

SDF V4.7

SDF Dialog Interface

Valid for

SDF-I V4.1

SDF-U V4.1

SDF-PAR V1.1

SDF-CONV V3.0B

DISPLAY V1.1

BS2000/OSD V8.0

JV V15.0

SDF-P V2.5A

Comments... Suggestions... Corrections...

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:

manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2008.

cognitas. Gesellschaft für Technik-Dokumentation mbH

www.cognitas.de

Copyright and Trademarks

Copyright © Fujitsu Technology Solutions GmbH 2010.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Contents

1	Preface	11
1.1	Brief product description	11
1.2	Target group	13
1.3	Summary of contents	13
1.4	Version compatibility	14
1.5	README file	15
2	Brief introduction to working with SDF	17
3	Input of commands and statements	33
3.1	Input in unguided dialog	33
3.1.1	EXPERT form	34
3.1.2	NO form	34
3.1.3	Special entries	35
3.1.4	Function keys	36
3.1.5	Blocked input	37
3.2	Input in guided dialog	39
3.2.1	Levels of user guidance	39
	Maximum user guidance	39
	Medium user guidance	40
	Minimum user guidance	40
3.2.2	Screen format	41
3.2.3	Special entries	43
3.2.4	Input in the NEXT line	45
3.2.5	Function keys	47
3.2.6	Effects of the quit functions	50
3.2.7	Application domain menu	52

3.2.8	Command/statement menu	53
3.2.9	Operand form	54
3.2.10	Special entries in the operand form	56
3.2.11	Positioning within operand forms	57
3.3	Input in temporarily guided dialog	58
3.3.1	Effects of the quit functions	60
3.4	Further input options	62
3.4.1	Input from procedures	62
3.4.2	Input in batch mode	69
3.5	Replacing expressions in the input	70
3.6	Input compression	77
3.6.1	Abbreviation of names	77
3.6.2	Default values	79
3.6.3	Positional operands	80
3.6.4	Structures	81
3.7	Logging	83
3.7.1	LOGGING parameter setting	83
3.7.2	MENU-LOGGING parameter setting	84
3.7.3	Restrictions	84
3.8	Selection menu for commands and statements	85
3.8.1	Wildcards in command or statement names	85
3.8.2	Ambiguous abbreviation of command and statement names	86
3.9	Reusing previous inputs	87
3.10	Test mode	89
3.11	Task-specific default values	93
4	Examples of command input in interactive mode	99
4.1	HELP-SDF	100
4.2	Unguided dialog	104
4.2.1	EXPERT form	104
4.2.2	NO form	106
4.3	Temporarily guided dialog	108
4.4	Guided dialog	120
4.4.1	Minimum user guidance	120
4.4.2	Medium user guidance	125

4.4.3	Maximum user guidance	129
5	SDF syntax files	137
5.1	System syntax file	138
5.2	Group syntax file	139
5.3	User syntax file	140
5.4	Privileges in syntax files	141
5.5	File hierarchy	141
5.6	Processing syntax files	143
6	SDF user interface	145
6.1	Commands of the SDF domain	145
6.2	Standard statements	146
	END	146
	EXECUTE-SYSTEM-CMD	147
	HELP-MSG-INFORMATION	148
	HOLD-PROGRAM	148
	MODIFY-SDF-OPTIONS	149
	REMARK	149
	RESET-INPUT-DEFAULTS	150
	RESTORE-SDF-INPUT	150
	SHOW-INPUT-DEFAULTS	151
	SHOW-INPUT-HISTORY	151
	SHOW-SDF-OPTIONS	152
	SHOW-STMT	152
	STEP	153
	WRITE-TEXT	153
6.3	SDF macros	154
6.4	SDF interface to high-level programming languages	155

7	Notes on OMNIS and CHECKPOINT/RESTART	157
8	Installation of SDF	159
8.1	SDF installation files	159
8.2	Generating the subsystem catalog	161
8.3	Preparing SDF	161
8.3.1	Preparing files for SDF	161
8.3.2	Starting SDF	161
9	Behavior in the event of errors	163
10	SDF-I	165
10.1	SDF-I inputs/outputs	165
10.2	Calling and executing SDF-I	166
10.3	Handling work files	168
10.4	Handling SDF standard statements and domains	169
10.5	Notes	169
10.6	SDF-I statements	171
10.6.1	Overview of functions	171
10.6.2	Description of the statements	172
	CONVERT-SYNTAX-FILE	
	Convert syntax file to new format	172
	END	
	Terminate SDF-I	173
	MERGE	
	Merge syntax files	174
	OPEN	
	Assign input and output syntax files	178
	REMOVE	
	Unmerge syntax files	181
	SHOW-SYNTAX-FILE	
	Output information on syntax file	183

11	SDF-U	187
11.1	SDF-U statements	189
11.1.1	Overview of functions	189
11.1.2	Description of the statements	191
	COPY	
	Copy contents of syntax file	191
	EDIT	
	Set current position to command in syntax file	194
	END	
	Terminate program execution	196
	MODIFY-CMD	
	Modify command definition	197
	MODIFY-VALUE	
	Modify operand value definition	198
	OPEN-SYNTAX-FILE	
	Open syntax file	199
	REMOVE	
	Delete objects from syntax file	200
	SET-GLOBALS	
	Modify global information	202
	SHOW	
	Output objects of syntax file	208
	SHOW-CORRECTION-INFORMATION	
	Output syntax file correction information	213
	SHOW-STATUS	
	Display status of opened syntax file	214
	STEP	
	Define checkpoint	215
12	SDF-PAR	217
12.1	SDF-PAR statements	218
12.1.1	Overview of functions	218
12.1.2	Description of the statements	220
	MODIFY-SYNTAX-FILE	
	Modify entries for syntax files	220
	MODIFY-SYSTEM-LOGON-PROCEDURE	
	Modify entry for global LOGON call procedure	222
	MODIFY-SYSTEM-LOGON-INCLUDE	
	Modify entry for global LOGON include procedure	223

	MODIFY-VERSION	
	Modify format of parameter file	224
	OPEN-PARAMETER-FILE	
	Create or open parameter file	225
	SHOW-PARAMETER-FILE	
	Display parameter file	228
13	SDF-CONV	229
13.1	Starting and terminating the program	230
13.2	Statements	231
13.2.1	Overview of all statements	231
13.2.2	CONVERT statement	232
13.2.3	Control functions	245
13.3	Outputs	246
13.3.1	Output procedure	246
13.3.2	Output to SYSOUT	248
13.3.3	Conversion log to SYSLST	248
13.3.4	Program monitoring using monitoring job variables	249
13.4	Restrictions	250
13.4.1	Basic preconditions and restrictions	250
13.4.2	Restrictions on the conversion into S procedures	252
13.4.3	Commands that cannot be converted	254
13.4.4	Non-convertible operand configurations and conditional conversions	256
13.4.5	Restrictions on procedure parameters as operands	260
13.5	Components and installation of SDF-CONV	263
13.6	Examples	265
	Example 1:	266
	Example 2:	271
	Example 3	279
13.7	Assignment of ISP to SDF commands	282
13.7.1	User commands	282
13.7.2	System administration commands	287
13.8	Conversion of ISP and old SDF commands by SDF-CONV	291
13.8.1	User commands	291
13.8.2	System administration commands	296
13.9	SDF-CONV messages	298

14	DISPLAY	301
<hr/>		
14.1	Product use	301
14.1.1	Restrictions	303
14.2	Commands	304
14.2.1	Overview of all commands	304
14.2.2	CREATE-DISPLAY-CMD	305
14.2.3	DELETE-DISPLAY-CMD	307
14.3	Installation	308
	Related publications	309
<hr/>		
	Index	311
<hr/>		

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**administration), 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 [chapter "SDF-CONV" on page 229](#)).

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.

Chapters 1 through 7 and parts of chapters 10 and 13 are relevant for all users. The other chapters are intended for the system administration.

Further SDF capabilities for systems support tasks as well as self-defined commands or the program interface are described in the “SDF-A” manual [4].

1.3 Summary of contents

This manual introduces the BS2000 dialog interface provided by the **System Dialog Facility (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 contains an overview of rules and regulations to be observed when entering commands or statements.
- Chapter 4 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 5 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 6 contains the SDF-specific commands, standard statements and an overview of the program interfaces available to the user.
- Chapter 7 provides notes on the use of OMNIS and CHECKPOINT/RESTART in conjunction with SDF.
- Chapter 8 shows how SDF is installed.
- Chapter 9 provides information on how to behave when errors occur.
- Chapter 10 describes the SDF-I utility routine for merging syntax files.
- Chapter 11 describes the SDF-U utility routine for modifying syntax files.
- Chapter 12 describes the SDF-PAR utility routine for managing SDF parameter files.

- Chapter 13 describes the SDF-CONV utility routine for converting command procedures to the current procedure and command format.
- Chapter 14 describes the DISPLAY subsystem, which permits the full output scope of most of the SHOW commands to be made available to nonprivileged users.

The BS2000 commands, including privileged commands, are described in the “Commands” manual [1].

Details on administering and operating the system are dealt with in the “Introductory Guide to Systems Support” [2].

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 V4.7 can be used with BS2000/OSD-BC V6.0 and higher. The present manual describes the use of SDF V4.7 in a system running BS2000/OSD-BC V8.0. In particular, the operation examples and sample sessions have been generated under BS2000/OSD-BC V8.0.

Contents

The manuals “Introduction to the SDF Dialog Interface”, “SDF Management” and “SDF-CONV” have been combined in this manual.

The DISPLAY subsystem is also described, which permits the full output scope of most of the SHOW commands to be made available to nonprivileged users.

Command language

The format of the SDF command language together with the SDF syntax representation and the command return codes is described in detail in the “Commands” manual [1] and is consequently no longer contained in this manual.

1.5 README file

Any additions to the manuals are described in the Readme files for the various product versions. These Readme files are available at <http://manuals.ts.fujitsu.com> under the various products.

Readme file under BS2000/OSD

On your BS2000 system you will find Readme files for the installed products under the file name:

`SYSRME.<product>.<version>.E`

Please refer to the appropriate system administrator for the user ID under which the required Readme file can be found. You can also obtain the path name of the Readme file directly by entering the following IMON command:

```
/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>, LOGICAL-ID=SYSRME.E
```

You can view the Readme file with `/SHOW-FILE` or by opening it in an editor or print it at a standard printer using the following command (e.g. SDF V4.7):

```
/PRINT-DOCUMENT FROM-FILE=SYSRME.SDF.047.E, LINE-SPACING=*BY-EBCDIC-CONTROL
```

Additional product informations

Current information, version and hardware dependencies and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available at <http://manuals.ts.fujitsu.com>.

2 Brief introduction to working with SDF

In BS2000/OSD V1.0 and higher the SDF command language (System Dialog Facility) replaced the ISP command language. Commands from the old command language ISP continue to be supported for the sake of compatibility, but the full functionality of SDF can be exploited only with commands and statements employing the SDF syntax (ISP commands are checked only with regard to their names).

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 33](#)) or by switching to a menu-driven input mode (guided dialog, [page 39](#)). 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 87](#)).

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 58](#)).

Outputting application domains

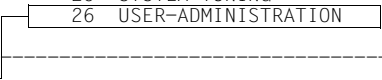
If you enter a question mark and press the transmission key (**[DUE]**), SDF provides a selection of application domains (see the application domain menu on [page 52](#)). For the following example, select the application domain USER-ADMINISTRATION by entering in the NEXT input field (see NEXT line, [page 45](#)) 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 53](#)). Select the MODIFY-USER-PROTECTION command by entering the number 13 in the NEXT input field.

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

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

10  SHOW-SDF-OPTIONS
11  SHOW-SYNTAX-VERSIONS
12  START-SDF-A
13  START-SDF-CONV
14  START-SDF-I
15  START-SDF-SIM
16  START-SDF-U
17  WRITE-TEXT

( ! )

-----
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 DU.

```

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 63](#)). To create the procedure file, call the (editor) program EDT¹. “?” 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-PROGRAM² 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 85](#)). Enter “8” to select the START-EXECUTABLE-PROGRAM command.

