *RESET, but it must be remembered that this will delete all saved inputs.

NUMBER-OF-INPUTS = *UNCHANGED / <integer 1..100>
Maximum number of inputs to be saved.

PASSWORD-PROTECTION = *UNCHANGED / *YES / *NO
This operand is not permitted in procedure mode.
Defines whether values specified for “secret” operands and ISP commands are stored in the input buffer.

PASSWORD-PROTECTION = *YES
Values specified for “secret” operands are saved with “^” (equivalent to the operand value *SECRET). ISP commands are not stored in the input buffer.

PASSWORD-PROTECTION = *NO
Values specified for “secret” operands are stored in plaintext. ISP commands are also stored in the input buffer.

Note

Under this setting, passwords are displayed on the screen in plaintext with the SHOW-INPUT-HISTORY or RESTORE-SDF-INPUT command, and so may be seen by unauthorized users. You should therefore ensure that whenever you leave your terminal no unauthorized users can output the contents of the input buffer. If your terminal does not possess any appropriate security mechanisms (e.g. chipcard terminal), you should at least delete the input buffer before you leave.

INPUT-HISTORY = *OFF
The input buffer is deactivated. Subsequent inputs are not saved, but all previously saved inputs can still be accessed.

INPUT-HISTORY = *RESET
The input buffer is reset. The stored inputs are deleted and can no longer be accessed, while subsequent inputs are saved.

Return codes

(SC2)	SC1	Maincode	Meaning / Guaranteed messages
	0	CMD0001	Command terminated without errors
1	32	CMD0500	Syntax description in current syntax file invalid. Guaranteed message: CMD0500
	64	CMD0554	Command execution not successful. Guaranteed messages: CMD0300, CMD0302, CMD0490, CMD0508, CMD0509, CMD0552, CMD0554, CMD0555, CMD0579

Examples

```
/show-sdf-opt          ———————————————————————————————————————————————————— (1)
%SYNTAX FILES CURRENTLY ACTIVATED :
%  SYSTEM    : :2OSH:$TSOS.SYSSDF.SDF.045
%              VERSION : SESD04.5A300
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.ACS.140
%              VERSION : SESD14.0B100
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.AIDSYSA.140
%              VERSION : SESD14.0A600
.
.
.
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.SPACEOPT.020
%              VERSION : SESD02.0A300
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.SDF-A.041
%              VERSION : SESD04.1E10
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.TASKDATE.140
%              VERSION : SESD14.0A100
%  GROUP     : *NONE
%  USER      : *NONE
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES

/mod-sdf-opt syntax=*add(syssdf.user.special.01) —————————————————————— (2)
```

```
/show-sdf-opt inf=*user     ——————————————————————————————————————————  (3)
%  USER      : :2OSG:$USER1.SYSSDF.USER.SPECIAL.01
%              VERSION : UNDEFINED
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES

/mod-sdf-opt guidance=*max     ————————————————————————————————————  (4)
 .
 .
 .            (guided dialog, terminated with MOD-SDF-OPT GUIDANCE=*EXPERT)
 .
 .
/mod-sdf-opt syntax=*none     —————————————————————————————————————  (5)

/show-sdf-opt inf=*user     ———————————————————————————————————————  (6)
%  USER      : *NONE
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
```

(1) The SHOW-SDF-OPTIONS command provides information on all activated syntax files and the SDF options set. The output of the subsystem syntax files can be very extensive, depending on the number of activated subsystems. This example shows only an excerpt.

(2) The user syntax file *SYSSDF.USER.SPECIAL01* is activated by means of the MODIFY-SDF-OPTIONS command.

(3) The SHOW-SDF-OPTIONS command with INFORMATION=*USER provides information on activated user syntax file(s) and the SDF options set.

(4) Guided dialog with the maximum help level is selected.

(5) The user syntax file is deactivated.

(6) The information on the user syntax file and the SDF options set is output again.

REMARK
Insert remarks in command file

Domain:	JOB PROCEDURE
Privileges:	STD-PROCESSING OPERATING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION
Routing code:	@

Function

The REMARK command allows remarks to be inserted in command files (ENTER or procedure files) in order to document the job or procedure run. The command is logged as entered; in interactive operation it is only meaningful if job execution is logged on SYSLST.

The command can be issued from all operator terminals and authorized user programs. It is also permitted for operator command files.

In programs with an SDF interface, REMARK is available as a standard statement with the same syntax and functionality.

Format

REMARK
TEXT = <cmd-rest 0..1800>

Operands

TEXT = <command-rest 0..1800>
Text of the remarks.The text can be up to 1800 characters in length.
An equals sign must not be the first significant character, as otherwise the command will be interpreted as a value assignment (SET-VARIABLE without command name). A semicolon outside parentheses is interpreted as a command separator, i.e. any subsequent characters are interpreted as the next command.
When the text is entered at the console, its length is limited to one screen line.

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without error

Example

A procedure file begins with the following commands:

```
/BEG-PROC LOG=*CMD
/REMARK PROCEDURE FOR LINKING
...
...
```

In the BEGIN-PROCEDURE command, the specification LOGGING=*CMD enables output of the remark to SYSOUT. Then, when the procedure is executed, the remark is logged as follows:

```
% /REMARK PROCEDURE FOR LINKING
```

RESET-INPUT-DEFAULTS Delete task-specific default values

Domain:	SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The RESET-INPUT-DEFAULTS command allows the user to delete task-specific default values. Within the task, default values can be defined for commands and statements. The user can either delete all default values or restrict deletion to default values of commands or of statements. If only defaults of commands/statements are to be deleted, deletion can be restricted to specific commands/statements.

The RESET-INPUT-DEFAULTS command can be applied to a task-specific default value with a particular input serial number. To facilitate this, the output of the SHOW-INPUT-DEFAULTS command can be requested with input serial numbers (operand INPUT-SERIAL-NUMBER=*YES).

In programs with an SDF interface, RESET-INPUT-DEFAULTS is available as a standard statement with the same functionality.

Format

```
RESET-INPUT-DEFAULTS                                              Alias: RSID

OBJECT = *CMD (...) / *STMT(...) / *ALL / <integer 1..9999>

  *CMD(...)
     |  CMD = *ALL / <structured-name 1..30 with-wild(50)>

  *STMT(...)
     |  STMT = *ALL / <structured-name 1..30 with-wild(50)>
     |
     | ,PROGRAM = *CURRENT / *ALL / <structured-name 1..30>
```

Operands

OBJECT = *CMD(...) / *ALL / *STMT / <integer 1..9999>
Specifies the type of input for which the task-specific default values are to be deleted.

OBJECT = *CMD(...)
Only the task-specific default values of commands are deleted. The defaults of all or only of selected commands can be deleted.

CMD = *ALL / <structured-name 1..30 with-wild(50)>
Specifies whether the task-specific default values of all commands or only of selected commands are to be deleted.

CMD = *ALL
All task-specific default values of commands are deleted.

CMD = <structured-name 1..30 with-wild(50)>
Name of the command whose task-specific default values are to be deleted. If wildcards are used, the defaults of all commands which match the search pattern entered will be deleted.

OBJECT = *STMT(...)
Only the task-specific default values of statements are deleted. The defaults of all or only of selected statements of a program can be deleted.

STMT = *ALL / <structured-name 1..30 with-wild(50)>
Specifies whether the task-specific default values of all statements or only of certain statements are to be deleted. In the PROGRAM operand, the user can specify whether the deletion is to apply to defaults of statements of a specific program or of all programs.

STMT= *ALL
All task-specific default values of statements are deleted.

STMT = <structured-name 1..30 with-wild(50)>
Name of the statement whose task-specific default values are to be deleted. If wildcards are used, the defaults of all statements which match the search pattern entered will be deleted.

PROGRAM = *CURRENT / *ALL / <structured-name 1..30>
Specifies the program for whose statements specified in the STMT operand the task-specific default values are to be deleted.

PROGRAM = *CURRENT
Only default values of statements of the program currently defined in the SDF options are deleted. The program name can be set using the MODIFY-SDF-OPTIONS command (DEFAULT-PROGRAM-NAME operand).

PROGRAM = *ALL
The default values of all statements are deleted, regardless of the program name.

PROGRAM = <structured-name 1..30>
Program name, defined in a currently assigned syntax file.
Only default values of statements of the specified program are deleted.

OBJECT = *ALL
All task-specific default values, i.e. from both commands and statements, are deleted.

OBJECT = <integer 1..9999>
Input serial number of the default value to be deleted.
The input serial number of a default value can be taken from the output of the SHOW-INPUT-DEFAULTS command (operand INPUT-SERIAL-NUMBER=*YES).

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed without errors
1	0	CMD0001	No task-specific default value that matches the specifications exists. Guaranteed message: CMD0561
1	32	CMD0500	Syntax description in current syntax file invalid. Guaranteed message: CMD0500
1	64	CMD0561	Command execution not successful.. Guaranteed message: CMD0561

Example

```
/!print-doc line-spacing=*by-ebcdic ———————————————————————————————————— (1)
/!cre-file sup=*priv(vol=work01,dev-type=d3490-30) ————————————————————— (2)
/!cre-file sup=*public ————————————————————————————————————————————————— (3)
/!cre-file basic-acl=*all-rx ——————————————————————————————————————————— (4)

/show-input-defaults input-serial-number=*yes —————————————————————————— (5)
/"   1 :" !PRINT-DOC LINE-SPACING=*BY-EBCDIC-CONTROL
/"   2 :" !CRE-FILE SUPPORT=*PRIVATE-DISK(VOLUME=WORK01,DEVICE-TYPE=D3490-30)
/"   3 :" !CRE-FILE SUPPORT=*PUBLIC-DISK
/"   4 :" !CRE-FILE BASIC-ACL=*ALL-RX
/reset-input-defaults 2 ———————————————————————————————————————————————— (6)
/reset-input-defaults *cmd(cmd=create-file) ———————————————————————————— (7)

/show-input-defaults ——————————————————————————————————————————————————— (8)
/!PRINT-DOC LINE-SPACING=*BY-EBCDIC-CONTROL
```

(1) Evaluation of EBCDIC print control characters is set as the default value for the PRINT-DOCUMENT command. The LINE-SPACING operand is specified without a structure-initiator because in the syntax file *TEXT-FORMAT the default value of the operand is DOCUMENT-FORMAT.

(2) The private disk *WORK01* of device type *D3490-30* is set as the default value for the CREATE-FILE command. In this case, the structure-initiator must be specified, because the VSN and the device type define a private disk. To create a file on a public disk, SUPPORT=*PUBLIC must now be specified.

(3) Public disks are again set as the default for the CREATE-FILE command; but to create a file on the private disk *WORK01*, it is sufficient to specify SUPPORT=*PRIVAT.

(4) For the CREATE-FILE command, the value set by default is a BASIC-ACL, which grants all access rights only to the owner, but allows the other users to read and execute the file. The protection attribute is set accordingly if the higher-ranking structure is activated with PROTECTION=*PARAMETERS.

(5) All task-specific default values are output with their input serial numbers.

(6) The definition with the input serial number 2 is deleted. In this case this is the definition with CREATE-FILE for the private disk.

(7) All definitions for the CREATE-FILE command are deleted.

(8) The output of the task-specific default values now shows only the definition for the PRINT-DOCUMENT command.

RESTORE-SDF-INPUT
Restore previous input

Domain:	SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The RESTORE-SDF-INPUT command redisplays an input which has already been made and stored in the input buffer. The user can then use the displayed command or statement again as it is or in modified form without having to retype it. For an input to be reused, however, at least one character in the input string must be changed (e.g. you can overwrite a character with the same character).