```

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

1  CANCEL-PROGRAM                (!) 6  RESUME-PROGRAM                (!)
2  INFORM-PROGRAM
3  LOAD-EXECUTABLE-PROGRAM
4  LOAD-PROGRAM
5  RESTART-PROGRAM
-----
6  RESUME-PROGRAM                (!)
7  SELECT-PROGRAM-VERSION
8  START-EXECUTABLE-PROGRAM
9  START-FOR1-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 **[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 :20SG:$SDFUSER1.ABK.ABKLIST
%      9 :20SG:$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
/

```

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 “?” → “16” (PROCEDURE) → Select CALL-PROCEDURE → 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 **[EFG]** (insert) key and inserting the missing letter “o”. Now send off this corrected command line with **[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  ← Please enter *YES

-----
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 dd-mm-yy FOR TSN '1PYT'
% EXC0421 CPU TIME USED: 0.8174
% JMS0150 INSTALLATION 'S150-40', BS2000 VERSION 'V170', 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 line¹), 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 dd-mm-yy AT 17:14, TSN = 1PY8
*** MINDESTENS 20 TAGE KEIN DATEIZUGRIFF AUF:
%      6 :20SG:$SDFUSER1.ABK.ABKLIST
%      9 :20SG:$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=sdfctest
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 99](#).

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 33](#).

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 137](#).

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 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 70](#)).

Most of the information contained in this chapter can be displayed on the screen by means of the HELP-SDF command (see the "Commands" manual [1]).

3.1 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 [39](#) and [58](#)).

3.1.1 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 58](#)).

3.1.2 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 58](#)). The last input can be redisplayed by means of the `RESTORE-SDF-INPUT` command (see the "Commands" manual [1]).

3.1.3 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 58](#)).
- ?? 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.

LZF key

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

LZE key

enables blocked input (see [section "Blocked input" on page 37](#)). (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 58](#)).

3.1.4 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 1: 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 2: 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 2: 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.

3.1.5 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*4
(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 'nn' FROM LIBRARY ':20SH:$TSOS.SYSLNK.SORT.
nnn' IN PROCESS
% BLS0524 LLM 'SRT80', VERSION 'nnn' OF 'yyyy-mm-dd 15:20:53' LOADED
% BLS0551 COPYRIGHT (C) FUJITSU TECHNOLOGY SOLUTIONS 2009.ALL RIGHTS RESERVED
% SRT1001 18:49:13/000000.00 SORT/MERGE STARTED, VERSION nn.nA00/BS2000Vnn.n
% SRT1130 PLEASE ENTER SORT STATEMENTS
(BL) sort-record
(BL) end
% SRT1016 SORT/MERGE INPUT RECORDS:.....688 (FROM 01)
% SRT1016 SORT/MERGE INPUT RECORDS:.....2.003 (FROM 02)
% SRT1017 RECORDS TO BE SORTED/MERGED:.....2.691
% SRT1030 SORT/MERGE OUTPUT RECORDS:.....2.691
% SRT1002 18:49:14/000000.19 SORT/MERGE COMPLETED
(BL) show-file-attr out.sort1-2
% 66 :20SG:$USER1.OUT.SORT1-2
%:20SG: PUBLIC: 1 FILE RES= 66 FRE= 2 REL= 0 PAGES
(BL) show-file-attr datei.
%PLEASE ACKNOWLEDGE
% 21 :20SG:$USER1.DATEI.1
% 48 :20SG:$USER1.DATEI.2
%:20SG: PUBLIC: 2 FILES RES= 69 FRE= 4 REL= 0 PAGES
(BL) show-file-link link=sort*
%-- LINK-NAME ----- FILE-NAME -----
% SORTOUT :20SG:$USER1.OUT.SORT1-2
% SORT01 :20SG:$USER1.DATEI.1
% SORT02 :20SG:$USER1.DATEI.2
/

```

3.2 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.

3.2.1 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.

3.2.2 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.

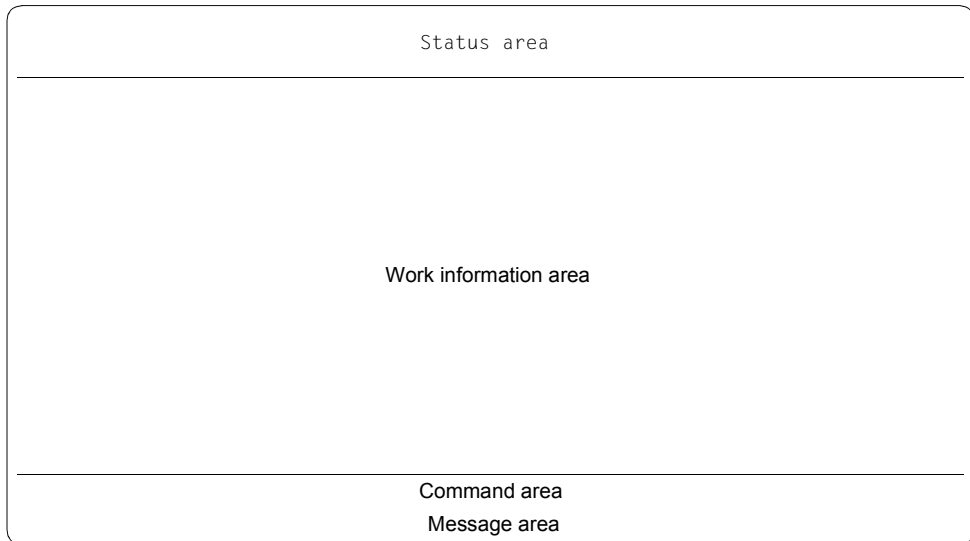


Figure 1: Screen format

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.

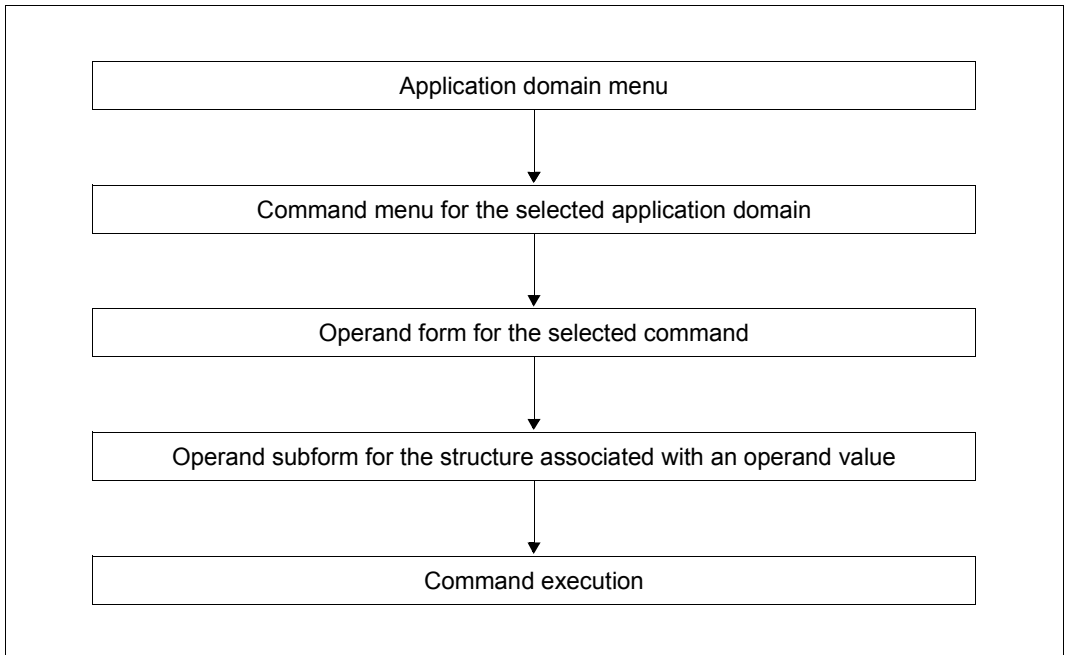


Figure 2: Path from the application domain menu to command execution

3.2.3 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 58](#)).
- ?? 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”.)

3.2.4 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.

3.2.5 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 3: 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 4: 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 4: 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.

3.2.6 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.

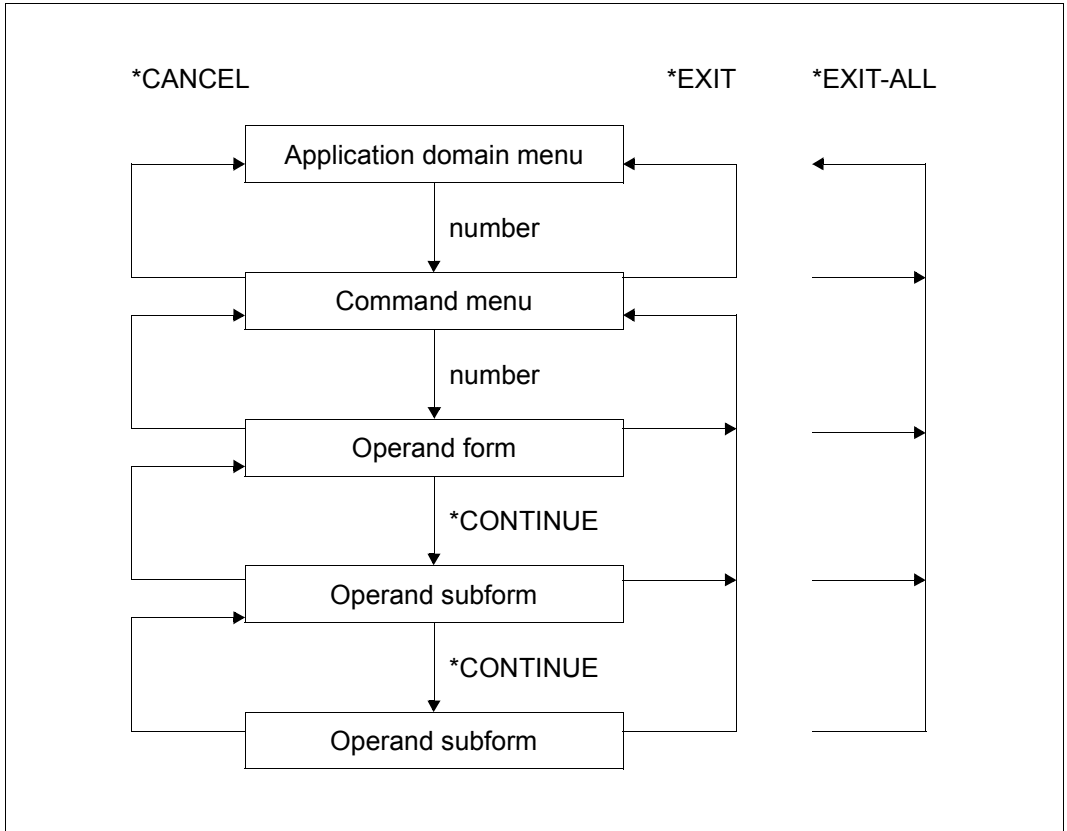


Figure 3: The quit functions *CANCEL, *EXIT and *EXIT-ALL in system mode (guided dialog)

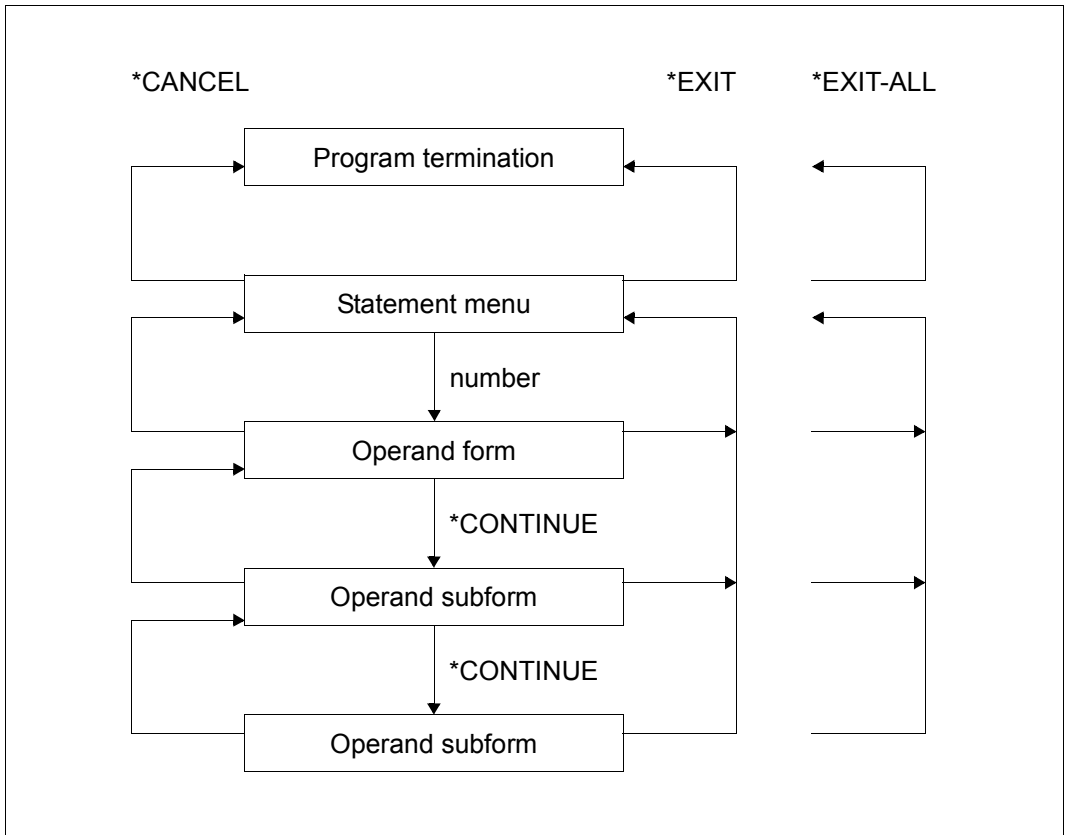


Figure 4: The quit functions *CANCEL, *EXIT and *EXIT-ALL in program mode (guided dialog)

3.2.7 Application domain menu

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

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

The above domains correspond to the status of BS2000/OSD-BC V8.0A. Domains without contents (commands disabled or not implemented) are not displayed.