The INPUT operand is used to select the input to be displayed. The default value *LAST-CMD recalls the last command input to be saved. Earlier inputs can be selected using the relative or absolute input serial number. The SHOW-INPUT-HISTORY command (or standard statement) provides information on all saved (and therefore available) inputs.

The output produced by RESTORE-SDF-INPUT depends on the current guidance mode (see the SHOW-SDF-OPTIONS command, output field *GUIDANCE*):

– In guided dialog a temporarily guided dialog is initiated for the command or statement that is to be output. The operand form contains all user inputs.
  A guided dialog is not possible for commands or statements without operands. Therefore only the help text and error message CMD0070 are displayed.
  If you select an AID command, the warning CMD0559 is output because AID commands are not supported in guided dialog.

– In unguided dialog the saved input string is displayed. To use dialog guidance for making modifications, the user can initiate a temporarily guided dialog by entering a question mark directly after the command/statement name.

The input buffer is controlled (activated/deactivated and deleted) via the MODIFY-SDF-OPTIONS command (or standard statement). Inputs in guided dialog are saved in ACCEPTED form, while inputs in unguided dialog are saved in INPUT form.
The RESTORE-SDF-INPUT command or statement is not saved.

Values specified for “secret” operands which match neither the default value nor a value defined via SECRET=*NO are saved in the input buffer with “^”.
In unguided dialog when these values are displayed again via RESTORE-SDF-INPUT, the user can do one of the following:

– send off the command/statement unchanged. In this case, SDF displays a blanked input field for each secret operand for the user to enter the desired value.
– delete the “^” and insert the desired value directly before sending off the command/statement.

In programs with an SDF interface, RESTORE-SDF-INPUT is available as a standard statement with approximately the same syntax and functionality.

Format

RESTORE-**SDF-IN**PUT	Alias: **RRSDFI**
INPUT = ***LAST-CMD** / <integer -100..-1> / <integer 1..9999>	

Operands

INPUT = *LAST-CMD / <integer -100..-1> / <integer 1..9999>
Determines which input from the input buffer is to be redisplayed.

INPUT = *LAST-CMD
The last saved command is displayed.

INPUT = <integer -100..-1>
Identifies the desired input relative to the current input.

INPUT = <integer 1..9999>
Identifies the desired input absolutely via its input serial number, which is automatically assigned by SDF when it is saved. The contents of the input buffer can be displayed with input serial numbers (see the SHOW-INPUT-HISTORY command, operand INPUT-SERIAL-NUMBER=*YES).

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
0	0	CMD0001	Command executed without errors
1	0	CMD0001	Cannot be specified since the input buffer is empty. Guaranteed message: CMD0558
2	0	CMD0559	AID command not supported in guided dialog. Guaranteed message: CMD0559
1	32	CMD0500	Syntax description in current syntax file invalid. Guaranteed message: CMD0500
	64	CMD0558	Command execution not successful

Example

```
/mod-f-attr sf.dummy,prot=(basic-acl=(owner=(y,y,y),group=(y,n,n),
                                      others=(y,n,n))) ────────────────── (1)
/show-job-sta job-id=tsn(1e9q),inf=envir ────────────────────────────── (2)
%  EXC0755 INFORMATION ON TASK WITH (&00) '1E9Q’ CANNOT BE GIVEN
/restore-sdf        ──────────────────────────────────────────────── (3)
/show-job-sta job-id=tsn(1e9w),inf=envir  ───────────────────────────── (4)
%NAME      TSN   STATION  PROCESSOR  HOLD MRSCAT
%         1E9W   $$$06002 FIREBALL

/cre-file test      ──────────────────────────────────────────────── (5)
/show-file-attr test,inf=min
%N NNN NW                3 :2OSG:$USER1.TEST

/show-input i-s-n=y ──────────────────────────────────────────────── (6)
"    1 :" mod-f-attr sf.dummy,prot=(basic-acl=(owner=(y,y,y),group=(y,n,n),
others=(y,n,n)))
/"   2 :" show-job-sta job-id=*tsn(1e9q),inf=envir
/"   3 :" show-job-sta job-id=*tsn(1e9w),inf=envir
/"   4 :" cre-file test
/"   5 :" show-file-attr test,inf=*min

/restore-sdf 1 ──────────────────────────────────────────────────── (7)
/mod-f-attr test    ,prot=(basic-acl=(owner=(y,y,y),group=(y,n,n),
                                      others=(y,n,n))) ────────────────── (8)
/restore-sdf
/mod-f-attr?test    ,prot=(basic-acl=(owner=(y,y,y),group=(y,n,n),
                                      others=(y,n,n)))  ───────────────── (9)
```

```
COMMAND  : MODIFY-FILE-ATTRIBUTES
OPERANDS : ...-NAME=TEST,PROTECTION=*PARAMETERS(BASIC-ACL=*PARAMETERS(OWNER=*PAR
           AMETERS(READ=*YES,WRITE=*YES,EXEC=*YES),GROUP=*PARAMETERS(READ=*YE
           S,WRITE=*NO,EXEC=*NO),OTHERS=*PARAMETERS(READ=*YES,WRITE=*NO,EXEC=...
--------------------------------------------------------------------------------
FILE-NAME              = TEST
NEW-NAME               = *SAME
SUPPORT                = *UNCHANGED
PROTECTION             = (PROTECTION-ATTR=*UNCHANGED,ACCESS=*BY-PROTECTION-ATTR,US
                         ER-ACCESS=*BY-PROTECTION-ATTR,BASIC-ACL=(OWNER=(READ=Y,WR
                         ITE=Y,EXEC=Y),GROUP=(READ=Y,WRITE=N,EXEC=N),OTHERS=(READ=
                         Y,WRITE=N,EXEC=N)),GUARDS=*BY-PROTECTION-ATTR,WRITE-PASSW
                         ORD=*BY-PROT-ATTR-OR-UNCH,READ-PASSWORD=*BY-PROT-ATTR-OR-
                         UNCH,EXEC-PASSWORD=*BY-PROT-ATTR-OR-UNCH,DESTROY-BY-DELET
                         E=*BY-PROTECTION-ATTR,AUDIT=*UNCHANGED,SPACE-RELEASE-LOCK
                         =*BY-PROTECTION-ATTR,EXPIRATION-DATE=*BY-PROTECTION-ATTR,
                         FREE-FOR-DELETION=*BY-PROT-ATTR-OR-UNCH)

--------------------------------------------------------------------------------
NEXT = *down(basic-acl) <------------------------------------------------- (10)
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

```
COMMAND  : MODIFY-FILE-ATTRIBUTES
STRUCTURE: BASIC-ACL=
OPERANDS : ...-NAME=TEST,PROTECTION=*PARAMETERS(BASIC-ACL=*PARAMETERS(OWNER=*PAR
           AMETERS(READ=*YES,WRITE=*YES,EXEC=*YES),GROUP=*PARAMETERS(READ=*YE
           S,WRITE=*NO,EXEC=*NO),OTHERS=*PARAMETERS(READ=*YES,WRITE=*NO,EXEC=...
--------------------------------------------------------------------------------
OWNER                  = (READ=Y,WRITE=Y,EXEC=Y)
GROUP                  = (READ=Y,WRITE=N,EXEC=y) <------------------------- (11)
OTHERS                 = (READ=Y,WRITE=N,EXEC=y)









--------------------------------------------------------------------------------
NEXT = *CONTINUE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F8=+   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

(1) BASIC-ACL protection is recorded in the catalog entry for the *SF.DUMMY* file (owners have unrestricted rights, groups and others have only read access).

(2) The SHOW-JOB-STATUS command is supposed to display information on the task with the TSN *1E9Q*, but no such task exists.

(3) The RESTORE-SDF-INPUT command is to redisplay the last input.

(4) The SHOW-JOB-STATUS command (see step 2) is output. In the character string which is output, the TSN is corrected to *1E9W* and sent off with DUE. Information on this TSN is displayed.

(5) The *TEST* file is cataloged. The protection rights are then output in abbreviated form via SHOW-FILE-ATTRIBUTES.

(6) SHOW-INPUT-HISTORY outputs the contents of the input buffer with input serial numbers (INPUT-SERIAL-NUMBER=*YES).

(7) RESTORE-SDF-INPUT redisplays the command with the serial number 45.

(8) In the displayed MODIFY-FILE-ATTRIBUTES, the file name is changed to *TEST* and sent off with DUE. The *TEST* file thus receives the same BASIC-ACL protection as the *SF.DUMMY* file did previously.

(9) The user wants to change the protection attributes of the *TEST* file again. RESTORE-SDF-INPUT displays the last command entered (see step 8). A question mark is inserted after the command name and the line is sent off with DUE.

(10) SDF switches to temporarily guided dialog and displays the operand form of the MODIFY-FILE-ATTRIBUTES command with the explicitly set specifications for BASIC-ACL. For an overview of the input, the user requests the subform for BASIC-ACL (*PARAMETERS structure) by entering "*down(basic-acl)" in the NEXT line.

(11) The user enters the desired values (groups and others receive execution rights with EXEC=*YES) in the subform and sends it off with DUE.

Note

The BASIC-ACL access rights can be changed to the shortest possible form by editing the command line output by RESTORE-SDF-INPUT (or by editing the corresponding command line in the output of the SHOW-INPUT-HISTORY command).

SHOW-CMD
Output command syntax description

Domain:	SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The SHOW-CMD command outputs the syntax description of a command. The name and type of the syntax file used are also displayed. CMD-NAME=*ALL produces a list of all command names. By using wildcards in the command name, the user can display a list of command names which match the search pattern.
Output can be directed to either SYSOUT or SYSLST.

The INFORMATION operand controls the scope of the syntax description. INFORMATION= *MINIMUM displays the command name, the name of the syntax file containing the syntax description, the operand names and any default values. Alternative operand values are displayed with INFORMATION=*MEDIUM, while with INFORMATION= *MAXIMUM help texts are also displayed.
The output only shows the syntax description of the assigned syntax files. In particular, task-specific default values are not displayed.

The FORM operand controls output of syntax objects whose input is not permitted in guided dialog. FORM=*UNGUIDED shows the command syntax, which can also be entered in unguided dialog, or a command list which also contains aliases of the command names. Restrictions regarding the input modes (interactive, batch, procedure) of commands or operands are indicated in the output with an asterisk (*). It is therefore possible, for example, to also find out information on operands which can only be input in batch mode.

The CHECK-PRIVILEGES operand determines whether the output will take account of the privileges of the user. If CHECK-PRIVILEGES=*YES is specified, the output only contains information on operands and operand values of commands which the user's privileges allow him/her to use. If a list of command names is output, commands which the user is not authorized to use are marked with an asterisk (*).

Format

```
SHOW-CMD

 CMD-NAME = *ALL / <structured-name 1..30 with-wild>
,INFORMATION = *MINIMUM / *MEDIUM / *MAXIMUM
,FORM = *GUIDED / *UNGUIDED
,CHECK-PRIVILEGES = *YES / *NO
,OUTPUT = *SYSOUT / *SYSLST(...)
    *SYSLST(...)
    |     SYSLST-NUMBER = *STD / <integer 1..99>
    |    ,LINES-PER-PAGE = *STD / *UNLIMITED / <integer 1..200>
```

Operands

CMD-NAME = *ALL / <structured-name 1..30 with-wild>
Name of the desired command.

CMD-NAME = *ALL
Lists all commands in alphabetical order. FORM=*UNGUIDED also displays defined aliases (in separate output lines).

CMD-NAME = <structured-name 1..30 with-wild>
Name of the command whose syntax is to be output. If you specify an alias, the real command name is displayed followed by the alias in parentheses.
If wildcards are used, all commands which match the search pattern are listed in alphabetical order. FORM=*UNGUIDED also outputs the wildcard string corresponding to the defined aliases (in separate output lines).

INFORMATION = *MINIMUM / *MEDIUM / *MAXIMUM
Determines the scope of the output. When lists of commands are output, the INFORMATION operand is ignored.

INFORMATION = *MINIMUM
The output contains the command names, operands and preset operand values.

INFORMATION = *MEDIUM
The output contains the command names, operands, preset operand values and alternative operand values.

INFORMATION = *MAXIMUM
The output contains the command names, operands, preset and alternative operand values and help texts.

FORM = <u>*GUIDED</u> / *UNGUIDED
Determines whether the syntax for guided or unguided dialog is to be output.

FORM = <u>*GUIDED</u>
Operands and operand values which are not allowed in guided dialog are not output.

FORM = *UNGUIDED
The output also contains operands and operand values which are not allowed in guided dialog. When lists of commands are displayed, these also contain the aliases in separate lines of output .

CHECK-PRIVILEGES = <u>*YES</u> / *NO
Specifies whether the output is to take account of the privileges of the user.

CHECK-PRIVILEGES = <u>*YES</u>
Information is displayed only on commands, operands and operand values which the user is authorized to use.

CHECK-PRIVILEGES = *NO
Information is displayed on all commands, operands and operand values, regardless of the user’s privileges.

OUTPUT = <u>*SYSOUT</u> / *SYSLST(...)
Specifies where the information is to be output.

OUTPUT = <u>*SYSOUT</u>
The information is output to the system file SYSOUT.

OUTPUT = *SYSLST(...)
The information is output ready for printing to the system file SYSLST.
The first byte of each output record is X'40'. The LINES-PER-PAGES operand defines after how many output records a header line with page feed is generated.

> **SYSLST-NUMBER = <u>*STD</u> / <integer 1..99>**
> Specifies whether the information is to be output to the system file SYSLST or to a SYSLST file from the set SYSLST01 through SYSLST99.
> The default is *STD, i.e. output is directed to the system file SYSLST.
>
> **LINES-PER-PAGE = <u>*STD</u> / *UNLIMITED / <integer 1..200>**
> Defines after how many output records a new page is to begin. Each printed page begins with a header line which contains a page-feed control character in the first byte. The header contains the name of the displayed command and the page number.
>
> **LINES-PER-PAGE = <u>*STD</u>**
> A new page begins after 55 output records.
>
> **LINES-PER-PAGE = *UNLIMITED**
> The output is not divided into printed pages. No header lines are output.

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed without errors
1	32	CMD0500	Syntax description in current syntax file invalid
1	64	CMD0812	Command execution not successful. Guaranteed messages: CMD0500, CMD0812

Examples

```
/show-cmd cmd=add-pass                ———————————————————————————————————————— (1)
%ADD-PASSWORD (ADD-PASS,ADPW)
%   FROM :2OSH:$TSOS.SYSSDF.BS2CP.140 (SYSTEM)
%        PASSWORD =
/show-cmd cmd=add-pass,information=max  ——————————————————————————————— (2)
%ADD-PASSWORD (ADD-PASS,ADPW)
%   FROM :2OSH:$TSOS.SYSSDF.BS2CP.140 (SYSTEM)
%Adds passwords for files or job variables to the password table of the
%task
%        PASSWORD =
%            -list-possible (63)-: x-string_1..8 or c-string_1..4 or
%            integer_-2147483648..2147483647
%            Specifies passwords to be added to the password table
/show-cmd cmd=write*                ——————————————————————————————————————————— (3)

% WRITE-ACCOUNTING-RECORD
% WRITE-DISKETTE
% WRITE-SPOOL-TAPE
% WRITE-TEXT
```

(1) The syntax of the ADD-PASSWORD command is output.

(2) The syntax of the ADD-PASSWORD command is output with all operands, operand values and help texts (INFORMATION=*MAXIMUM).

(3) All commands beginning with the string “WRITE” are output.

SHOW-INPUT-DEFAULTS
Output task-specific default values

Domain:	SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The SHOW-INPUT-DEFAULTS command outputs all currently defined task-specific default values. Within the task, default values can be defined for commands and statements. The user can either display all default values or restrict the display to default values of commands or statements. If only defaults of commands/statements are to be displayed, the output can be restricted to specific commands or statements.
Output can be directed to either SYSOUT or SYSLST.

Task-specific default values can be deleted using the RESET-INPUT-DEFAULTS command. To identify a specific default value in the RESET-INPUT-DEFAULTS command, the output of the SHOW-INPUT-DEFAULTS command can be requested with input serial numbers (operand INPUT-SERIAL-NUMBER=*YES).

In programs with an SDF interface, SHOW-INPUT-DEFAULTS is available as a standard statement with the same functionality.

Format

```
SHOW-INPUT-DEFAULTS                                              Alias: SHID

OBJECT = *CMD (...) / *STMT(...) / *ALL

   *CMD(...)

     |  CMD = *ALL / <structured-name 1..30 with-wild(50)>

   *STMT(...)

     |  STMT = *ALL / <structured-name 1..30 with-wild(50)>
     |
     | ,PROGRAM = *CURRENT / *ALL / <structured-name 1..30>

,OUTPUT = *SYSOUT / *SYSLST(...)

   *SYSLST(...)

     |  SYSLST-NUMBER = *STD / <integer 1..99>

,INPUT-SERIAL-NUMBER = *NO / *YES
```

Operands

OBJECT = *CMD(...) / *ALL / *STMT
Specifies the type of input for which the task-specific default values are to be output.

OBJECT = *CMD(...)
Only the task-specific default values of commands are output. The defaults of all or only of selected commands can be requested.

CMD = *ALL / <structured-name 1..30 with-wild(50)>
Specifies whether the task-specific default values of all commands or only of selected commands are to be output.

CMD = *ALL
All task-specific default values of commands are output.

CMD = <structured-name 1..30 with-wild(50)>
Name of the command whose task-specific default values are to be output. If wildcards are used, the default values of all commands which match the specified search pattern will be displayed.

OBJECT = *STMT(...)
Only the task-specific default values of statements are displayed. The user can request the default values of all or only of selected statements of a program to be output.

STMT = <u>*ALL</u> / <structured-name 1..30 with-wild(50)>
Specifies whether the task-specific default values of all statements or only of selected statements are to be output. In the PROGRAM operand the user can specify whether the output is to contain default values of statements of a specific program or of all programs.

STMT= <u>*ALL</u>
All task-specific default values of statements are output.

STMT = <structured-name 1..30 with-wild(50)>
Name of the statement whose task-specific default values are to be output. If wildcards are used, the default values of all statements which match the specified wildcard string will be output.

PROGRAM = <u>*CURRENT</u> / *ALL / <structured-name 1..30>
Specifies the program for whose statements specified in the STMT operand the task-specific default values are to be output.

PROGRAM = <u>*CURRENT</u>
Only default values of statements of the program currently defined in the SDF options are output. The program name can be set using the MODIFY-SDF-OPTIONS command (DEFAULT-PROGRAM-NAME operand).

PROGRAM = *ALL
The default values of all statements are output, regardless of the program name.

PROGRAM = <structured-name 1..30>
Program name, defined in a currently assigned syntax file.
Only default values of statements of the specified program are output.

OBJECT = *ALL
All task-specific default values, i.e. from both commands and statements, are output.

OUTPUT = <u>*SYSOUT</u> / *SYSLST(...)
Specifies where the information is to be output.

OUTPUT = <u>*SYSOUT</u>
The information is output to the system file SYSOUT.

OUTPUT = *SYSLST(...)
The information is output to the system file SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Specifies whether the information is to be output to the system file SYSLST or to a SYSLST file from the set SYSLST01 through SYSLST99.
The default is *STD, i.e. output is directed to the system file SYSLST.

INPUT-SERIAL-NUMBER = *NO / *YES
Specifies whether the inputs are to be shown with their input serial numbers. The default value is *NO for output without input serial numbers. The input serial number can be used in the RESET-INPUT-DEFAULTS command to delete a specific default value.

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed without errors
1	0	CMD0001	No task-specific default value that matches the specifications exists. Guaranteed message: CMD0561
1	32	CMD0500	Syntax description in current syntax file invalid. Guaranteed message: CMD0500
1	64	CMD0561	Command execution not successful.. Guaranteed message: CMD0561

Example

See the RESET-INPUT-DEFAULTS command.

SHOW-INPUT-HISTORY Output buffered input to SYSOUT

Domain:	SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The SHOW-INPUT-HISTORY command outputs the contents of the input buffer to SYSOUT. This enables information to be provided on previous inputs. The user can then request a specific input via the RESTORE-SDF-INPUT command and enter it again, either as it is or in modified form, without having to retype it. For an input to be reused, however, at least one character in the input string must be changed (e.g. you can overwrite a character with the same character).

For identification purposes in the RESTORE-SDF-INPUT command, the information can be requested with input serial numbers (operand INPUT-SERIAL-NUMBER=*YES).

The user can restrict the scope of the output of the input buffer as follows:
– Specify the maximum number of saved inputs to be displayed (ENTRIES operand); the default value is eight.
– Specify the desired input type (SELECT operand) - either commands or statements; the default value is commands.
– Specify a wildcard search pattern (PATTERN operand). Only commands or statements which match this string are displayed; the default value is no search pattern used.

The input buffer is controlled (activated/deactivated and deleted) via the MODIFY-SDF-OPTIONS command. Inputs in guided dialog are saved in ACCEPTED form, while inputs in unguided dialog are saved in INPUT form.
The SHOW-INPUT-HISTORY command or statement is not saved.

Values specified for “secret” operands which match neither the default value nor a value defined via SECRET=*NO are saved in the input buffer with “^”.
In unguided dialog when these values are displayed again via RESTORE-SDF-INPUT, the user can do one of the following:

– send off the command/statement unchanged. In this case, SDF displays a blanked input field for each secret operand for the user to enter the desired value.
– delete the “^” and insert the desired value directly before sending off the command/statement.

In programs with an SDF interface, SHOW-INPUT-HISTORY is available as a standard statement with the same functionality.

Format

SHOW-INPUT-HISTORY	Alias: **SHIH**
ENTRIES = **8** / <integer 1..100> / ***ALL**	
,**SEL**ECT = ***CMD** / ***ALL** / ***STMT**	
,**PATTERN** = ***NONE** / <structured-name 1..30 with-wild>	
,**INPUT-SERIAL-NUMBER** = ***NO** / ***Y**ES	

Operands

ENTRIES = 8 / <integer 1..100> / *ALL
Determines the maximum number of entries to be displayed. If *ALL is specified, the entire contents of the buffer are displayed.

SELECT = *CMD / *ALL / *STMT
Specifies the type of entries to be displayed.

SELECT = *CMD
Only commands are displayed.

SELECT = *ALL
Commands and statements are displayed.

SELECT = *STMT
Only statements are displayed.

PATTERN = *NONE / <structured-name 1..30 with-wild>
Specifies whether the entries to be displayed are to be selected according to a pattern string. The default value is *NONE (i.e. no pattern string). If a pattern string is specified, only entries which match it are displayed.

INPUT-SERIAL-NUMBER = *NO / *YES
Specifies whether the entries are to be displayed with their input serial numbers. The default value is *NONE (i.e. input serial numbers not shown). The input serial numbers can be used with RESTORE-SDF-INPUT to select the desired entry.

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed without errors
1	0	CMD0001	Cannot be specified since the input buffer is empty. Guaranteed message: CMD0560
1	32	CMD0500	Syntax description in current syntax file invalid. Guaranteed message: CMD0500
1	64	CMD0560	Command execution not successful. Guaranteed message: CMD0560

Example

See the RESTORE-SDF-INPUT command.