Comments:

- 1) The application domain contains all commands which can be input in the dialog.
- 2) The application domain will replace MULTI-CATALOG in the long term.
- 3) The application domain will replace RSO-SPOOL-ADMINISTRATION in the long term.
- 4) The application domain will replace SPOOL in the long term.
- 5) The application domain 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.

3.2.8 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

3.2.9 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 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 **F7** key in Styleguide mode)
- after entering *CANCEL in the NEXT line of a subform (or pressing the **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.

3.2.10 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"). |

3.2.11 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.

3.3 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 39](#)). 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).

3.3.1 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.

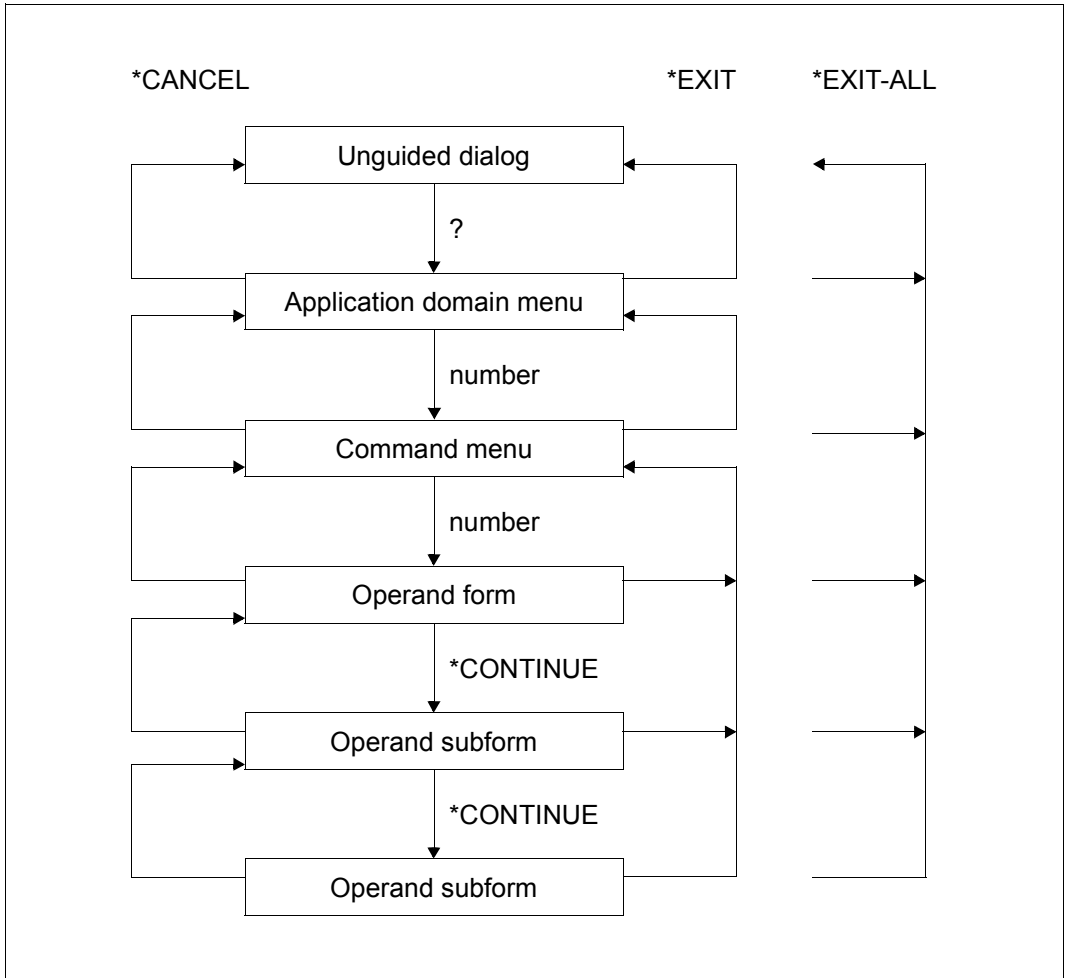


Figure 5: The quit functions *CANCEL, *EXIT and *EXIT-ALL in system mode (temporarily guided dialog)

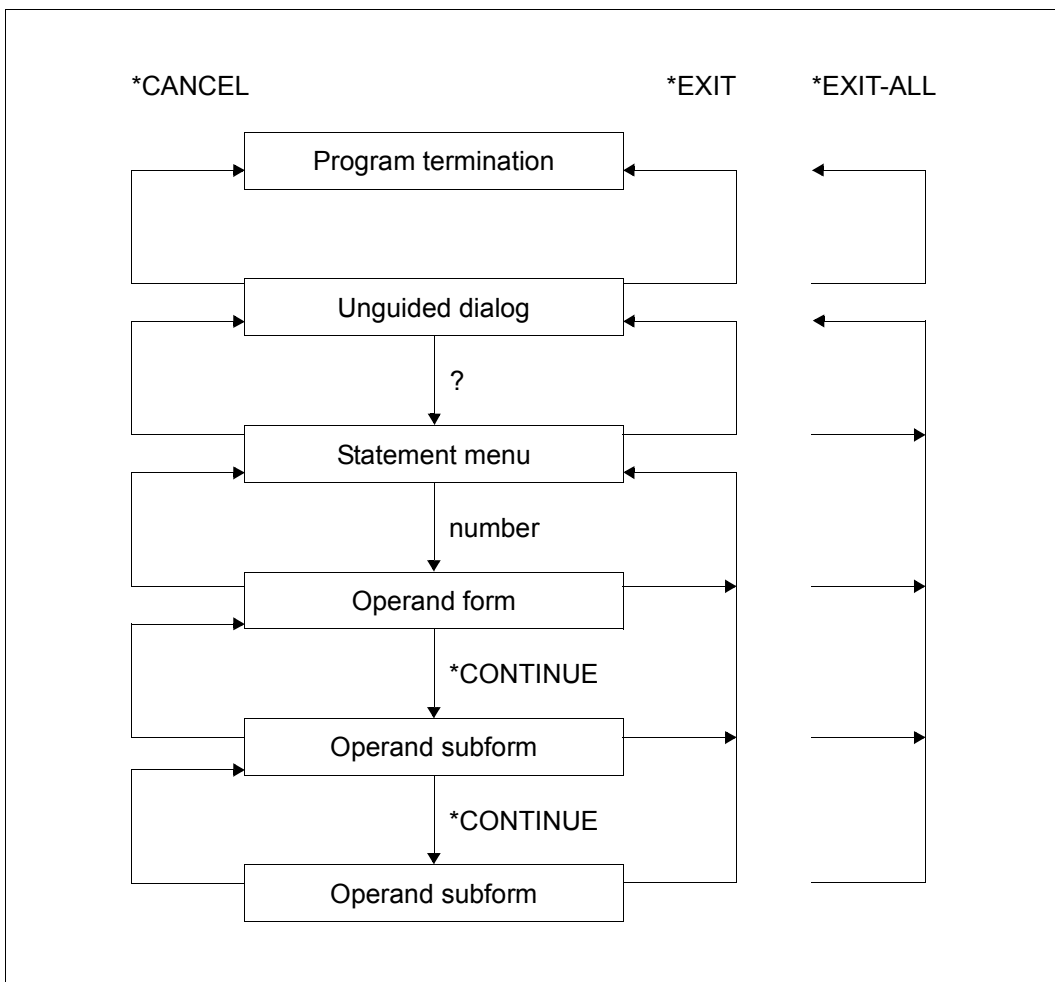


Figure 6: Effects of the functions *CANCEL, *EXIT and *EXIT-ALL in program mode (temporarily guided dialog)

3.4 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.

3.4.1 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 93ff](#).

Implemented procedures

Users can define their own commands in a user syntax file (see pages [137](#) and [143](#)). 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 [5]).

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 [“Administration of system-wide procedure files” on page 68](#)).

A system LOGON procedure is not started for tasks in which a group syntax file is assigned without evaluating the system syntax file.

If the user ID of a user job is assigned a PROFILE-ID defined with HIERARCHY=*NO (MODIFY-SDF-PARAMETERS command), then no system-wide LOGON procedures are called for this user job. Only the user-specific LOGON procedures, when present, are called in this case.

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-ADMINISTRATION, 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).

If a user-specific or system-wide LOGON procedure file cannot be opened, then an error message is output. LOGON processing is then continued without activating this procedure.

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 [5]). LOGOFF procedures may not contain DO commands and may not be terminated with the ENDP-RESUME command. Furthermore, they should be uninterruptible (see [“Interruptibility of procedures” on page 65](#)).

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 [“Administration of system-wide procedure files” on page 68](#)). 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.

If the user ID of a user job is assigned a PROFILE-ID defined with HIERARCHY=*NO (MODIFY-SDF-PARAMETERS command), then no system-wide LOGOFF procedures are called for this user job. Only the user-specific LOGOFF procedures, when present, are called in this case.

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-ADMINISTRATION, SAT-FILE-MANAGEMENT and SAT-FILE-EVALUATION privileges.

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

If a user-specific or system-wide LOGOFF procedure file cannot be opened, then an error message is output. LOGOFF processing is then continued without activating this procedure.

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 **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.

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 via	Interrupt allowed			
	Program	no	yes	
	Procedure	yes / no	no	yes
<code>[K2]</code> key		R	R	A
<code>[K2]</code> -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
<code>//HOLD-PROGRAM</code>		R	C	C
<code>//EXECUTE-SYSTEM-CMD</code>		R	C	A
<code>/HOLD-PROGRAM</code>		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 (CLISSET).
- 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 \neq SYSCMD).

Rejection means:

- `[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 supported:

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

Administration of system-wide procedure files

The system-wide procedure files are created by systems support. Systems support can also provide one call and one include procedure each for LOGON and LOGOFF processing. LOGON and LOGOFF procedures are activated and deactivated with the MODIFY-SDF-PARAMETERS command. Activation and deactivation can be defined via the SCOPE operand temporarily or permanently for the current session and immediately or starting with the next session. Permanent changes are also entered in the SDF parameter file. All definitions can be displayed with the SHOW-SDF-PARAMETERS command. During activation, the file name can be specified explicitly for the desired procedure type (e.g. in the SYSTEM-LOGON-PROC operand for the call procedure for LOGON) or, if the procedure is stored under a standard name, using the value *STD. The following standard names can be used:

- \$TSOS.SYS.SDF.LOGON.SYSPROC (call procedure for LOGON)
- \$TSOS.SYS.SDF.LOGON.SYSINCL (include procedure for LOGON)
- \$TSOS.SYS.SDF.LOGOFF.USERPROC (call procedure for LOGOFF)
- \$TSOS.SYS.SDF.LOGOFF.USERINCL (include procedure for LOGOFF)

The system-wide procedures must have the USER-ACCESS=*ALL-USERS and ACCESS=*READ protection attributes. If a BASIC-ACL is set up for a procedure file, then all users must have execution rights.

The names of the system-wide procedure files can be entered in the SDF parameter file with the MODIFY-SDF-PARAMETERS command. It is also possible with this command to deactivate the system-wide LOGON procedures for all following LOGON processes in the current session without changing any entries in the parameter file.

3.4.2 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.