SHOW-RETURNCODE Output the command return code of the last command

Domain:	SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The SHOW-RETURNCODE command outputs the command return code of the last command entered to SYSOUT.

Format

SHOW-RETURNCODE	Alias: **SHRTC**

The command has no operands and is executed immediately.

Command return code

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Command executed without errors

Example

```
/cre-file test
%  CMD0051 INVALID OPERAND 'FILE-NAME'
%  DMS0683 FILE ':20SG:$USER1.TEST' ALREADY EXISTS
/show-returncode
%Returncode:
%Subcode2  = 0
%Subcode1  = 64
%Maincode  = DMS0683
```

SHOW-SDF-OPTIONS
Show active syntax files and SDF options

Domain:	SDF
Privileges:	STD-PROCESSING TSOS HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The SHOW-SDF-OPTIONS command displays the names and versions of the currently active syntax files as well as the current settings of the SDF options. For syntax files, it always displays their real file names, even if the user entered a MODIFY-SDF-OPTIONS command in order to specify a user syntax file using the appropriate alias name declared in the alias catalog (see the ADD-ALIAS-CATALOG-ENTRY command).

The INFORMATION operand controls the scope of the information to be displayed. INFORMATION=*USER provides information only on settings which the user can modify via MODIFY-SDF-OPTIONS for each task: this information includes the names and versions of all activated user syntax files and all settings of the SDF options.

The MODIFY-SDF-OPTIONS command can then be used to activate or deactivate user syntax files and to change the settings of SDF options for a specific task.

If the SDF-P subsystem is used (as of V2.0A), output can be in S variables.

Format

SHOW-SDF-OPTIONS	Alias: **SHSDFO**
INFORMATION = ***ALL** / ***USER**	

Operands

INFORMATION = *ALL / *USER
Determines the scope of the information to be displayed.

INFORMATION = *ALL
Outputs information on the activated syntax files and the current settings of the SDF options.

INFORMATION = *USER
Outputs information on the activated user syntax files and the current settings of the SDF options.

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors

Output format

The output fields and values under the heading “CURRENT SDF OPTIONS” correspond to the operands and operand values of the MODIFY-SDF-OPTIONS command.

Output field	Meaning and possible values
SYNTAX FILES CURRENTLY ACTIVATED:	Header line for information on the activated syntax files
SYSTEM	Name of the active system syntax file, or *NONE if only a group syntax file is assigned
VERSION	Version of the system syntax file
SUBSYSTEM	Name of an active subsystem syntax file
VERSION	Version of the subsystem syntax file
GROUP	Name of an active group syntax file
VERSION	Version of the group syntax file; omitted if no user syntax file is activated (GROUP=*NONE)
USER	Name of an active user syntax file
VERSION	Version of the user syntax file; omitted if no user syntax file is activated (USER=*NONE)

Table 8: Output format of the SHOW-SDF-OPTIONS command (part 1 of 2)

Output field	Meaning and possible values
CURRENT SDF OPTIONS:	Header line for information on the current settings of the SDF options
GUIDANCE	Guidance level: *EXPERT / *NO / *MINIMUM / *MEDIUM / *MAXIMUM
LOGGING	Scope of the logging of commands and statements: *INPUT-FORM / *ACCEPTED-FORM / *INVARIANT-FORM
CONTINUATION	Command continuation: *OLD-MODE / *NEW-MODE
UTILITY-INTERFACE	Input interface for utilities: *OLD-MODE / *NEW-MODE
PROCEDURE-DIALOGUE	Syntax error dialog or help dialog in procedures: *YES /*NO
MENU-LOGGING	Menu logging: *YES /*NO
CMD-STATISTICS	*Output for privileged users only (TSOS)* Statistical recording of commands: *YES /*NO
MODE	Syntax check of commands: *EXECUTION / *TEST
CHECK-PRIVILEGES	Check privileges in the test mode: *YES / *NO
DEFAULT-PROGRAM-NAME	Syntax check of statements of a program: *NONE / <structured-name 1..20>
FUNCTION-KEYS	Function key assignment: *OLD-MODE / *NEW-MODE
INPUT-HISTORY	Input buffer: *ON / *OFF
NUMBER-OF-INPUTS	Size of the input buffer (maximum number of inputs that can be saved): <integer 1..100>
PASSWORD-PROTECTION	Protection of secret operand values in the input buffer: *YES / *NO

Table 8: Output format of the SHOW-SDF-OPTIONS command (part 2 of 2)

Output in S variables

Output information	Name of the S variable	T	Contents
Consideration of privileges during syntax check	var(*LIST).CHECK-PRIV	S	*NO *YES
Statistical recording of commands	var(*LIST).CMD-STATIS	S	Empty string *NO *YES
Command continuation type	var(*LIST).CONTI	S	*NEW-MODE *OLD-MODE
Function key assignment	var(*LIST).FUNC-KEY	S	*OLD-MODE *STYLE-GUIDE-MODE
Dialog guidance	var(*LIST).GUIDE	S	*EXPERT *MAX *MED *MIN *NO
Recording in the input buffer	var(*LIST).INPUT-HIST	S	*OFF *ON
Command logging	var(*LIST).LOG	S	*ACCEPT-FORM *INPUT-FORM *INVARIANT-FORM
Menu logging	var(*LIST).MENU-LOG	S	*NO *YES
Syntax check of commands	var(*LIST).MODE	S	*EXEC *TEST
Size of the input buffer	var(*LIST).NUM-OF-INPUT	I	<integer 0..100>
Recording of passwords and ISP commands in the input buffer	var(*LIST).PASSWORD-PROT	S	*NO *YES
Syntax error dialog	var(*LIST).PROC-DIALOG	S	*NO *YES
Name of the syntax file	var(*LIST).SF(*LIST).F-NAME	S	<filename 1..54>
Type of the syntax file	var(*LIST).SF(*LIST).TYPE	S	*GROUP *SUBSYS *SYS *USER
Version of the syntax file	var.SF(*LIST).VERSION	S	<text 1..12>

Table 9: Output variables of the SHOW-SDF-OPTIONS command (part 1 of 2)

Output information	Name of the S variable	T	Contents
Program name for checking the syntax of statements	var(*LIST).TEST-PROG-NAME	S	*NONE <structured-name 1..30>
Input interface for utilities	var(*LIST).UTILITY-INTERF	S	*NEW-MODE *OLD-MODE

Table 9: Output variables of the SHOW-SDF-OPTIONS command (part 2 of 2)

Note on the contents of variables:

CMD-STATIS: For non-privileged users this variable contains only the empty string.

– F-NAME, TYPE and VERSION:
  Only elements for activated syntax file types are created. If there is no user syntax file activated, the elements are created one time anyway for syntax files where TYPE= *USER. F-NAME and VERSION contain only an empty string.
  When INFORMATION=*USER is specified the information on system, subsystem and group syntax files is omitted.
  VERSION contains the version specified in the global information of the syntax file. If no version was specified for it, then it contains the value UNDEFINED.

Examples

```
/show-sdf-opt  ———————————————————————————————————————————————  (1)
%SYNTAX FILES CURRENTLY ACTIVATED :
%  SYSTEM    : :2OSH:$TSOS.SYSSDF.SDF.045
%              VERSION : SESD04.5A300
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.ACS.140
%              VERSION : SESD14.0B100
   .
   .
   .
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.SPACEOPT.020
%              VERSION : SESD02.0A300
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.SDF-A.041
%              VERSION : SESD04.1E10
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.TASKDATE.140
%              VERSION : SESD14.0A100
%  GROUP     : *NONE
%  USER      : :2OSG:$USER1.SDF.USER.SYNTAX
%              VERSION : UNDEFINED
%CURRENT SDF OPTIONS :
%  GUIDANCE          : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES

/mod-sdf-opt syntax=*add(syssdf.user.special.01) ———————————————————— (2)
/show-sdf-opt inf=*user    ——————————————————————————————————————————— (3)

%  USER      : :2OSG:$USER1.SDF.USER.SYNTAX
%              VERSION : UNDEFINED
%  USER      : :2OSG:$USER1.SYSSDF.USER.SPECIAL.01
%              VERSION : UNDEFINED
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
```

(1) The SHOW-SDF-OPTIONS command displays information on all activated syntax files and the SDF options set. The output of the subsystem syntax files can be very extensive, depending on the number of activated subsystems. This example shows only an excerpt. The user syntax file `SYSSDF.SDF.USER.SYNTAX` is activated.

(2) The user syntax file `SYSSDF.USER.SPECIAL01` is also activated with the MODIFY-SDF-OPTIONS command.

(3) The SHOW-SDF-OPTIONS command with INFORMATION=*USER provides information on the user syntax file and the SDF options set.

SHOW-SYNTAX-VERSIONS Display syntax file versions

Domain:	SDF
Privileges:	STANDARD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The SHOW-SYNTAX-VERSIONS command outputs to SYSOUT information on the system, subsystem and group syntax files currently activated for the task. This information includes the names and version statuses of all currently valid syntax descriptions of the software units and components used.
This information can also be requested for a specific software unit or component or a list of software units or components.

If the SDF-P subsystem is used (as of V2.0A), output can be in S variables.

Format

SHOW-SYNTAX-**VERS**IONS
SOFTWARE-UNIT-NAME = ***ALL** / list-poss(2000): <structured-name 1..16>

Operands

SOFTWARE-UNIT-NAME = *ALL / list-poss(2000): <structured-name 1..16>
Specifies the names of the software units and components on which information is to be output.

Return codes

(SC2)	SC1	Maincode	Meaning / Guaranteed messages
	0	CMD0001	Command executed without errors
1	32	CMD0500	Syntax description in current syntax file invalid
1	64	CMD0811	Command execution not successful. Guaranteed messages: CMD0300, CMD0500, CMD0811

Output in S variables

Output information	Name of the S variable	T	Contents
Path name of the syntax file	var(*LIST).F-NAME	S	<full-filename 1..54>
Name of the software component	var(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).NAME	S	<structured-name 1..13>
Version of the software component	var(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).VERSION	S	<text 3..3> / <text 8..8>
Name of the software unit	var(*LIST).SW-UNIT(*LIST).NAME	S	<structured-name 1..16>
Version of the software unit	var(*LIST).SW-UNIT(*LIST).VERSION	S	<text 7..8>
Type of the syntax file	var(*LIST).TYPE	S	*GROUP *SYS

Table 10: Output variables of the SHOW-SYNTAX-VERSIONS command

Note on the contents of variables:

*SYS	Identifies a system or subsystem syntax file.

Examples

```
/show-syn-vers (sdf-p,lms)  ——————————————————————————————————————————————  (1)
%SYNTAX VERSION OF SDF-P              :
%FOR SYSTEM SYNTAX FILE :
%:2OSH:$TSOS.SYSSDF.SDF-P.021
%-------------------------------------------------------
% SDF-P              = 02.1C100
%                        CLI              = 550
%
%SYNTAX VERSION OF LMS                :
%FOR SYSTEM SYNTAX FILE :
%:2OSH:$TSOS.SYSSDF.LMS.033
%-------------------------------------------------------
% LMS                = 03.3A200
%                        LMS              = 033
%

/show-syn-vers sdf  ——————————————————————————————————————————————————————  (2)
%SYNTAX VERSION OF SDF               :
%FOR SYSTEM SYNTAX FILE :
%:2OSH:$TSOS.SYSSDF.SDF.045
%-------------------------------------------------------
% SDF               = 04.5A300
%                       CMD              = 045
%
```

(1) Information on the software units SDF-P and LMS is output.