3.5 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 or job variable or by an 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 '¶meter', 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.

Note

Before job variable replacement, ACS, if required, replaces the alias of a JV by its real path name (in accordance with the alias catalog entry).

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 SDF interface like commands and not like input data.
- In procedures, 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 3 [on page 75](#)).
- Expressions cannot be nested.
- With interactive mode or 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 builtin function JV() in the “Commands” manual [1] or in the “SDF-P” manual [5]. 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”.

Procedure parameters of an S procedure are S variables and are replaced in expressions also.

Expressions are replaced in commands. 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)
/&(amp;cmd) _____ (3)
%      3 :20SG:$USER1.ALT.SYS.LOGON.USERPROC.X1
%     51 :20SG:$USER1.ALT.SYSSDF.USER.EXAMPLE.1
%     21 :20SG:$USER1.DATEI.1
%     48 :20SG:$USER1.DATEI.2
%     84 :20SG:$USER1.DATEI.3
%     66 :20SG:$USER1.OUT.SORT1-2
%      3 :20SG:$USER1.PROC.JV
%:20SG: 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 :20SG:$USER1.PROC.JV _____ (6)
%:20SG: PUBLIC:      1 FILE RES=      3 FRE=      2 REL=      0 PAGES

```

- (1) The job variable name 'EGON' is declared.
- (2) The job variable EGON is assigned the value 'SHOW-FILE-ATTR' (short for the name of the command SHOW-FILE-ATTRIBUTES).
- (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 EGON is changed. It now contains only part of the command name ('-FILE-ATTRIBUTES') and the partially qualified file name 'PROC.'
- (5) Job variable EGON 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
/&(amp) proc.jv ----- (2)
%      3 :20SG:$USER1.PROC.JV
%:20SG: PUBLIC:      1 FILE RES=          3 FRE=          2 REL=          0 PAGES
/cmd='PRINT-DOCUMENT' ----- (3)
/sh-var cmd
CMD = PRINT-DOCUMENT
/&(amp) proc.jv ----- (4)
% SCPO810 SPOOLOUT FOR FILE ':20SG:$USER1.PROC.JV' ACCEPTED. TSN: '1FAL', SPOOL
OUT-NAME: 'SDFTEST', MONJV: '*NONE'
% SCP1025 PRINT JOB ACCEPTED BY SERVER 'GH5090Y0' WITH TSN '5BXC'
/&(amp;:20sg:cmd) proc.jv ----- (5)
%      3 :20SG:$USER1.PROC.JV
%:20SG: PUBLIC:      1 FILE RES=          3 FRE=          2 REL=          0 PAGES
/&(amp;jv(jv-name='CMD')) proc.jv ----- (6)
%      3 :20SG:$USER1.PROC.JV
%:20SG: 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 command PRINT-DOCUMENT 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 (4) 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)
then 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
/ERR:     IF-BLOCK-ERROR
/          WRITE-TEXT   C'*** Error &MC ***'
/          ELSE
/          WRITE-TEXT   C'*** Command &(&(JV-1)) executed ***'
/          END-IF
/END:     EXIT-PROC

```

Procedure execution:

```

/show-jv jv=cmd ----- (1)
%SHOW-FILE-ATTR
/call-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 for 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 passed to the builtin function via S variable JV-1.
- (3) The command SHOW-FILE-ATTRIBUTES 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-OPTIONS.

3.6 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 use abbreviations sparingly in automated command sequences.

3.6.1 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 (name parts linked by a hyphen), portions can be omitted from right to left, together with the respective hyphens.
- Within a name portion 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 keyword value from some other possible operand value whose range includes the string of the keyword value. The asterisk alone, even if it is unique, does not represent a valid abbreviation.
- 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 to allow for potential extensions in subsequent versions. For reasons of compatibility, operand values which were previously written without a leading asterisk are still accepted without the asterisk.
- The name or partial name of a keyword value can also contain a period (e.g. *V4.7 or *OSD-V8.0). 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 among all valid command names
If a partial name of a command is specified in full, this command is unique compared to 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 among all valid statements of a loaded program
- an operand name is unique among 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 against renaming of commands, all names should be specified unabbreviated.

3.6.2 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 that are not to have the default settings need to be specified explicitly, input can often be dramatically reduced.

If operands do not have default settings, or if they do not have the defaults that you require, you can define your own defaults for interactive input on a task-specific basis (see [section "Task-specific default values" on page 93](#)).

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).

3.6.3 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 <operand-name>=<operand-value>. 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 change. For this reason, only keyword operands should be used in procedures.

3.6.4 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 on a long-term basis.

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 several 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.

Example

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

is the abbreviated notation for

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

3.7 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.

3.7.1 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.

3.7.2 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.

3.7.3 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.

3.8 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.

3.8.1 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.

3.8.2 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.

3.9 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 redisplay 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.

3.10 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 [5]). 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  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:

```

/c1p 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              : :20SG:$USER1.SYSSDF.USER.EXAMPLE.1
%                    VERSION : UNDEFINED
%CURRENT SDF OPTIONS :
% GUIDANCE           : *EXPERT
% LOGGING            : *INPUT-FORM
% CONTINUATION       : *NEW-MODE
% UTILITY-INTERFACE : *NEW-MODE
% PROCEDURE-DIALOGUE : *NO
% MENU-LOGGING       : *NO
% MODE               : *TEST
% CHECK-PRIVILEGES  : *YES
% DEFAULT-PROGRAM-NAME : PERCON
% FUNCTION-KEYS     : *STYLE-GUIDE-MODE
% INPUT-HISTORY      : *ON
% NUMBER-OF-INPUTS  : 20
% PASSWORD-PROTECTION: *YES
%/SHOW-FILE-ATTR     F-NAME=TEST                     _____ (5)
%/START-PERCON
%///SHOW-SDF-OPT
%///ASS-INPUT-F       FILE=*DISK-F(NAME=TEST)          _____ (7)
%///ASS-OUT-F         FILE=*DISK-F(NAME=&OUTFILE)
%/&OUTFILE=test4                                       _____ (8)
%///ASS-OUT-F         FILE=*DISK-F(NAME=TEST4)         _____ (9)
%///SEL-INPUT-REC    COND= ((20,4)=NUMERIC AND (25,5)=ALPHA)
%///START-CONV
%///END
%/SHOW-FILE-ATTR     F-NAME=TEST4
%/MOD-JOB-SW         ON=(1)
%/START-PROG         FROM-FILE=$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.

3.11 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 :20SG:$USER1.DATEI.3
%      3 :20SG:$USER1.PROC.JV
%:20SG: 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 :20SG:$USER1.DATEI.3
%S NNN NW      3 :20SG:$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).

4 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 under BS2000/OSD-BC V8.0A. 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 (if required the user ID-specific language settings can be modified with the MODIFY-MSG-ATTRIBUTES command). 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 47](#)).

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.

4.1 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 dd-mm-yy 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.  
%
```

4.2 Unguided dialog

4.2.1 EXPERT form

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

```

/sh-f-attr test.example.,file-struct=i _____ (1)
%      33 :20SG:$USER1.TEST.EXAMPLE.1 _____ (2)
%      72 :20SG:$USER1.TEST.EXAMPLE.2
%:20SG: PUBLIC:      2 FILES RES=      105 FRE=      26 REL=      9 PAGES
/sh-f-attr test,example.,file-struct=(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-struct=(i,s) _____ (6)
%      33 :20SG:$USER1.TEST.EXAMPLE.1 _____ (7)
%      72 :20SG:$USER1.TEST.EXAMPLE.2
%      48 :20SG:$USER1.TEST.EXAMPLE.3
%       3 :20SG:$USER1.TEST.EXAMPLE.4
%:20SG: 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-struct=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-struct=(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.

4.2.2 NO form

The dialog is continued as follows:

```
%CMD:sh-f-a test.,file-struct=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 :20SG:$USER1.TEST.EXAMPLE.1
%      72 :20SG:$USER1.TEST.EXAMPLE.2
%:20SG: PUBLIC:      2 FILES RES=      105 FRE=      26 REL=      9 PAGES
%CMD:sh-f-attr test,example.,file-struct=(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-struct=(i,s) _____ (6)
test.example.,inf=name _____ (7)
%      33 :20SG:$USER1.TEST.EXAMPLE.1 _____ (8)
%      72 :20SG:$USER1.TEST.EXAMPLE.2
%      48 :20SG:$USER1.TEST.EXAMPLE.3
%      3  :20SG:$USER1.TEST.EXAMPLE.4
%:20SG: 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-struct=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-struct=(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.

4.3 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 by a sign (+ or -)
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 ([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
SO-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 :20SG:$USER1.TEST.EXAMPLE.1
%:20SG: 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-struct=(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-struct=i)”. The entry “file-struct=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 :20SG:$USER1.TEST.EXAMPLE.1
%          72 :20SG:$USER1.TEST.EXAMPLE.2
%:20SG: PUBLIC:          2 FILES RES=          105 FRE=          26 REL=          9 PAGES

```

The dialog is continued as follows:

```
%CMD:sh-f-attr test,example.,select=(file-struct=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-struct=i) _____ (12)
? _____ (13)
```

- (1) The SHOW-FILE-ATTRIBUTES command calls up the default information (size and name) on all ISAM files (“select=(file-struct=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-STRUCTURE=*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 :20SG:$USER1.TEST.EXAMPLE.1
%          72 :20SG:$USER1.TEST.EXAMPLE.2
%:20SG: 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 (`[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 `[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 `[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.

4.4 Guided dialog

4.4.1 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 (**[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 (**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 :20SG:$USER1.TEST.EXAMPLE.1
%          72 :20SG:$USER1.TEST.EXAMPLE.2
%          48 :20SG:$USER1.TEST.EXAMPLE.3
%           3 :20SG:$USER1.TEST.EXAMPLE.4
%:20SG: PUBLIC:      4 FILES RES=      156 FRE=      30 REL=      9 PAGES
%PLEASE ACKNOWLEDGE

```

4.4.2 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-struct=(i,s))
              *ALL or *BY-ATTRIBUTES()
OUTPUT      = *SYSOUT
              *NONE or *SYSOUT or *SYSLIST() 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-struct=(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 (DUE) key.

```

%      33 :20SG:$USER1.TEST.EXAMPLE.1
%      72 :20SG:$USER1.TEST.EXAMPLE.2
%      48 :20SG:$USER1.TEST.EXAMPLE.3
%       3 :20SG:$USER1.TEST.EXAMPLE.4
%:20SG: PUBLIC:      4 FILES RES=      156 FRE=      30 REL=      9 PAGES
%PLEASE ACKNOWLEDGE

```


4.4.3 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-struct=(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-struct=(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 **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 (**DUE**) key is pressed, SDF displays the application domain menu.

```

%          33 :20SG:$USER1.TEST.EXAMPLE.1
%          72 :20SG:$USER1.TEST.EXAMPLE.2
%          48 :20SG:$USER1.TEST.EXAMPLE.3
%           3 :20SG:$USER1.TEST.EXAMPLE.4
%:20SG: 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 **[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 **[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.

5 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**dmistration) 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 or more user syntax files 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.

5.1 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. 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.047 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:

- The 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.

5.2 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 Technology Solutions, 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 Technology Solutions.

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.

5.3 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.

5.4 Privileges in syntax files

In BS2000/OSD-BC 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].

5.5 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.

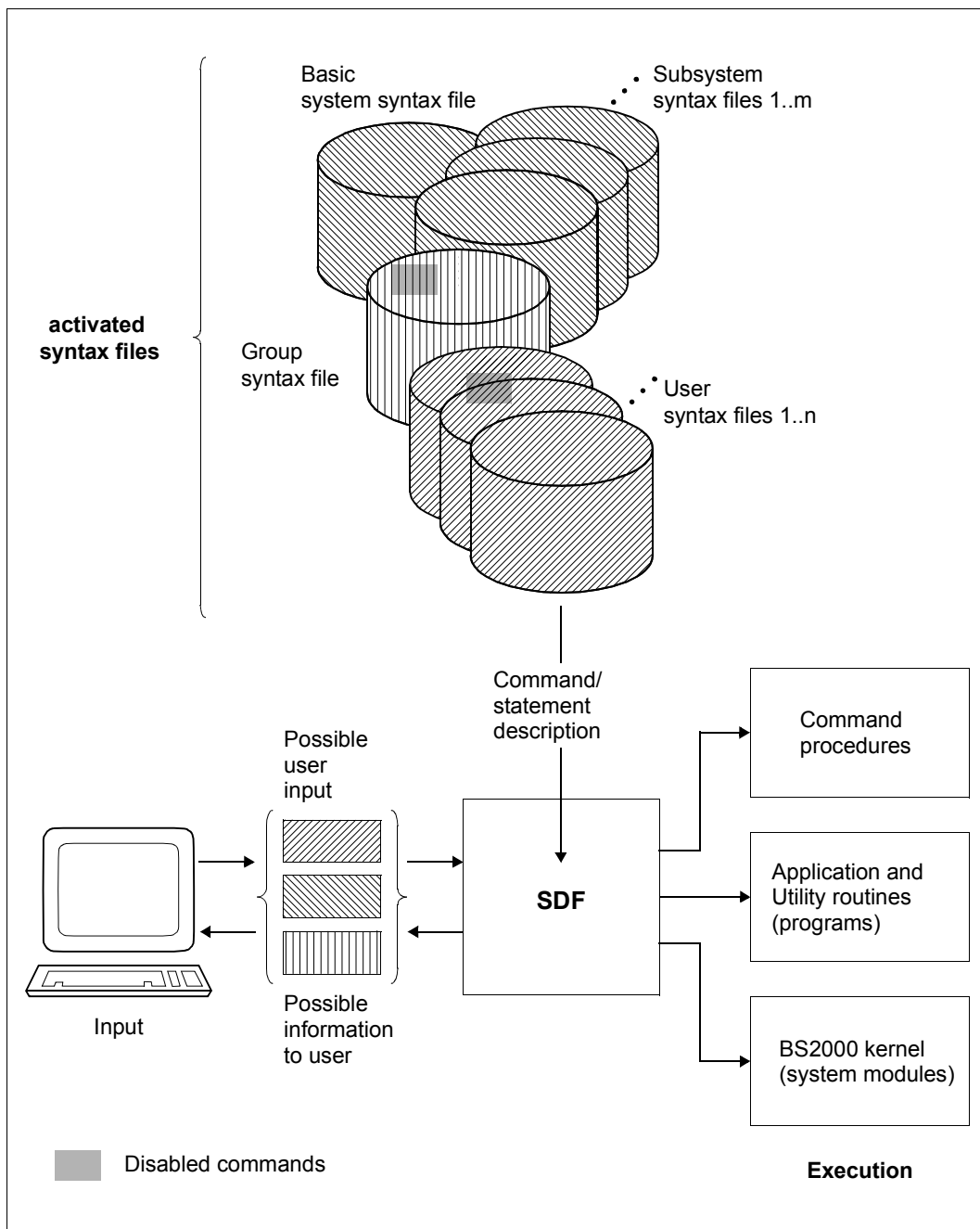


Figure 49: How SDF works when different syntax file types are activated

5.6 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” on page 171](#).

- Converting of a syntax file in the old format. With SDF V2.0A, the internal logical structure of syntax files has been changed. SDF also supports the old format, however, with reduced performance. If there is a syntax file in the old format still existing, it can be converted from old to new format by means of the SDF-I statement CONVERT-SYNTAX-FILE (see [“Compatibility of syntax file formats” on page 169](#)).
- 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.

6 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.

6.1 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 and can be found in the manual “Commands” [1].

Kommando	Meaning
HELP-SDF	Provide information on SDF
MODIFY-SDF-OPTIONS	Activate user syntax file and change SDF option settings
MODIFY-SDF-PARAMETERS	Modify SDF parameters
REMARK	Insert remarks in command file
RESET-INPUT-DEFAULTS	Delete task-specific default values
RESTORE-SDF-INPUT	Restore previous input
SHOW-CMD	Output command syntax description
SHOW-INPUT-DEFAULTS	Output task-specific default values
SHOW-INPUT-HISTORY	Output buffered input to SYSOUT
SHOW-SDF-OPTIONS	Show active syntax files and SDF options
SHOW-SDF-PARAMETERS	Show entries in SDF parameter file
SHOW-SYNTAX-VERSIONS	Display syntax file versions
WRITE-TEXT	Write text to SYSOUT or SYSLST

Table 5: Commands of the SDF domain

6.2 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 138](#)) 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 153](#)).

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).

EXECUTE-SYSTEM-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> ,INFORMATION-LEVEL = * MAXIMUM / * MEDIUM / * MINIMUM ,LANGUAGE = * 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).

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
<p>SYNTAX-FILE = *<u>UNCHANGED</u> / *ADD(...) / *REMOVE(...) / *NONE</p> <p>*ADD(...)</p> <p> ADD-NAME = *<u>STD</u> / list-poss(2000): *<u>STD</u> / <filename 1..54></p> <p>*REMOVE(...)</p> <p> REMOVE-NAME = *<u>LAST</u> / *<u>ALL</u> / *<u>BY-SELECTION</u> / list-poss(2000): <filename 1..54> / *<u>STD</u></p> <p>,GUIDANCE = *<u>UNCHANGED</u> / *<u>EXPERT</u> / *<u>NO</u> / *<u>MAXIMUM</u> / *<u>MEDIUM</u> / *<u>MINIMUM</u></p> <p>,LOGGING = *<u>UNCHANGED</u> / *<u>INPUT-FORM</u> / *<u>ACCEPTED-FORM</u> / *<u>INVARIANT-FORM</u></p> <p>,UTILITY-INTERFACE = *<u>UNCHANGED</u> / *<u>OLD-MODE</u> / *<u>NEW-MODE</u></p> <p>,PROCEDURE-DIALOGUE = *<u>UNCHANGED</u> / *<u>YES</u> / *<u>NO</u></p> <p>,CONTINUATION = *<u>UNCHANGED</u> / *<u>OLD-MODE</u> / *<u>NEW-MODE</u></p> <p>,MENU-LOGGING = *<u>UNCHANGED</u> / *<u>NO</u> / *<u>YES</u></p> <p>,FUNCTION-KEYS = *<u>UNCHANGED</u> / *<u>STYLE-GUIDE-MODE</u> / *<u>BY-TERMINAL-TYPE</u> / *<u>OLD-MODE</u></p> <p>,INPUT-HISTORY = *<u>UNCHANGED</u> / *<u>ON(...)</u> / *<u>OFF</u> / *<u>RESET</u></p> <p>*ON(...)</p> <p> NUMBER-OF-INPUTS = *<u>UNCHANGED</u> / <integer 1..100></p> <p> ,PASSWORD-PROTECTION = *<u>UNCHANGED</u> / *<u>NO</u> / *<u>YES</u></p>	

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 = <u>*STMT</u>(...)/ *CMD (...)/ *ALL / <integer 1..9999> <u>*STMT</u>(...) STMT = <u>*ALL</u> / <structured-name 1..30 with-wild(50)> ,PROGRAM = <u>*CURRENT</u> / *ALL / <structured-name 1..30> *CMD(...) CMD = <u>*ALL</u> / <structured-name 1..30 with-wild(50)>	

RESTORE-SDF-INPUT

The RESTORE-SDF-INPUT statement redisplay 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 = <u>*LAST-STMT</u> / <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
<p>OBJECT = <u>*STMT</u>(...) / *CMD(...) / *ALL</p> <p><u>*STMT</u>(...)</p> <ul style="list-style-type: none"> STMT = <u>*ALL</u> / <structured-name 1..30 with-wild(50)> ,PROGRAM = <u>*CURRENT</u> / *ALL / <structured-name 1..30> <p>*CMD(...)</p> <ul style="list-style-type: none"> CMD = <u>*ALL</u> / <structured-name 1..30 with-wild(50)> <p>,OUTPUT = <u>*SYSOUT</u> / *SYSLST(...)</p> <p>*SYSLST(...)</p> <ul style="list-style-type: none"> SYSLST-NUMBER = <u>*STD</u> / <integer 1..99> <p>,INPUT-SERIAL-NUMBER = <u>*NO</u> / *YES</p>	

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
<p>ENTRIES = <u>g</u> / <integer 1..100> / *ALL</p> <p>,SELECT = <u>*STMT</u> / *CMD / *ALL</p> <p>,PATTERN = <u>*NONE</u> / <structured-name 1..30></p> <p>,INPUT-SERIAL-NUMBER = <u>*NO</u> / *YES</p>	

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>

,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>

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 the “Commands” manual [1]).

STEP

WRITE-TEXT

The WRITE-TEXT statement outputs any specified text to SYSOUT.

WRITE-TEXT	Alias: WRTX
<p>TEXT = ' ' / <c-string 1..1024 with-low> ,OUTPUT = *SYSOUT / *SYSLST(...) *SYSLST(...) SYSLST-NUMBER = *STD / <integer 1..99></p>	

6.3 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 83).
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.

6.4 SDF interface to high-level programming languages

SDF offers an interface for programs written in **High Level Languages**. 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 [\[4\]](#).



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!

7 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.

8 Installation of SDF

Detailed information on installing SDF and on hardware and software requirements can be found in the release notice for the software product SDF.

8.1 SDF installation files

SDF V4.7 can be run under BS2000/OSD-BC V8.0 or OSD/XC V4.0.

The following files are part of the standard product and are supplied along with SDF V4.7:

File	Notes
SIPLIB.SDF.047	TPR macro library
SPMLNK.SDF.047	Load library for SDF (for SQ servers)
SKMLNK.SDF.047	Load library for SDF (for SX servers)
SYSFGM.SDF.047.D	Release notice for SDF in German
SYSFGM.SDF.047.E	Release notice for SDF in English
SYSLIB.SDF.047	PLAM library containing the SDF macros, the definitions of interface functions for COBOL, FORTRAN and C, the object module CMDCSTM, and the interface modules for higher programming languages
SYSLNK.SDF.047	Load library for SDF (for S servers)
SYSMES.SDF.047	SDF message file
SYSRMS.SDF.047	RMS for SDF
SYSSDF.SDF.047	Syntax file for the commands from the SDF application area
SYSSII.SDF.047 ¹	Installation information file for installation with IMON
SYSSSC.SDF.047	SSCM catalog declarations for the SDF subsystem

File	Notes
SYSMES.SDF-I.041	SDF-I message file
SYSPRG.SDF-I.041	Program for merging software unit syntax files (see chapter “SDF-I” on page 165)
SYSSDF.SDF-I.041	Syntax file containing the START-SDF-I command
SYSSII.SDF-I.041 ¹⁾	Installation information file for installation with IMON
SYSPRG.SDF-U.041	Program that enables systems support staff to modify syntax files (see chapter “SDF-U” on page 187)
SYSSDF.SDF-U.041	Syntax file containing the SDF-U statements
SYSLNK.SDF-U.041	Program modules of SDF-U
SYSMES.SDF-U.041	SDF-U message file
SYSRMS.SDF-U.041	RMS for SDF-U V4.1
SYSPRG.SDF-PAR.011	Program for creating, modifying and displaying the SDF parameter file (see chapter “SDF-PAR” on page 217)
SYSMES.SDF-PAR.011	SDF-PAR message file
SYSPRC.SDF-PAR.011.112	Startup procedure for SDF-PAR
SYSSDF.SDF-PAR.010.USER	User syntax file containing the SDF-PAR statements
SYSSII.SDF-PAR.011 ¹⁾	Installation information file for installation with IMON

¹ SYSSII files are deleted from the target system after successful installation. However, they are saved as X elements in the \$SYSSAG.SOLLIB.IMON.SYSSII library.

8.2 Generating the subsystem catalog

SDF is loaded via DSSM. The subsystem SDF must be declared in the subsystem catalog for this purpose. Generation of the subsystem catalog with SSCM is described in the manual “Introductory Guide to Systems Support” [2].

The declarations for this are supplied in the file **SYSSSC.SDF.047**.

The name of the SDF parameter file is defined as \$TSOS.SYSPAR.SDF in the supplied declarations as a subsystem attribute.

The subsystem catalog is normally generated by IMON.

8.3 Preparing SDF

8.3.1 Preparing files for SDF

The IMON installation monitor provides the files for SDF and activates the necessary system syntax files and message files.



IMON activates the SDF syntax file (i.e. the file SYSSDF.SDF.047) as the basic syntax file. All other system syntax files, including SYSSDF.BS2CP.<bs2vers>, are activated as subsystem syntax files.

8.3.2 Starting SDF

SDF is automatically loaded at startup, after which the MODIFY-SDF-PARAMETERS command can be used to change the names for the syntax files.