(2) Information on the software unit SDF is output.

WRITE-TEXT Write text to SYSOUT or SYSLST

Domain:	JOB PROCEDURE SDF
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SAT-FILE-EVALUATION SAT-FILE-MANAGEMENT SECURITY-ADMINISTRATION

Function

The WRITE-TEXT command outputs the specified text to SYSOUT or to SYSLST.

In programs with an SDF interface, WRITE-TEXT is available as a standard statement with identical syntax and functionality.

Format

```
WRITE-TEXT                                                    Alias: WRTX

 TEXT = '␣' / <c-string 1..1024 with-low>
,OUTPUT = *SYSOUT / *SYSLST(...)
   *SYSLST(...)
     |  SYSLST-NUMBER = *STD / <integer 1..99>
```

Operands

TEXT = '␣' / <c-string 1..1024 with-low>
Text to be output to SYSOUT or SYSLST. The default value is a string containing a blank.

OUTPUT = *SYSOUT / *SYSLST(...)
Specifies where the information is to be output.

OUTPUT = *SYSOUT
The information is to be output to the system file SYSOUT.

OUTPUT = *SYSLST(...)
The information is to be output to the system file SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Specifies whether the information is to be output to the system file SYSLST or to a SYSLST file from the set SYSLST01 through SYSLST99.
The default is *STD, i.e. output is directed to the system file SYSLST.

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without error

Example

A procedure file (`PROC.1`) begins with the following commands:

```
/BEGIN-PROC  PAR=*YES(PROC-PAR=(&FILE=))
/WRITE-TEXT 'procedure for linking'
...
...
...
/WRITE-TEXT 'procedure ended without error'
/END-PROC
```

When the procedure is run, the text specified in WRITE-TEXT will be output to SYSOUT:

```
/call-proc name=proc.1
procedure for linking
  .
  .
  .
procedure ended without error
```

7.2 Standard statements

Programs which read their statements via SDF have so-called standard statements, whose functionality is independent of the program-specific statements.

The standard statements are automatically available via the basic syntax file (see page 156) of all programs with an SDF interface.

The standard statements are intended to promote user interface standardization. They offer functions which are generally regarded as useful in any program. Such identical functions should therefore be able to be addressed with one standardized statement name.

END

The END statement terminates input to the program called. It has no operands and is executed immediately.

END

EXECUTE-SYSTEM-CMD

The EXECUTE-SYSTEM-CMD statement allows a command to be executed during the program run. This statement interrupts the program that has been called, issues the specified command and, after its execution, returns to program mode. In the event of errors in command execution, the spin-off mechanism is activated at program level (see standard statement STEP, page 231).
The statement is rejected if the program interrupt is currently not allowed within the procedure or by the program itself (e.g. for security reasons).

Note

This statement is only supported as of BS2000/OSD-BC V2.0.

EXECUTE-**SYS**TEM-**CMD**
CMD = <text 0..1800 with-low>

CMD = <text 0..1800 with-low>
Command to be executed without a leading slash, enclosed in parentheses.

HELP-MSG-INFORMATION

The HELP-MSG-INFORMATION statement outputs the text of a system message to SYSOUT. It also allows the user to request explanations of messages and to define the language in which the text is to be output.
The statement functions in the same way as the command of the same name.

Note

The operands must be entered as shown below. They are subject to the SDF abbreviation rules. SDF functions, such as obtaining information on possible operand values or correction dialogs, are not available at operand level.

HELP-MSG-INFORMATION	Alias: **HP** / **HPMSGI**
MSG-IDENTIFICATION = ***LAST** / <alphanum-name 4..7>	
,**INF**ORMATION**-LEV**EL = ***MAX**IMUM / ***MED**IUM / ***MIN**IMUM	
,**LANG**UAGE = ***STD** / <name 1..1>	

HOLD-PROGRAM

The HOLD-PROGRAM statement interrupts a program waiting for input, thereby enabling the user to enter commands. The RESUME-PROGRAM command terminates the command input and returns to program mode.
The HOLD-PROGRAM statement functions in the same way as the command of the same name.
The statement is rejected if the program interrupt is currently not allowed within the procedure or by the program itself (e.g. for security reasons).

Note

This statement is only supported as of BS2000/OSD-BC V2.0.

HOLD-PROGRAM

MODIFY-SDF-OPTIONS

The MODIFY-SDF-OPTIONS statement permits the user to activate/deactivate a user syntax file, and to change SDF settings, during program execution. Its function range is identical to that of the MODIFY-SDF-OPTIONS command. Test mode and the command statistics, however, can only be specified at command level.

```
MODIFY-SDF-OPTIONS                                                        Alias: MDSDFO

SYNTAX-FILE = *UNCHANGED / *ADD(...) / *REMOVE(...) / *NONE
   *ADD(...)
     |  ADD-NAME = *STD / list-poss(2000): *STD / <filename 1..54>
   *REMOVE(...)
     |  REMOVE-NAME = *LAST / *ALL / *BY-SELECTION  / list-poss(2000): <filename 1..54> / *STD
,GUIDANCE = *UNCHANGED / *EXPERT / *NO / *MAXIMUM / *MEDIUM / *MINIMUM
,LOGGING = *UNCHANGED / *INPUT-FORM / *ACCEPTED-FORM / *INVARIANT-FORM
,UTILITY-INTERFACE = *UNCHANGED / *OLD-MODE / *NEW-MODE
,PROCEDURE-DIALOGUE = *UNCHANGED / *YES / *NO
,CONTINUATION = *UNCHANGED / *OLD-MODE / *NEW-MODE
,MENU-LOGGING = *UNCHANGED / *NO / *YES
,FUNCTION-KEYS = *UNCHANGED / *STYLE-GUIDE-MODE / *BY-TERMINAL-TYPE / *OLD-MODE
,INPUT-HISTORY = *UNCHANGED / *ON(...) / *OFF / *RESET
   *ON(...)
     |  NUMBER-OF-INPUTS = *UNCHANGED / <integer 1..100>
     | ,PASSWORD-PROTECTION = *UNCHANGED / *NO / *YES
```

REMARK

The REMARK statement identifies an ensuing text as a comment and is only of relevance for the documentation of job/procedure execution. A semicolon outside the brackets is interpreted as a statement separator, i.e. any characters that follow are interpreted as the next statement.

```
REMARK

TEXT = <cmd-rest 0..1800>
```

RESET-INPUT-DEFAULTS

The RESET-INPUT-DEFAULTS statement allows the user to delete task-specific default values for statements or commands.
The statement functions in the same way as the command of the same name. However, the default value in this case is the deletion of the task-specific default values of statements.

```
RESET-INPUT-DEFAULTS                                              Alias: RSID

OBJECT = *STMT(...) / *CMD (...) / *ALL / <integer 1..9999>
  *STMT(...)
    | STMT = *ALL / <structured-name 1..30 with-wild(50)>
    | ,PROGRAM = *CURRENT / *ALL / <structured-name 1..30>
  *CMD(...)
    | CMD = *ALL / <structured-name 1..30 with-wild(50)>
```

RESTORE-SDF-INPUT

The RESTORE-SDF-INPUT statement redisplays an input that has already been entered and is stored in the input buffer.
The statement functions in the same way as the command of the same name. However, the default value is the output of the last statement stored.

```
RESTORE-SDF-INPUT                                                 Alias: RRSDFI

INPUT = *LAST-STMT / <integer -100..-1> / <integer 1..9999>
```

INPUT = *LAST-STMT
The last saved statement is displayed.

See the RESTORE-SDF-INPUT command for an application example.

SHOW-INPUT-DEFAULTS

The SHOW-INPUT-DEFAULTS statement allows the user to find out about all currently defined task-specific default values.
The statement functions in the same way as the command of the same name. However, the default value in this case is the output of the task-specific default values of statements.

SHOW-INPUT-DEFAULTS	Alias: **SHID**
OBJECT = ***STMT**(...) / ***CMD**(...) / ***ALL**	
***STMT**(...)	
STMT = ***ALL** / <structured-name 1..30 with-wild(50)>	
,**PROG**RAM = ***CURR**ENT / ***ALL** / <structured-name 1..30>	
***CMD**(...)	
CMD = ***ALL** / <structured-name 1..30 with-wild(50)>	
,**OUTPUT** = ***SYSOUT** / ***SYSLST**(...)	
***SYSLST**(...)	
SYSLST-NUMBER = ***STD** / <integer 1..99>	
,**INP**UT-**SERIAL-NUM**BER = ***NO** / ***Y**ES	

SHOW-INPUT-HISTORY

The SHOW-INPUT-HISTORY statement outputs the contents of the input buffer to SYSOUT. The statement functions in the same way as the command of the same name, except that the output of statements is preset.

SHOW-INPUT-HISTORY	Alias: **SHIH**
ENTRIES = **8** / <integer 1..100> / ***ALL**	
,**SEL**ECT = ***STMT** / ***CMD** / ***ALL**	
,**PATTERN** = ***NONE** / <structured-name 1..30>	
,**INPUT-SERIAL-NUMBER** = ***NO** / ***Y**ES	

SHOW-SDF-OPTIONS

The SHOW-SDF-OPTIONS statement is used to call down up-to-date information on all active syntax files and SDF settings for a particular user task. If the program has opened a separate syntax file hierarchy, the names of these syntax files are displayed.
The function of this statement corresponds to that of the SHOW-SDF-OPTIONS command.

SHOW-SDF-OPTIONS	Alias: **SHSDFO**
INFORMATION = ***ALL** / ***USER**	

SHOW-STMT

The user can display the syntax description of a statement of the currently loaded program with the SHOW-STMT statement. The user receives a list of all statements using STMT-NAME=*ALL. When wildcards are used, a list of the functions matching the wildcard expression is output.
The statement functions just like the command of the same name, except that the syntax descriptions of statements are output instead of the syntax descriptions of commands (the STMT-NAME operand corresponds in this case to the CMD-NAME operand).

SHOW-STMT
STMT-NAME = ***ALL** / <structured-name 1..30 with-wild>
,**INF**ORMATION = ***MIN**IMUM / ***MED**IUM / ***MAX**IMUM
,**FORM** = ***GUIDED** / ***UNGUIDED**
,**CHECK-PRIV**ILEGES = ***Y**ES / ***NO**
,**OUTPUT** = ***SYSOUT** / ***SYSLST**(...)
***SYSLST**(...)
SYSLST-NUMBER = ***STD** / <integer 1..99>
,**LINE**S-**P**ER-**PAGE** = ***STD** / ***UNLIM**ITED / <integer 1..200>

STEP

The STEP statement identifies a section of program statements within a command file. If an errored statement is issued, the spin-off mechanism is initiated. This means that all subsequent statements up to such a section are ignored. If no STEP statement is encountered prior to the END statement, the program receives a return code to which it can react with abnormal termination. Following abnormal termination, spin-off continues at command level (see the SET-JOB-STEP command in Volume 4 of the “Commands” manual [1]).

STEP

WRITE-TEXT

The WRITE-TEXT statement outputs any specified text to SYSOUT.

WRITE**-TEXT**	Alias: **WRTX**
TEXT = '␣' / <c-string 1..1024 with-low>	
,**OUTPUT** = ***SYSOUT** / ***SYSLST**(...)	
***SYSLST**(...)	
│ **SYSLST-NUM**BER = ***STD** / <integer 1..99>	

7.3 SDF-I statements

The SDF-I utility routine is described in detail in the “SDF Management” manual [5]. The program is started with the START-SDF-I command. Below, only the CONVERT-SYNTAX-FILE and SHOW-SYNTAX-FILE statements are discussed because they also contain functions for non-privileged users (processing of user syntax files). User syntax files created with an SDF-A version earlier than V2.0 should be converted, for performance reasons, to the new format supported by SDF V2.0.