9 Behavior in the event of errors

When the parameter file cannot be opened during loading:

If the parameter file cannot be opened when SDF is being loaded, then SDF requests a new parameter file via a console message. When the reply is “*STD”, then SDF activates the file \$TSOS.SYSSDF.SDF.047 as the basic system syntax file and the file \$TSOS.SYSSDF.BS2CP.<bs2vers> as the subsystem system syntax file. Once SDF has been loaded, the parameter file can be deleted, and a new parameter file can be created by means of the MODIFY-SDF-PARAMETERS statement.

When the basic system syntax file cannot be activated during loading:

If it is not possible to activate the basic system syntax file while loading SDF, an error message is displayed and a new system syntax file is requested. If the operator responds by entering the name of a valid basic system syntax file, the file in question is activated, but its name is not written into the parameter file. If no valid system syntax file has been specified, the loading operation is aborted. In this case the system must be loaded from the backup pubset. If no such pubset is available, then the disks must be restored with FDDRL (see the “FDDRL” manual [8]) or the system may need to be regenerated.

When the system syntax file is not cataloged with USER-ACCESS=*SPECIAL

If a system syntax file is shareable but not cataloged with USER-ACCESS= *SPECIAL, an error message is output and the USER-ACCESS attribute is set to *SPECIAL. If the system syntax file is not shareable, the operator must confirm this attribute change, otherwise the system syntax file cannot be activated.

Basic system syntax file without the MODIFY-SDF-PARAMETERS command

An error message is output if the basic system syntax file does not contain the MODIFY-SDF-PARAMETERS command. In this case, the operator can continue with the loading process, abort it or enter the name of another basic system syntax file.

10 SDF-I

Version: SDF-I V4.1
Privileges: STD-PROCESSING (for nonprivileged functions)
TSOS (for privileged functions)

The SDF-I utility routine merges group or system syntax files and performs consistency checks. The syntax files to be merged may be files supplied by Fujitsu Technology Solutions and/or created by the user (with SDF-A). Syntax files merged with SDF-I can also be unmerged using the same program. SDF-I supports syntax files with or without a PAM key.



Installation is normally performed using the installation monitor IMON. Since IMON does not merge the SUSFs (software unit syntax files), but activates them as the basic system syntax file and subsystem syntax files instead, SDF-I is only used now in special cases (e.g. when merging subsystem syntax files or group syntax files created by the customer or when displaying syntax file information).

10.1 SDF-I inputs/outputs

SDF-I merges one or more syntax files and an input syntax file to produce an output syntax file. The input file and the syntax files to be merged must be of the same type (system or group syntax files). The output syntax file is given the same type as the input syntax file.

The following files may serve as input files:

- SUSFs (software unit syntax files) supplied by Fujitsu Technology Solutions (e.g. the SUSF for the BS2000 basic configuration)
- the current INSF (installation syntax file) used by the system.

The following files may be merged with the input syntax file:

- SUSFs
- any other syntax file generated by the user with the aid of SDF-A (see the “SDF-A” manual [4]).

If there are syntax files (user, group or system) in the old format still existing, they can be converted to the new format with the CONVERT-SYNTAX-FILE statement (see also).



Warning!

Conversion from the old to the new format is irreversible. It is therefore advisable to save the old syntax file so that it can be accessed in case a problem occurs. You should also have a procedure with SDF-A statements to regenerate the syntax file.

More information on this subject can be found in [section “Notes” on page 169](#).

10.2 Calling and executing SDF-I

SDF-I does not read its statements via SDF and thus does not require activated syntax files for execution. Consequently, there is no guided dialog for SDF-I statements. Only the START-SDF-I command is defined in the syntax file SYSSDF.SDF-I.041, which is supplied with SDF-I.

The SDF-I utility routine can be started under any user ID with the aid of the following command:

START-SDF-I	Alias: SDF-I
VERSION = *STD / <product-version> ,MONJV = *NONE / <filename 1..54 without-gen-vers> ,CPU-LIMIT = *JOB-REST / <integer 1..32767 seconds>	

VERSION =

Allows the user to select the desired SDF-I version if multiple versions of SDF-I were installed with IMON. If the version is specified within single quotes, it may be preceded by the letter C (C-STRING syntax).

If the product was not installed using IMON or if the specified version does not exist, VERSION=*STD applies.

VERSION = *STD

Calls the SDF-I version with the highest version number.

VERSION = <product-version>

Specifies the SDF-I version in the format [[C]][V][m].nasos[.].

MONJV =

Specifies a monitoring job variable to monitor the SDF-I run.

MONJV = *NONE

No monitoring job variable is used.

MONJV = <filename 1..54 without-gen-vers>

Name of the job variable to be used.

CPU-LIMIT =

Maximum CPU time (in seconds) which the program may use during execution.

CPU-LIMIT = *JOB-REST

The remaining CPU time for the job is to be used.

CPU-LIMIT = <integer 1..32767 seconds>

Only the specified time is to be used.

SDF-I can run interactively or as a batch job. If several syntax files are to be merged, it is advisable to run SDF-I as a batch job.

The statements CONVERT-SYNTAX-FILE and SHOW-SYNTAX-FILE may be entered without a preceding OPEN statement, i.e. independent of the merge operation.

The OPEN statement must be used prior to each merge operation to assign the input syntax file and the output syntax file. MERGE statements are then used to specify the syntax files to be added. A separate MERGE statement is required for each syntax file to be incorporated.

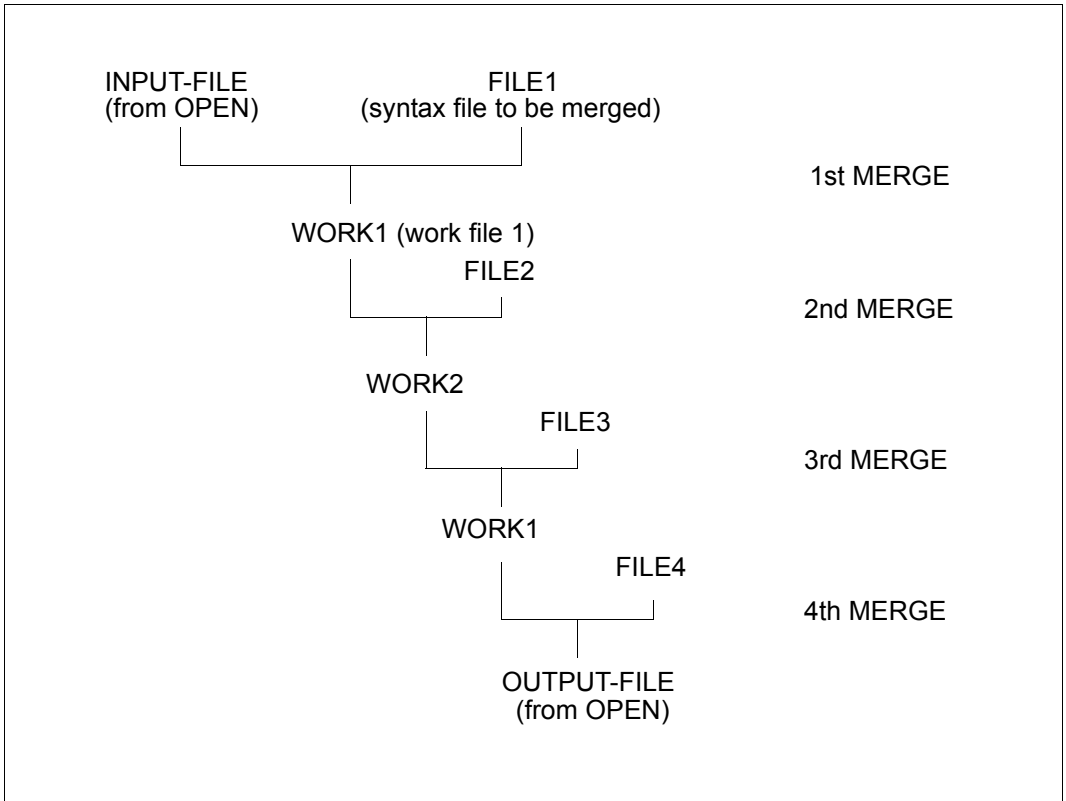
Each MERGE statement produces a work file. This work file contains all syntax files merged up to that point, including the input syntax file. The END statement or the OPEN statement for the next syntax file converts the work file into the output file. SDF-I is terminated by means of the END statement.

The REMOVE statement can be used to unmerge syntax files merged with the aid of SDF-I. Just like MERGE, a REMOVE statement must be preceded by an OPEN statement defining the input and output syntax files.

10.3 Handling work files

SDF-I uses up to two work files, which it creates and deletes as required. These files are created under the name SYSSDF.SDF-I.tsn. (where *version* is the SDF-I version, e.g. 041 for SDF-I V4.1), tsn is the TSN of the calling task and n=1 or 2 is the number of the work file.

In general, users must make sure that there are no own files existing with these names. The two work files are used alternately when multiple syntax files are to be merged into the input syntax file (INPUT-FILE operand in the OPEN statement). The illustration below shows how they are used in the merge procedure.



How two work files are used in a merge run

These work files have the size of the output file. The execution of SDF-I therefore requires a sufficient reserve of disk storage space; otherwise, MERGE statement processing will be aborted.

10.4 Handling SDF standard statements and domains

The SDF standard statements are contained in the SDF syntax file. A separate SUSF is supplied for SDF.

SDF-I merges the standard statements when the syntax file for SDF is to be merged. The standard statements are then available to all the programs in the output syntax file. If a given SUSF is merged into a syntax file that already includes the SDF syntax file, the programs in the newly included syntax file also have access to the standard statements.

All application domains from the input syntax files are merged into the list of application domains of the output syntax file. Domains to which no commands belong are not removed by SDF-I. If desired, you can remove the redundant domain by using the SDF-A/SDF-U statement REMOVE.

10.5 Notes

Input

- The statements and their operands can be abbreviated as long as they remain unique. This option should be used in interactive mode only.
- SDF-I supports continuation lines. The maximum length of a statement is 2044 characters.
- The statements for a merge operation must be issued in the following order (several consecutive MERGE statements are possible):

```
OPEN  
MERGE/REMOVE  
END
```

Compatibility of syntax file formats

In the past, further development of SDF sometimes also included changes to the syntax file format. This resulted in formats V1, V2, V3, V4 and, most recently, the format which is currently used, V4.1, being created. Formats V1 through V4 are historical and can only occur in older syntax files. The format of a syntax file is contained in the output information of the SHOW-SYNTAX-FILE statement.

The internal logical structure of a syntax file altered with the transition from format V1 to V2. Consequently syntax files with format V2 or higher are also referred to as syntax files in the “new” format (V1 was the “old” format).

The following must be observed when processing with SDF-I:

- SDF-I processes only input syntax files in the new format (format V2 and higher). If syntax files in the old format still exist, you can convert them to the new format using the CONVERT-SYNTAX-FILE statement.
- Syntax files with the current syntax file format (V4.1 format) can not be processed with SDF-I versions < 4.1.
- When files are merged, the format of the output syntax file depends on the format of the input syntax files. SDF-I generates the output syntax file with the highest format that appears among the input syntax files. If only syntax files with V2 and V3 format are merged, for example, the output syntax file receives the V3 format.
- SDF-I enters the name of the product SDF-A in the output syntax file as "SDF-A". In syntax files with earlier formats (OLD or V1-FORMAT), the name of SDF-A is defined as "SDFA". If you process such a syntax file is processed with SDF-I, you should have the actually entered name displayed with the SHOW-SYNTAX-FILE statement and only use that name in the SDF-I statements.
- During an OPEN/MERGE sequence, products that are already contained in the syntax file are replaced by the new versions when merged. The product SDF with a version < V4.0 is an exception.

10.6 SDF-I statements

10.6.1 Overview of functions

Display function

SHOW-SYNTAX-FILE Outputs information on group syntax files and system syntax files.

Convert syntax files

CONVERT-SYNTAX-FILE Converts any syntax file (group, system, user) in old format into a new-format syntax file.

Merge and unmerge syntax files *(Privileged functions)*

MERGE Specifies the syntax file to be merged to the input syntax file and checks the SUSF (if any) for consistency, comparing it to a specific product name or product version.

OPEN Specifies the following:

- the input syntax file to which a syntax file is to be merged or from which a syntax file is to be unmerged
- the output syntax file in which the result of the merging/unmerging procedure is to be kept.

REMOVE Removes syntax files previously merged by SDF-I from the input syntax file.

Terminate SDF-I

END Terminates the SDF-I run.

10.6.2 Description of the statements

CONVERT-SYNTAX-FILE Convert syntax file to new format

Privileges: **STD-PROCESSING, TSOS**

The CONVERT-SYNTAX-FILE statement converts any syntax file (group, system, user) with an old format (format V1) into the new format (format V4.1).