CONVERT-SYNTAX-FILE

The CONVERT-SYNTAX-FILE statement enables any syntax file (group, system, user) in old format (SDF < V2.0A) to be converted into a new-format syntax file that is compatible with SDF V4.1 and higher.

CONVERT-**SYN**TAX-**FILE**
INPUT-**FILE** = <filename 1..54>
,**OUT**PUT-**FILE** = <filename 1..54>

INPUT-FILE = <filename 1..54>
Name of the old-format syntax file.

OUTPUT-FILE = <filename 1..54>
Name of the new-format syntax file to be created.

SHOW-SYNTAX-FILE

Outputs information on a syntax file. This statement corresponds largely to the SDF command SHOW-SYNTAX-VERSIONS, but also shows the syntax file type (SYSTEM or GROUP) and the syntax file format (V1, V2, V3, V4 or V4.1).

SHOW-SYNTAX-FILE
FILE = ***CUR**RENT / ***INP**UT-**FILE** / <filename 1..54> **,INF**ORMATION = **ALL-ATTR**IBUTES / **VERS**IONS / **GLOB**ALS / **CMD-INTERF**ACE ,**PROD**UCT-**NAME** = ***ALL** / <structured-name 1..16>

FILE =
Name of the syntax file about which information is desired.

FILE = *CURRENT
This operand value is only allowed after at least one preceding OPEN statement. The information refers to the syntax file specified under INPUT-FILE (immediately after an OPEN statement) or to the current temporary SDF-I work file (after a MERGE statement).

FILE = *INPUT-FILE
This operand value is only allowed after at least one preceding OPEN statement. The information refers to the input syntax file specified under INPUT-FILE in the OPEN statement.

FILE = <filename 1..54>
The information refers to the explicitly specified syntax file.

INFORMATION =
Determines the type of the displayed information. All output includes the type (GROUP or SYSTEM) and format (V1, V2, V3, V4 or V4.1) of the syntax file.

INFORMATION = ALL-ATTRIBUTES
The output additionally includes the versions of the software units contained (for group and system syntax files only) and the global information. In the case of syntax files created by customers, the version is displayed with **CUSTOM**.

INFORMATION = VERSIONS
The output additionally includes the versions of the software units contained (for group and system syntax files only). In the case of syntax files created by customers, the version is displayed with **CUSTOM**.

INFORMATION = *GLOBALS
The output additionally includes the global information.

INFORMATION = CMD-INTERFACE
The output additionally includes a list of commands.

PRODUCT-NAME =
This operand is only effective when INFORMATION=CMD-INTERFACE is specified and determines whether command output is to be restricted to a specific product.

PRODUCT-NAME = <u>*ALL</u>
All commands of the specified syntax file are output. Commands written by the user that were not merged are not displayed in the output of an SESD (software unit syntax file) or INSD (installation syntax file) supplied by Fujitsu Siemens Computers.

PRODUCT-NAME = <structured-name 1..16>
All commands of a selected product are output.
If commands written by the user that were merged into an SESD (software unit syntax file) or INSD (installation syntax file) supplied by Fujitsu Siemens Computers are to be displayed, then the REMOVE-ID used for the merge must be specified as the product name.

7.4 SDF macros

The SDF user interface can also be employed for user programs. For this purpose, SDF offers the following macro calls (for a detailed description see the “SDF-A” manual [3]):

CLSCALL	Closes a syntax file hierarchy opened in the program
CMDALLW	Generates a list of permissible operations
CMDANALY	Generates EQUATEs for return codes
CMDCST	Initiates a semantic error dialog
CMDMEM	Generates a transfer area for status information
CMDRC	Sets a command return code
CMDRETC	Generates a DSECT for the command return code
CMDRST	Reads and analyzes a statement and returns the results
CMDSEL	Creates a selection menu for defined objects and returns the selection result (both single and multiple selection are possible)
CMDSTA	Provides information on active syntax files
CMDSTRUC	Generates a standardized transfer area for an analyzed statement in the old format. The transfer area is only evaluated by the RDSTMT, TRSTMT and CORSTMT macro calls that are still compatibly supported (for the new format, see CMDTA).
CMDTA	Generates a standardized transfer area for an analyzed statement in the new format
CMDTST	Analyzes a statement
CMDVAL	Checks whether a value matches a data type
CMDWCC	Checks wildcard syntax and compares the patterns
CMDWCO	Generates a name from a selector, a name and a constructor
CORSTMT	Initiates a semantic error dialog. The macro call uses the standardized transfer area in the old format and is only supported when compatible. Applications that use the new format of the standardized transfer area must use the CMDCST macro call.
OPNCALL	Opens a separate syntax file hierarchy at program level
RDSTMT	Reads and analyzes a statement and returns the results. The macro call uses the standardized transfer area in the old format and is only supported when compatible. Applications that use the new format of the standardized transfer area must use the CMDRST macro call.

TRCMD	Analyzes a command input and generates from it the desired logging format (cf. section “Logging” on page 100).
TRSTMT	Analyzes a statement. The macro call uses the standardized transfer area in the old format and is only supported when compatible. Applications that use the new format of the standardized transfer area must use the CMDTST macro call.

7.5 SDF interface to high-level programming languages

As of V3.0, SDF offers an interface for programs written in **H**igh **L**evel **L**anguages. Programs written in any of the programming languages COBOL, FORTRAN or C can use the functions of SDF macros directly, i.e. no Assembler subprograms are required in order to provide the program with an SDF user interface. SDF supplies the appropriate function calls in support of the functions of the SDF macros RDSTMT, CORSTMT, TRSTMT, CMDRC, CMDSTA and CMD. In addition, SDF offers functions that facilitate the analysis of the standardized transfer area.

For a detailed description of the SDF interface to high-level programming languages please refer to the “SDF-A” manual [4].

i The interface to higher programming languages only supports the old format of the standardized transfer area. The new format can only be used through the Assembler interface!

8 Structured output in S variables

The information provided by SHOW commands is output on the terminal in interactive mode if the SYSOUT system file does not contain a different assignment. Some SHOW commands also allow the information to be output to the SYSLST system file or to a cataloged file. The global information, however, is only prepared for screen output.
Parts of the global information cannot be accessed directly in procedures. Instead, you must save the global information to a file (or S variable list) by redirecting SYSOUT (see Volume 1 of the “Commands“ manual [1], ASSIGN-SYSOUT command). Each screen line is thus equivalent to an unstructured data record (or list element), which can contain both specific information and constant identifiers. Current specific information can only be selected if its output layout is recognized exactly via string processing (e.g. with an EDT procedure or builtin functions).
Because SYSOUT outputs could change in subsequent versions (e.g. provide additional information or have a new layout), these procedures must be continually adjusted.

As of SDF-P V2.0, SHOW commands can output their information in compound S variables of the type ‘structure’. This allows the user to access specific information directly. Every SHOW command with this functionality predefines the layout of the structure:

– A structure is defined for an object specified in the SHOW command (e.g. a file or device). If more than one object is specified (e.g. as a wildcard search pattern), a list of structures is created.

– For each specific item of information on this object, an S variable is defined as an element of this structure and the specific information is assigned to it as the content.

– If information on an object can be further subdivided hierarchically, for each hierarchy a compound S variable is defined as an element of the higher-ranking structure.
A hierarchically lower-ranking S variable can therefore be a simple S variable, a structure or a list of simple S variables and/or structures.

– The names of the elements are preset for the respective SHOW commands. As far as possible, they match the corresponding operand names or a unique abbreviation. Elements containing the same information are given the same name across all SHOW commands.

– As far as possible, the contents of the S variables (the specific information) match corresponding operand values or unique abbreviations.

– The S variables have a defined type: string, integer or Boolean.

As of SDF V4.0, the commands SHOW-SDF-OPTIONS and SHOW-SYNTAX-VERSIONS and the privileged command SHOW-SDF-PARAMETERS offer this functionality. The variable structure can be taken from the respective command descriptions. Further subsystems will also offer this functionality in future versions.

The user can create and evaluate structured system outputs if the chargeable SDF-P subsystem (as of V2.0) is loaded. The procedure is as follows:

1. Declare S variable

   The user declares a compound S variable of the type ‘structure’, where the structure is defined as an element in a list. Any name that complies with the SDF-P naming conventions can be chosen. The structure should be dynamically expandable (default).

   ```
   /DECLARE-VARIABLE NAME=USERVAR(TYPE=*STRUCTURE),MULTIPLE-ELEMENTS=*LIST
   ```

   If, on the other hand, the structure is only statically created, the user can only receive specific information for which he/she has also declared structure elements explicitly with the respective defined names (see the “SDF-P“ manual [6]).

2. Create structured output

   For structured output of individual commands, the user calls a SHOW command via EXECUTE-CMD and specifies that the structured system output is to be routed into the declared structure.

   ```
   /EXECUTE-CMD CMD =( SHOW-SYNTAX-VERSIONS SOFTWARE-UNIT-NAME=(JV, LMS) ), -
                STRUCTURED-OUTPUT = USERVAR ,TEXT-OUTPUT = *NONE
   ```

   The output to SYSOUT is here suppressed with TEXT-OUTPUT=*NONE.

   Instead of EXECUTE-CMD, the user can also assign the system output stream for structured system outputs SYSINF to the declared structure via the ASSIGN-STREAM command. During the assignment, however, the S variable is extended accordingly for each command issued that supports structured system output. With EXECUTE-CMD, on the other hand, output of the specified command is explicitly requested.
   For information on SYSINF, see the “SDF-P“ manual [6].

3. Output contents of the S variable

   `/SHOW-VARIABLE  USERVAR`

```
/show-var uservar
USERVAR(*LIST).F-NAME = :2OSH:$TSOS.SYSSDF.JV.130
USERVAR(*LIST).TYPE = *SYS
USERVAR(*LIST).SW-UNIT(*LIST).NAME = JV
USERVAR(*LIST).SW-UNIT(*LIST).VERSION = 13.0C800
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).NAME = JVS                    (1)
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).VERSION = 320      (4)   (3)
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).NAME = CJC
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).VERSION = 200      (5)
USERVAR(*LIST).F-NAME = :2OSH:$TSOS.SYSSDF.LMS.033
USERVAR(*LIST).TYPE = *SYS
USERVAR(*LIST).SW-UNIT(*LIST).NAME = LMS
USERVAR(*LIST).SW-UNIT(*LIST).VERSION = 03.3A200                              (2)
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).NAME = LMS
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).VERSION = 033
/
```

Explanation of the output:

The S variable defined by the user (here `USERVAR`) contains the global output. The string “(*LIST)” specifies that the S variable can contain several elements in a list. The S variable `USERVAR` contains 2 elements:
Firstly the structure for information on the software unit JV, numbered (1), and secondly the structure for information on the software unit LMS, numbered (2).
The information for a software unit consists of the elements F-NAME, TYPE and SW-UNIT, where SW-UNIT itself can contain a further list. In the case of the software unit JV, SW-UNIT contains a list element, namely the structure which is formed from the elements NAME, VERSION and COMPONENT (numbered (3)).
COMPONENT, on the other hand, can be a list: the software unit JV consists of 2 components, i.e. COMPONENT contains 2 list elements. Each list element is a structure with the elements NAME and VERSION. In the output the components CJC and JVS are numbered (4) and (5) respectively.