CONVERT-SYNTAX-FILE
INPUT-FILE = <filename 1..54> , OUTPUT-FILE = <filename 1..54>

INPUT-FILE = <filename 1..54>

Defines the old-format syntax file to be converted.

OUTPUT-FILE = <filename 1..54>

Defines the name of the new-format output syntax file.

Notes

- Conversion from the old to the new format is irreversible. It is therefore advisable to save the old syntax file so that it can be accessed in case a problem occurs. You should also have a procedure with SDF-A statements to be used to regenerate the syntax file if required.
- Syntax files converted using SDF-I V4.1A must not be processed with SDF-I versions < 4.1.

END
Terminate SDF-I**Privileges: STD-PROCESSING, TSOS**

The END statement terminates the SDF-I run and generates the output file whose name was specified in the OUTPUT-FILE operand of the OPEN statement.

END

This statement has no operands.

MERGE

Merge syntax files

Privileges: TSOS

The MERGE statement merges the input syntax file with the syntax file specified here. The latter must be of the same type (system or group syntax file) as the input syntax file.

During merging, programs with their associated statements are transferred as one unit; i.e. they may replace an entire program which already exists in the input file. It is not possible to merge multiple versions of a product.

Format 1: Merging software unit syntax files

MERGE

```

FILE = <filename 1..54>
, CHECK-PRODUCT = *NO / <structured-name 1..15>
, CHECK-VERSION = *NO / <c-string 1..7>
, REPLACE-PRODUCT = YES / NO
, UPDATE-SDF-GLOBALS = YES / NO

```

FILE = <filename 1..54>

Specifies the syntax file to be merged with the input file named under OPEN. If MERGE statements were already executed subsequent to the OPEN, the syntax file is merged into the work file currently in use, which is the file containing the syntax files merged beforehand with MERGE (see [page 168](#)).

CHECK-PRODUCT =

Determines whether the syntax file to be merged is subjected to a check for consistency with a given product name. If the specified syntax file contains more than one software unit, message SDI0023 is issued but the merge run is completed. It is then no longer possible, however, to unmerge (with REMOVE) one or all of the SUSFs contained in this syntax file.

CHECK-PRODUCT = *NO

No name consistency check takes place.

CHECK-PRODUCT = <structured-name 1..15>

Gives the product name of the software unit contained in the specified syntax file (e.g. SPOOL, TIAM, etc.).

CHECK-VERSION =

Determines whether the syntax file to be merged is subjected to a check for consistency with a given version number.

CHECK-VERSION = *NO

No version consistency check takes place.

CHECK-VERSION = <c-string 1..7>

Version number of a software unit. The version numbers of the SUSFs supplied by Fujitsu Technology Solutions correspond to the SDF data type <product-version>(see the "Commands" manual [1]).

Format: 'nn.naxx', where:

nn.na = version number of the software unit,
xx = update level of the syntax file.

It is possible to specify only the left-hand part of the version number. The check uses that length which has been specified.

REPLACE-PRODUCT =

Specifies whether any existing objects in the input syntax file are to be overwritten by objects with the same internal or external name from the syntax file specified here.

REPLACE-PRODUCT = YES

Objects in the input syntax file are overwritten by objects with the same internal or external name. This enables internal or external names to be renamed.

REPLACE-PRODUCT = NO

If objects with the same internal or external name are encountered, the merge run is aborted.

UPDATE-SDF-GLOBALS =

Specifies whether the global information of the input syntax file is to be overwritten (i.e. updated).

UPDATE-SDF-GLOBALS = YES

The global information of the input syntax file is overwritten with the global information of the syntax file specified here, but the version number of the input syntax file is retained.

UPDATE-SDF-GLOBALS = NO

The global information of the input syntax file is not overwritten.

Format 2: Merging user-generated syntax files

MERGE
FILE = <filename 1..54> , REMOVE-ID = <structured-name 1..15> , REPLACE-PRODUCT = <u>YES</u> / NO

FILE = <filename 1..54>

Specifies the syntax file to be merged with the input file. If MERGE statements were already executed subsequent to the OPEN, the syntax file is merged into the work file currently in use, which is the file containing the syntax files merged beforehand with MERGE (see [page 168](#)).

REMOVE-ID = <structured-name 1..15>

Short name of the syntax file (required for subsequent deletion).

REPLACE-PRODUCT =

Specifies whether any existing objects in the input syntax file are to be overwritten by objects with the same internal or external name from the syntax file specified here.

REPLACE-PRODUCT = YES

Objects in the input syntax file are overwritten by objects with the same internal or external name. This enables internal or external names to be renamed.

REPLACE-PRODUCT = NO

If objects with the same internal or external name are encountered, the merge run is aborted.

Notes

- If an error occurs when handling a MERGE operation, processing resumes with the next MERGE statement. In this case SDF-I continues with the status of the last successful MERGE statement. The spin-off mechanism is initiated on program termination. If subsequent MERGE statements cannot be executed either, the output file reflects the status of the last successful MERGE. It will not contain all the desired products but will still be a valid syntax file.
- If message SDI0035 appears after a MERGE statement for one of the syntax files supplied by Fujitsu Technology Solutions, the supplied syntax file must be handled like a user-generated syntax file (see format 2), not an SUSF (see format 1).

- If multiple syntax files for a number of products are to be merged, it is better to regenerate the system and group syntax files from the original syntax files by means of a batch task. The additional overhead in terms of memory is offset in this case by the increased integrity of the generated syntax files.

OPEN

Assign input and output syntax files

Privileges: TSOS

The OPEN statement assigns an input syntax file and an output syntax file. The input syntax file is merged with the new syntax files (MERGE statement). SDF-I stores the result of this process in the output syntax file. SDF-I does not modify the input syntax file.

This statement is mandatory for both merging and unmerging.

The input syntax file must be a system or group syntax file. The output syntax file automatically receives the same type as the input syntax file.

A version number may optionally be specified for the output file. If the output file was activated with SDF (e.g. via MODIFY-SDF-OPTIONS), the SHOW-SDF-OPTIONS command can be used to display this version number..

OPEN
<pre> INPUT-FILE = <filename 1..54> , OUTPUT-FILE = <filename 1..54> , VERSION = <u>'UNDEFINED'</u> / <c-string 1..12> / <alphanum-name 1..12> , FORMAT = *<u>STD</u> / *OLD / *NEW , BLKCTRL = *<u>STD</u> / *PAMKEY / *DATA , WRITE-MODE = *<u>NEW</u> / *REPLACE </pre>

INPUT-FILE = <filename 1..54>

Name of the syntax file to be used as input file. This file is merged with the file specified in the MERGE statement.

SDF-I does not check the logical structure of input syntax files but assumes that they have been correctly generated using SDF-A.

As a rule, the input file is an INSF either supplied by Fujitsu Technology Solutions or created in a previous SDF-I run. If such an INSF is not available, the SUSF of SDF should be specified as the input file.

OUTPUT-FILE = <filename 1..54>

Name of the output file after completion of all MERGE and REMOVE statements. It is given the same type (system or group syntax file) as the input file.

An output file under a foreign user ID can only be created by system administration.

VERSION = 'UNDEFINED' / <c-string 1..12> / <alphanum-name 1..12>

Version number stored in the global information of the output file and displayed (following activation of this syntax file) as a result of the SHOW-SDF-OPTIONS command. If no value is explicitly specified, 'UNDEFINED' is stored in the global information.

FORMAT =

Defines the format with which the output syntax file is to be created. The SHOW-SYNTAX-FILE statement (see [page 183](#)) shows the format entry for a syntax file as V1, V2, V3, V4 or V4.1. The current format is V4.1. The other formats are historical and can only occur in older syntax files.

FORMAT = *STD

Creates an old-format output syntax file if at least one of the syntax files that were processed for it has the new structure. If only syntax files with the old format are present, *STD functions in the same way as FORMAT=*OLD.

FORMAT = *NEW

Explicitly creates a new-format output syntax file. The input syntax files may have any structure.

FORMAT=*NEW is effective only if followed by at least one successful MERGE statement. Any subsequent REMOVE statement does not convert the entire syntax file but only the file contents affected by the REMOVE statement (e.g. global information, command lists, program lists, ...).

FORMAT = *OLD

The operand value is supported for compatibility reasons only.

Syntax files with the old structure (format V1) should not be created any more. Furthermore in this case SDF-I only processes input syntax files with this old structure.

BLKCTRL =

Defines whether the output syntax file and SDF-I files are created with or without PAM keys.

BLKCTRL = *STD

The block control setting of the current task is taken over.

BLKCTRL = *DATA

The output syntax file and SDF-I files are created without PAM keys.

BLKCTRL = *PAMKEY

The output syntax file and SDF-I files are created with PAM keys. The files can only be saved to disks that support PAM keys.

WRITE-MODE =

Defines whether or not the output syntax file exists.

SDF-I works faster if an output file of roughly the required size exists. For example, the user can create a file with the necessary file attributes (primary/secondary allocation) using the CREATE-FILE command, before executing SDF-I.

WRITE-MODE = *NEW

An error message is output if the output syntax file already exists. SDF-I calculates the output syntax file size from the size of the input syntax files.

WRITE-MODE = *REPLACE

The output syntax file must already exist, otherwise an error message is output. SDF-I uses the cataloged file attribute (primary/secondary allocation) for the output syntax file and SDF-I files. The existing output file is overwritten.

Note

If an error occurs during OPEN processing, all subsequent MERGE statements until a new OPEN statement are rejected.

REMOVE

Unmerge syntax files

Privileges: TSOS

A syntax file merged with SDF-I can be unmerged using the REMOVE statement.

Format 1: Merging software unit syntax files

REMOVE
PRODUCT-NAME = <structured-name 1..15>

PRODUCT-NAME = <structured-name 1..15>

Defines the name of the product whose syntax file is to be removed from the input file assigned in the preceding OPEN statement.

This operand can only be used for Fujitsu Technology Solutions products. It is identical with the product name (SUSF name), which is output for syntax files via the SHOW-SYNTAX-FILE statement.

Format 2: Unmerging user-generated syntax files

REMOVE
REMOVE-ID = <structured-name 1..15>

REMOVE-ID = <structured-name 1..15>

Defines the name of the product whose syntax file is to be removed from the input file assigned in the preceding OPEN statement.

This operand can only be used for syntax files the user himself has generated. It is identical with the REMOVE-ID declared by means of MERGE.

Example

Unmerging a software unit syntax file

The input syntax file *SF.base* contains the following SUSFs: L1 Version 1, L2 Version 1, L3 Version 1, L4 Version 1.

The syntax file *SF.in* to be merged contains the SUSF L2 Version 2.

```

OPEN INPUT-FILE=SF.base,OUTPUT-FILE=SF.out _____ (1)
REMOVE PRODUCT-NAME=L2 _____ (2)
MERGE FILE=SF.in _____ (3)
END _____ (4)

```

- (1) The input syntax file *SF.base* and the name of the output syntax file *SF.out* are defined.
- (2) Product L2 is removed from the input syntax file *SF.base* because the next step involves merging of the syntax file *SF.in* which contains a new version of product L2. This ensures that no residues of Version 1 remain in the syntax file (e.g. if a command had been omitted in Version 2, this command would not be removed by merely merging Version 2).
- (3) The syntax file *SF.in* is merged with the input syntax file *SF.base* to produce the internal SDF-I work file (see [section "Handling work files" on page 168](#)).
- (4) SDF-I is terminated. The work file receives the name *SF.out* defined via OPEN and contains the following SUSFs: L1 Version 1, L2 Version 2, L3 Version 1, L4 Version 1. SUSF L2 is now present in its new version.

Notes

- Only syntax files merged beforehand with the SDF-I statement MERGE can be unmerged.
- Objects overwritten with MERGE... REPLACE-PRODUCT=YES are not restored.
- If multiple syntax files for a number of products are to be merged, it is better to regenerate the system and group syntax files from the original syntax files by means of a batch task. The additional overhead in terms of memory is offset in this case by the increased integrity of the generated syntax files.

SHOW-SYNTAX-FILE

Output information on syntax file

Privileges: **STD-PROCESSING, TSOS**

The SHOW-SYNTAX-FILE statement returns information on a system or group syntax file; this can be either a currently processed or explicitly defined syntax file.