4. Access specific information

   Specific information can be accessed via the names of the S variables. The name to be entered is composed as follows:

   uservar#i.element

   where the components have the following meanings:

uservar	Name of the structure that was declared by the user
#i	i-th element in the list For i=1, “i” can be omitted, i.e. only “#” is specified.
Period	Delimiter in names of compound S variables.
element	Predefined name of the structure element. element can, however, be a compound variable: e.g. uservar#.SW-UNIT#.NAME

   The contents can be displayed, e.g. with ”SHOW-VARIABLE uservar#.element“, or via variable substitution:

```
/show-var (uservar#.sw-unit#.component#1.name,uservar#.sw-unit#.component#2.name)
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).NAME = CJC
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).NAME = JVS

/write-text '*** SW-UNIT &(uservar#2.sw-unit#.name) -
/mit der Version &(uservar#2.sw-unit#.version) ***'
*** SW-UNIT LMS mit der Version 03.3A200 ***

/show-var uservar#2
USERVAR(*LIST).F-NAME = :2OSH:$TSOS.SYSSDF.LMS.033
USERVAR(*LIST).TYPE = *SYS
USERVAR(*LIST).SW-UNIT(*LIST).NAME = LMS
USERVAR(*LIST).SW-UNIT(*LIST).VERSION = 03.3A200
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).NAME = LMS
USERVAR(*LIST).SW-UNIT(*LIST).COMPONENT(*LIST).VERSION = 033
```

Examples

Save current SDF options for subsequent resetting

Contents of the procedure file PROC.RESET-SDF-OPT:

```
   /DECL-PAR  OLD-SDF-OPT ( INIT=*PROMPT )
/            "Prompt S variable with SDF options"
/WRITE-TEXT  ´Take over task-specific SDF options from S variable´
/WRITE-TEXT  ´Enter name of S variable: ´
/REMARK       &(OLD-SDF-OPT)
/            "Does OLD-SDF-OPT contain the name of a user syntax file? "
/
/IMPORT-VAR   &(OLD-SDF-OPT)
/SHOW-VAR  &(OLD-SDF-OPT)
/DECL-VAR  SDF-OPT (TYPE=*STRUCTURE(DEF=*DYN)),MULT-ELEM=*LIST, -
/          CONTAINER = *VAR( &(OLD-SDF-OPT) )
/
/SF-USER: IF    ( IS-DECLARED(´SDF-OPT#.SF´))
/             "information available on syntax files"
/     "deactivate currently active user syntax file:"
/     MOD-SDF-OPT  SYNTAX-FILE = *NONE
/     DECL-VAR  SF-VAR(TYPE=*STRUC) "Define control variable"
/     FOR1: FOR  SF-VAR = *LIST( SDF-OPT#.SF)
/       IF  ( SF-VAR.TYPE = ´*USER´ )
/          IF  ( SF-VAR.F-NAME = ´´ )
/             "no user syntax file was activated"
/             USER-SF = ´*NONE´
/             CYCLE  FOR1   "terminate FOR loop"
/          ELSE
/             "at least one user syntax file was activated"
/             MOD-SDF-OPT  SYNTAX-FILE = *ADD( &(SF-VAR.F-NAME) )
/          END-IF
/       END-IF
/          "System or group syntax file! no action"
/          "the SDF options were all saved with INF=*ALL!"
/     END-FOR
/ELSE    "No information on syntax files"
/        "so no user syntax file was activated"
/       MOD-SDF-OPT  SYNTAX-FILE = *NONE
/END-IF
/
/           "Set the saved SDF options"
/MOD-SDF-OPT  SYNTAX-FILE = *UNCHANGED  -
/            ,GUIDANCE         = &(SDF-OPT#.GUIDE)  -
/            ,LOGGING          = &(SDF-OPT#.LOG)  -
/            ,UTILITY-INTERFACE= &(SDF-OPT#.UTILITY-INTERF) -
/            ,PROCEDURE-DIALOG = &(SDF-OPT#.PROC-DIALOG) -
/           ,CONTINUATION     = &(SDF-OPT#.CONTI) -
/           ,MENU-LOGGING     = &(SDF-OPT#.MENU-LOG) -
/           ,MODE             = &(SDF-OPT#.MODE) -
/           ,TEST-PROGRAM-NAME= &(SDF-OPT#.TEST-PROG-NAME) -
/           ,FUNCTION-KEYS    = &(SDF-OPT#.FUNC-KEY) -
/           ,INPUT-HISTORY    = &(SDF-OPT#.INPUT-HIST)
/WRITE-TEXT  ´SDF options reset!´
/SHOW-SDF-OPT  INF = *USER
```

The user declares the S variable SDF-OPT-1 and saves the current SDF options in it. To enable access to this S variable from a procedure, SCOPE=*TASK must also be defined in the declaration:

```
/decl-var sdf-opt-1( type=*struc ), mult-elem=*list, scope=*task

/exec-cmd (show-sdf-opt inf=*user),struct-out=sdf-opt-1,text-out=*none

/show-var sdf-opt-1
SDF-OPT-1(*LIST).SF(*LIST).F-NAME = :2OSG:$USER1.SYSSDF.USER.SPECIAL.01
SDF-OPT-1(*LIST).SF(*LIST).TYPE = *USER
SDF-OPT-1(*LIST).SF(*LIST).VERSION = UNDEFINED
SDF-OPT-1(*LIST).SF(*LIST).F-NAME = :2OSG:$USER1.SYSSDF.USER.SPECIAL.02
SDF-OPT-1(*LIST).SF(*LIST).TYPE = *USER
SDF-OPT-1(*LIST).SF(*LIST).VERSION = UNDEFINED
SDF-OPT-1(*LIST).GUIDE = *EXPERT
SDF-OPT-1(*LIST).LOG = *INPUT-FORM
SDF-OPT-1(*LIST).CONTI = *NEW-MODE
SDF-OPT-1(*LIST).UTILITY-INTERF = *NEW-MODE
SDF-OPT-1(*LIST).PROC-DIALOG = *NO
SDF-OPT-1(*LIST).MENU-LOG = *NO
SDF-OPT-1(*LIST).CMD-STATIS =
SDF-OPT-1(*LIST).MODE = *EXEC
SDF-OPT-1(*LIST).CHECK-PRIV = *YES
SDF-OPT-1(*LIST).TEST-PROG-NAME = *NONE
SDF-OPT-1(*LIST).FUNC-KEY = *STYLE-GUIDE-MODE
SDF-OPT-1(*LIST).INPUT-HIST = *ON
SDF-OPT-1(*LIST).NUM-OF-INPUT = 20
SDF-OPT-1(*LIST).PASSWORD-PROT = *YES
/...
```

During further processing, some of the SDF options are changed (e.g. the user syntax files are deactivated and a different one activated). The original settings can be restored by calling the PROC.RESET-SDF-OPT procedure:

```
/show-sdf-opt inf=*user
%  USER      : :2OSG:$USER1.SYSSDF.USER.SPECIAL.03
%              VERSION : UNDEFINED
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : LMS
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 99
%     PASSWORD-PROTECTION: *YES
```

```
/call-proc proc.reset-sdf-opt,log=*yes
```

Tracer listing:

```
/call-proc proc.reset-sdf-opt,log=*yes
%          1  1 /DECL-PAR  OLD-SDF-OPT ( INIT=*PROMPT )
%          4  1 /WRITE-TEXT 'Take over task-specific SDF options from S variable'
Take over task-specific SDF options from S variable
%          5  1 /WRITE-TEXT  'nter name of S variable: '
nter name of S variable::
%OLD-SDF-OPT: sdf-opt-1
%OLD-SDF-OPT: sdf-opt-1
%          6  1 /REMARK      SDF-OPT-1
%         12  1 /IMPORT-VAR   SDF-OPT-1
%         13  1 /SHOW-VAR  SDF-OPT-1
SDF-OPT-1(*LIST).SF(*LIST).F-NAME = :2OSG:$USER1.SYSSDF.USER.SPECIAL.01
SDF-OPT-1(*LIST).SF(*LIST).TYPE = *USER
SDF-OPT-1(*LIST).SF(*LIST).VERSION = UNDEFINED
SDF-OPT-1(*LIST).SF(*LIST).F-NAME = :2OSG:$USER1.SYSSDF.USER.SPECIAL.02
SDF-OPT-1(*LIST).SF(*LIST).TYPE = *USER
SDF-OPT-1(*LIST).SF(*LIST).VERSION = UNDEFINED
SDF-OPT-1(*LIST).GUIDE = *EXPERT
SDF-OPT-1(*LIST).LOG = *INPUT-FORM
SDF-OPT-1(*LIST).CONTI = *NEW-MODE
SDF-OPT-1(*LIST).UTILITY-INTERF = *NEW-MODE
SDF-OPT-1(*LIST).PROC-DIALOG = *NO
SDF-OPT-1(*LIST).MENU-LOG = *NO
SDF-OPT-1(*LIST).CMD-STATIS =
SDF-OPT-1(*LIST).MODE = *EXEC
SDF-OPT-1(*LIST).CHECK-PRIV = *YES
SDF-OPT-1(*LIST).TEST-PROG-NAME = *NONE
SDF-OPT-1(*LIST).FUNC-KEY = *STYLE-GUIDE-MODE
SDF-OPT-1(*LIST).INPUT-HIST = *ON
SDF-OPT-1(*LIST).NUM-OF-INPUT = 20
```

```
SDF-OPT-1(*LIST).PASSWORD-PROT = *YES
%          14  1 /DECL-VAR  SDF-OPT (TYPE=*STRUCTURE(DEF=*DYN)),MULT-ELEM=*LIST,
           CONTAINER = *VAR( SDF-OPT-1 )
%          17  1 /SF-USER:
%          17  1 / IF    ( IS-DECLARED('SDF-OPT#.SF'))
%          20  1 /MOD-SDF-OPT  SYNTAX-FILE = *NONE
%          21  1 /DECL-VAR  SF-VAR(TYPE=*STRUC) "Laufvariable definieren"
%          22  1 /FOR1:
%          22  1 / FOR  SF-VAR = *LIST( SDF-OPT#.SF)
%          23  1 /IF  ( SF-VAR.TYPE = '*USER' )
%          24  1 /IF  ( SF-VAR.F-NAME = '' )
%          28  1 /ELSE
%          30  1 /MOD-SDF-OPT  SYNTAX-FILE = *ADD( :2OSG:$USER1.SYSSDF.USER.SPECI
AL.01 )
%          31  1 /END-IF
%          32  1 /END-IF
%          35  1 /END-FOR
%          23  1 /IF  ( SF-VAR.TYPE = '*USER' )
%          24  1 /IF  ( SF-VAR.F-NAME = '' )
%          28  1 /ELSE
%          30  1 /MOD-SDF-OPT  SYNTAX-FILE = *ADD( :2OSG:$USER1.SYSSDF.USER.SPECI
AL.02 )
%          31  1 /END-IF
%          32  1 /END-IF
%          35  1 /END-FOR
%          39  1 /END-IF
%          42  1 /MOD-SDF-OPT  SYNTAX-FILE = *UNCHANGED               ,GUIDANCE
        = *EXPERT              ,LOGGING          = *INPUT-FORM              ,UTILI
TY-INTERFACE= *NEW-MODE             ,PROCEDURE-DIALOG = *NO             ,CONTINU
ATION     = *NEW-MODE             ,MENU-LOGGING     = *NO             ,MODE
        = *EXEC             ,TEST-PROGRAM-NAME= *NONE             ,FUNCTION-KEYS
    = *OLD-MODE             ,INPUT-HISTORY    = *ON
%          53  1 /WRITE-TEXT  'SDF options reset!'
SDF options reset!
%          54  1 /SHOW-SDF-OPT  INF = *USER
%  USER      : :2OSG:$USER1.SYSSDF.USER.SPECIAL.01
%              VERSION : UNDEFINED
%  USER      : :2OSG:$USER1.SYSSDF.USER.SPECIAL.02
%              VERSION : UNDEFINED
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 99
%     PASSWORD-PROTECTION: *YES
%              1 /EXIT-PROCEDURE ERROR=*NO
/
```

Examples of access to different list elements

```
/decl-var syn1(type=*struct),mult-elem=*list
```

```
/exec-cmd (show-syntax-versions (archive,acs)),struct-out=syn1,text-out=*none
```

Display of the global output generated:

```
/show-var syn1,inf=*par(list-index-number=*yes)
SYN1#1.F-NAME = :2OSH:$TSOS.SYSSDF.ARCHIVE.060
SYN1#1.TYPE = *SYS
SYN1#1.SW-UNIT#1.NAME = ARCHIVE
SYN1#1.SW-UNIT#1.VERSION = 06.0P600
SYN1#1.SW-UNIT#1.COMPONENT#1.NAME = ARC
SYN1#1.SW-UNIT#1.COMPONENT#1.VERSION = 560
SYN1#2.F-NAME = :2OSH:$TSOS.SYSSDF.ACS.140
SYN1#2.TYPE = *SYS
SYN1#2.SW-UNIT#1.NAME = ACS
SYN1#2.SW-UNIT#1.VERSION = 14.0B100
SYN1#2.SW-UNIT#1.COMPONENT#1.NAME = ACS
SYN1#2.SW-UNIT#1.COMPONENT#1.VERSION = 016
SYN1#3.F-NAME = :2OSH:$TSOS.SYSSDF.ACS.140
SYN1#3.TYPE = *SYS
SYN1#3.SW-UNIT#1.NAME = ACS
SYN1#3.SW-UNIT#1.VERSION = 14.0B100
SYN1#3.SW-UNIT#1.COMPONENT#1.NAME = ACS
SYN1#3.SW-UNIT#1.COMPONENT#1.VERSION = 016
```

Display of the second list element:

```
/show-var syn1#2,inf=*par(list-index-number=*yes)
SYN1#2.F-NAME = :2OSH:$TSOS.SYSSDF.ACS.140
SYN1#2.TYPE = *SYS
SYN1#2.SW-UNIT#1.NAME = ACS
SYN1#2.SW-UNIT#1.VERSION = 14.0B100
SYN1#2.SW-UNIT#1.COMPONENT#1.NAME = ACS
SYN1#2.SW-UNIT#1.COMPONENT#1.VERSION = 016
```

Display of the first component name with the version for the second list element:

```
/show-var syn1#2.sw-unit#.component#
```

```
/show-var syn1#2.SW-UNIT#.COMPONENT#,inf=*par(list-index-number=*yes)
SYN1#2.SW-UNIT#1.COMPONENT#1.NAME = ACS
SYN1#2.SW-UNIT#1.COMPONENT#1.VERSION = 016
```

Example of a static structure

The user only wants to store the current setting of the guidance mode in a variable:

```
/decl-var sdf-opt-2(type=*struct(def=*by-syscmd)),mult-elem=*list

/beg-struc

/decl-elem guide

/end-struc

/exec-cmd (show-sdf-opt inf=*user),struct-out=sdf-opt-2,text-out=*none

/show-var sdf-opt-2

SDF-OPT-2(*LIST).GUIDE = *EXPERT
```

Only an explicitly declared variable is initialized. The disadvantage of this is that the user must define the desired structure element explicitly. This element can also be accessed directly with the dynamic structure.

9 Notes on OMNIS and CHECKPOINT/RESTART

OMNIS

In the case of jobs initiated via OMNIS, GUIDANCE=*EXPERT is always set after logon, regardless of any conflicting definition in the global information of the activated syntax files. When activating or switching the user syntax file, GUIDANCE=*EXPERT is likewise set. The setting FUNCTION-KEYS=*STYLE-GUIDE-MODE in the global informations is activated only for a 9763 Terminal.
However, it is possible to switch to another dialog form or to change the function key assignment with the MODIFY-SDF-OPTIONS command or statement.

CHECKPOINT/RESTART

After RESTART-PROGRAM, the following syntax file environment is restored:

- system and group syntax files as currently assigned to the task
- user syntax file as assigned when the checkpoint was written
- program-specific syntax file environment which was open when the checkpoint was written, i.e. all syntax files explicitly opened by a program via OPNCALL.

CHECKPOINT/RESTART cannot be used in the following cases:

- The version of the SDF subsystem is different from the version used when the checkpoint was written.
- The system or group syntax file has been incompatibly changed since the checkpoint was written.

Figures

Tables

Related publications

Please apply to your local office for ordering the manuals.

[1] **BS2000/OSD-BC V5.0**
Commands, Volumes 1 - 5
User Guide

Target group
This manual is addressed to nonprivileged users and systems support staff.
Contents
Volumes 1 through 5 contain the BS2000/OSD commands ADD-... to WRITE-... (basic configuration and selected products) with the functionality for all privileges. The command and operand functions are described in detail, supported by examples to aid understanding. An introductory overview provides information on all the commands described in Volumes 1 through 5.
The Appendix of Volume 1 includes information on command input, conditional job variable expressions, system files, job switches, and device and volume types.
The Appendix of Volumes 4 and 5 contains an overview of the output columns of the SHOW commands of the component NDM. The Appendix of Volume 5 contains additionally an overview of all START commands.
There is a comprehensive index covering all entries for Volumes 1 through 5.
Order numbers

U2338-J-Z125-15-76	Commands, Volume 1, A – C
U41074-J-Z125-2-76	Commands, Volume 2, D – MOD-JO
U21070-J-Z125-5-76	Commands, Volume 3, MOD-JV – R
U41075-J-Z125-2-76	Commands, Volume 4, S – SH-PRI
U23164-J-Z125-4-76	Commands, Volume 5, SH-PUB – Z

[2] **BS2000/OSD-BC V5.0**
Commands, Volume 6, Output in S Variables and SDF-P-BASYS
User Guide

Target group
This manual is addressed to programmers and users who write procedures.
Contents
Volume 6 contains tables of all S variables that are supplied with values by the SHOW commands in conjunction with structured output. Further chapters deal with:
– introduction to working with S variables
– SDF-P-BASYS V2.2A
Order number
U23165-J-Z125-4-76

[3] **BS2000/OSD-BC V5.0**
Introductory Guide to Systems Support
User Guide

Target group
This manual is addressed to BS2000/OSD systems support staff and operators.
Contents
The manual covers the following topics relating to the management and monitoring of the BS2000/OSD basic configuration: system initialization, parameter service, job and task control, memory/device/system time/user/file/pubset management, assignment of privileges, accounting and operator functions.
Order number
U2417-J-Z125-14-76

[4] **SDF-A V4.1E** (BS2000/OSD)
User Guide

Target group
This manual is intended for experienced BS2000 users and system administration staff.
Contents
It describes how to process syntax files and explains the SDF-A functions on the basis of examples. The SDF-A statements are listed in alphabetical order.
The manual also includes a description of the SDF-SIM utility routine.
Bestellnummer
U2284-J-Z125-9-76

[5] **SDF V4.5A** (BS2000/OSD)
SDF Management
User Guide

Target group
This manual is intended for system administrators and experienced BS2000 users.
Contents
It describes how SDF is installed and administered using SDF commands and the SDF-I, SDF-U and SDF-PAR utility routines. It includes a description of SDF-I, SDF-U and SDF-PAR statements.
Order number
U2622-J-Z125-10-76

[6] **SDF-P V2.2A** (BS2000/OSD)
Programming in the Command Language
User Guide

Target group
This manual is addressed to BS2000 users and systems support staff.
Contents
SDF-P is a structured procedure language in BS2000. The manual begins with introductory chapters dealing with the basic principles of procedures and variables, and goes on to provide detailed descriptions of SDF-P commands, functions and macros.
Overview of contents:
– brief introduction to SDF-P
– procedure concept in SDF-P
– creating, testing, calling and controlling S procedures
– S variables, S variable streams, functions, expressions
– converting non-S procedures
– macros, predefined (built-in) functions, SDF-P commands

SDF-P V2.2A can only be used in conjunction with SDF-P-BASYS ≥ V2.1A, VAS ≥ V2.0A and SDF ≥ V4.1A.
Order number
U6442-J-Z125-5-76

[7] **SDF-CONV V3.0A** (BS2000/OSD)
User Guide

Target group
This manual is addressed to all BS2000 users.
Contents
The procedure format and command language of procedures can be converted as follows:
– from ISP to SDF command language
– from non-S to S procedure format
– simultaneous conversion of command language and procedure format.
The complete functionality of SDF-CONV V3.0A is available as of BS2000/OSD V1.0 and SDF V4.0A.
There is a README file for SDF-CONV V3.0B.
Order number
U6540-J-Z125-3-76

[8] **POSIX (BS2000/OSD)**
POSIX Basics for Users and System Administrators
User Guide

Target group
BS2000 system administrators, POSIX administrators, BS2000 users, users of UNIX/SINIX workstations.
Contents
This manual describes the following: introduction to and working with POSIX; BS2000 software products in a POSIX environment; installing, controlling and exiting the POSIX subsystem; managing POSIX users via BS2000.
Order number
U22795-J-Z125-2

[9] **POSIX (BS2000/OSD)**
Commands
User Guide

Target grou
The manual addresses all users of the POSIX shell.
Contents
The manual is designed as a work of reference. It describes working with the POSIX shell and the commands of the POSIX shell in alphabetical order.
Order number
U22794-J-Z125-3

[10] **SECOS V4.0A** (BS2000/OSD
Security Control System
User Guide

Target group
- BS2000 system administrators
- BS2000 users working with extended access protection for files

Contents
Capabilities and application of the functional units:
- SRPM (System Resources and Privileges Management)
- SRPMSSO (Single Sign On)
- GUARDS (Generally Usable Access Control Administration System)
- GUARDDEF (Default Protection)
- GUARDCOO (Co-owner Protection)
- SAT (Security Audit Trail).

Order number
U5605-J-Z125-6-76

[11] **XHCS V1.3** (BS2000/OSD)
8-Bit Code Processing in BS2000/OSD
User Guide

Target group
Users of the DCAM, TIAM and UTM access methods, system administrators, and users migrating from EHCS to XHCS.
Contents
XHCS (Extended Host Code Support) is a software package of BS2000/OSD that lets you use extended character sets in conjunction with 8-bit terminals. XHCS is also the central source of information on the coded character sets in BS2000/OSD. XHCS replaces EHCS.
Order number
U9232-J-Z135-4-76
Additional functions see README file for XHCS V1.4

[12] **BS2000/OSD**
Softbooks English
CD-ROM

Target group
BS2000/OSD users
Contents
The CD-ROM "BS2000/OSD SoftBooks English" contains almost all of the English manuals and README files for the BS2000 system software of the latest BS2000/OSD version and also of the previous versions, including the manuals listed here.
These Softbooks can also be found in the Internet on our manual server. You can browse in any of these manuals or download the entire manual.
Order number
U26175-J8-Z125-1-76
Internet address
http://manuals.fujitsu-siemens.com

Index

Contents

SDF V4.5A (BS2000/OSD)

Introductory Guide to the SDF Dialog Interface

Target group
BS2000/OSD users

Contents
This manual describes the interactive input of commands and statements in SDF format. A Getting Started chapter with easy-to-understand examples and further comprehensive examples facilitates use of SDF. SDF syntax files are discussed.

Edition: March 2002

File: sdf_einf.pdf

This manual was produced by
cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Fujitsu Siemens computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00000

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on SDF V4.5A
Introductory Guide to the SDF Dialog Interface