This statement displays the following information:

- name, type, version and format of the syntax file
 - The current format is V4.1. Formats V1 through V4 are historical and can only occur in older syntax files.
- version information on products included in the syntax file
- global information from the syntax file (in sections)
- privileges of commands only for INFORMATION=CMD-INTERFACE)

SHOW-SYNTAX-FILE
<p>FILE = *CURRENT / *INPUT-FILE / <filename 1..54> ,INFORMATION = ALL-ATTRIBUTES / VERSIONS / GLOBALS / CMD-INTERFACE ,PRODUCT-NAME = *ALL / <structured-name 1..15></p>

FILE =

Defines the system or group syntax file for which information is to be output.

FILE = *CURRENT

The operand value can only be specified after at least one preceding OPEN statement. Information relates to:

- the syntax file specified under INPUT-FILE in an immediately preceding OPEN statement
- the current temporary SDF-I work file after a MERGE or REMOVE statement.

FILE = *INPUT-FILE

The operand value can only be specified after at least one preceding OPEN statement. Information relates to the input syntax file specified under INPUT-FILE in the OPEN statement.

FILE = <filename 1..54>

Information related to the explicitly specified system or group syntax file is output.

INFORMATION =

Defines the type of information which is output. All outputs include the syntax file type (GROUP or SYSTEM).

INFORMATION = ALL-ATTRIBUTES

The following information is output:

- name, type, version and format of the syntax file
- versions of all SUSFs contained in the specified syntax file (*CUSTOM* is output as the version for user-created syntax files)
- SDF global information (in sections)

INFORMATION = VERSIONS

The following information is output:

- name, type, version and format of the syntax file
- versions of all SUSFs contained in the specified syntax file (*CUSTOM* is output as the version for user-created syntax files)

INFORMATION = GLOBALS

The following information is output:

- name, type, version and format of the syntax file
- SDF global information (in sections)

INFORMATION = CMD-INTERFACE

Outputs the name, type, version and format of the syntax file and a list of commands defined in the syntax file. The list also includes the privileges for commands (see example on [page 185](#)).

PRODUCT-NAME =

This operand is effective only if INFORMATION=CMD-INTERFACE is set. It specifies if only the commands for a specific product are to be output.

PRODUCT-NAME = *ALL

All commands of the specified syntax file are output.

When an SUSF (software unit syntax file) or INSF (installation syntax file) supplied by Fujitsu Technology Solutions is output, commands created by the customer and merged into these files are not displayed.

PRODUCT-NAME = <structured-name 1..15>

All commands of the specified product are output.

If commands created by the customer were merged into an SUSF (software unit syntax file) or INSF (installation syntax file) supplied by Fujitsu Technology Solutions are displayed, then the REMOVE-ID used during the merge must be used as the product name.

Example

```

/start-sdf-i _____ (1)
*% BLS0500 PROGRAM 'SDF-I', VERSION 'Vnn.nAnn' OF 'yyyy-mm-dd' LOADED
% BLS0552 COPYRIGHT (C) FUJITSU TECHNOLOGY SOLUTIONS 2009.ALL RIGHTS
RESERVED
*show-syntax-file $.syssdf.bs2cp.170,information=cmd-interface _____ (2)
% SYNTAX FILE : :10SH:$TSOS.SYSSDF.BS2CP.170
% TYPE : SYSTEM VERSION : SESD17.0A300 FORMAT : V4.1 _____ (3)
%
% COMMANDS INFORMATIONS :
% -----
%          EXT.NAME                  INT.NAME  ENTRY  INT  PRODUCT
%          -----                  -
% $CMSSCAS                             $CMSSCAS DSCSCCH ISL
% ALL 10000000000000000000000000000000000000000000000000000000000000000000
% $CMSSCAV                             $CMSSCAV DSCVLCH ISL
% ALL 10000000000000000000000000000000000000000000000000000000000000000000
% $DMAWCC1                             $DMAWCC1 DMAWICA ISL
% ALL 11111111111111111111111111111111111111111111111111111111111111111111
.
.
% ADD-CONSOLE-FILTER                  $OPRACF  NBOAFIL ISL
% ALL 00000000000000000000000000000000010000000000000000000000000000000000
% ADD-DEVICE-DEPOT                   NKCADDE  NKCADDE ISL
% ALL 00000000000000000000000000000000010000000000000000000000000000000000
% ADD-FILE-LINK                      $DCOAFD  DCOADFL ISL
% ALL 00000100011000000000000000000000010000000000000000000000000000000000
.
.
% SHOW-FILE                          SHFIL   DSHOWCM ISL
% ALL 01000100011000000000000000000000010000000000000000000000000000000000
% BA 00000000000000000000000000000000000000000000000000000000000000000000
% BPA 00000000000000000000000000000000000000000000000000000000000000000000
% DA 01000100011000000000000000000000010000000000000000000000000000000000
% DPA 01000100011000000000000000000000010000000000000000000000000000000000
% GA 01000100011000000000000000000000010000000000000000000000000000000000
% CA 01000100011000000000000000000000010000000000000000000000000000000000
% SHOW-FILE-ATTRIBUTES                SHFAT   DCOFSDF ISL
% ALL 01000100011000000000000000000000010000000000000000000000000000000000
% SHOW-FILE-LINK                      SHFILI  DCORTF2 ISL
% ALL 00000100011000000000000000000000010000000000000000000000000000000000
.
.
% WRITE-ACCOUNTING-RECORD             WRACRE  NACPWAC ISL
% ALL 00000000010000000000000000000000010000000000000000000000000000000000
* _____ (5)

```

- (1) SDF-I is started.
- (2) Information is requested on the commands and their privileges contained in the basic BS2000 system syntax file. The output is displayed in the example in sections.
- (3) The syntax file has the current V4.1 format. Only for older syntax files another format (V1, V2, V3, V4) can be displayed.
- (4) The output includes, among other things, the privileges for the SHOW-FILE command (ALL line) and the possible input modes. The input modes correspond to the modes that were specified in the command definition with ADD-CMD (see the "SDF-A" [4] manual):

```
BA:  BATCH-ALLOWED
BPA: BATCH-PROC-ALLOWED
DA:  DIALOG-ALLOWED
DPA: DIALOG-PROC-ALLOWED
GA:  GUIDED-ALLOWED
CA:  CMD-ALLOWED
```

Following the input mode, the related privileges are output in the remainder of the line. Each character (0 or 1) represents a privilege. 1 means that execution of the command is permitted in this mode for a user job with this privilege. The order of privileges in this string corresponds to the order in which the privileges have been currently defined in the system. If no line is output for a particular input mode, then PRIVILEGE=*SAME is defined for the input mode implicitly, i.e. the privileges are exactly the same as those defined for the command.

- (5) Following the output, SDF-I requests further inputs with the '*' prompt.

11 SDF-U

Version: SDF-U V4.1

Privileges: TSOS

SDF-U is a program that allows system administration to modify group and system syntax files in accordance with the needs of the computer center. SDF-U is designed primarily for installing new syntax file versions and uses a subset of the functions and statements provided by the software product SDF-A (see the “SDF-A” manual [4]).

The statements to SDF-U are defined in a separate SDF-U syntax file.

Starting SDF-U

SDF-U can be started under any user ID with the aid of the following command:

START-SDF-U	Alias: SDF-U
VERSION = *STD / <product-version> ,MONJV = *NONE / <filename 1..54 without-gen-vers> ,CPU-LIMIT = *JOB-REST / <integer 1..32767 seconds>	

VERSION =

Allows the user to select the desired SDF-U version if multiple versions of SDF-U were installed with IMON. If the version is specified within single quotes, it may be preceded by the letter C (C-STRING syntax).

If the product was not installed using IMON or if the specified version does not exist, VERSION=*STD applies.

VERSION = *STD

Calls the SDF-U version with the highest version number.

VERSION = <product-version>

Specifies the SDF-U version in the format [[C]][V][m]m.nasol[.].

MONJV =

Specifies a monitoring job variable to monitor the SDF-U run.

MONJV = *NONE

No monitoring job variable is used.

MONJV = <filename 1..54 without-gen-vers>

Name of the job variable to be used.

CPU-LIMIT =

Maximum CPU time (in seconds) which the program may use during execution.

CPU-LIMIT = *JOB-REST

The remaining CPU time for the job is to be used.

CPU-LIMIT = <integer 1..32767 seconds>

Only the specified time is to be used.

SDF standard statements

The SDF standard statements (see [page 189](#)) are not defined in the SDF-U syntax file, but rather in the SDF syntax file. They are offered globally to user programs that read in their statements via SDF.

Interrupting and resuming SDF-U

If SDF-U is interrupted via [K2](#) while a statement is being processed, the program may be resumed in one of two ways:

- If SDF-U is resumed using RESUME-PROGRAM, processing of the interrupted statement continues.
- If SDF-U is resumed using INFORM-PROGRAM, processing of the interrupted statement is aborted. SDF-U outputs message SDU0443 and awaits the input of a new statement.

11.1 SDF-U statements

11.1.1 Overview of functions

SDF standard statements

A detailed description of the SDF standard statements can be found in the [section “Standard statements” on page 146](#).

END	Terminate the SDF-U session
EXECUTE-SYSTEM-CMD	Execute a system command during the SDF-U session
HELP-MSG-INFORMATION	Output system message text to SYSOUT
HOLD-PROGRAM	Hold the SDF-U run
MODIFY-SDF-OPTIONS	Activate or deactivate a user syntax file and change the SDF settings
REMARK	Write comments into the statement string and document execution of a program
RESET-INPUT-DEFAULTS	Reset task-specific default values
RESTORE-SDF-INPUT	Redisplay previously entered statements or commands on the screen
SHOW-INPUT-DEFAULTS	Display task-specific default values
SHOW-INPUT-HISTORY	Display the contents of the input buffer
SHOW-SDF-OPTIONS	Display task-specific information on the active syntax files and the specifications for statement input and statement processing
SHOW-STMT	Display the syntax of a statement
STEP	Define a checkpoint in a sequence of SDF-U statements
WRITE-TEXT	Output a specific text to SYSOUT or SYSLST

Process and create syntax files

COPY	Copy the contents of a syntax file to the syntax file which has been opened
EDIT	Position to a command of the syntax file which has been opened
MODIFY-CMD	Modify the definition of a command in the syntax file which has been opened
MODIFY-VALUE	Modify an operand value definition in the open syntax file
OPEN-SYNTAX-FILE	Open a syntax file for processing with SDF-U (first statement after calling SDF-U)
REMOVE	Delete objects from the syntax file which has been opened
SET-GLOBALS	Modify general specifications pertaining to command input and processing

Display functions

SHOW	Output the contents of an open syntax file to SYSOUT or SYSLST
SHOW-CORRECTION- INFORMATION	Output correction information from the opened syntax file (for diagnostic purposes only)
SHOW-STATUS	Output the name of the opened syntax file

11.1.2 Description of the statements

COPY

Copy contents of syntax file

The COPY statement copies the contents of a syntax file. SDF-U inserts the copies into the opened syntax file. The opened syntax file and the syntax file whose contents are copied must be of the same type.

COPY
<p>OBJECT = *DOMAIN(...) / *COMMAND(...) / *PROGRAM(...)</p> <p>*DOMAIN(...)</p> <ul style="list-style-type: none"> NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *ALL(...) <ul style="list-style-type: none"> EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> <p>*COMMAND(...)</p> <ul style="list-style-type: none"> NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *ALL(...) <ul style="list-style-type: none"> EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> <p>*PROGRAM(...)</p> <ul style="list-style-type: none"> NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *ALL(...) <ul style="list-style-type: none"> EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> <p>,FROM-FILE = <filename 1..54 without-gen-vers></p> <p>,ATTACHED-INFO = *YES / *NO / *ONLY</p> <p>,OVERWRITE-POSSIBLE = *NO / *EXTERNAL-ATTRIBUTES / *YES</p>

OBJECT =

Type of object whose definition is to be copied.

OBJECT = *DOMAIN (...)

The definitions of domains will be copied.

NAME = *ALL(...)

The definitions of all domains will be copied.

EXCEPT = *NONE / <structured-name 1..30 with-wild> /**list-poss(2000): <structured-name 1..30>**

The definitions of the application areas specified here will not be copied.

NAME = <structured-name 1..30 with-wild> /**list-poss(2000): <structured-name 1..30>**

The definitions of the named domains, or of those whose names match the wildcard search criteria, will be copied.

OBJECT = *COMMAND (...)

The definitions of commands will be copied.

NAME = *ALL(...)

The definitions of all commands will be copied.

EXCEPT = *NONE / <structured-name 1..30 with-wild> /**list-poss(2000): <structured-name 1..30>**

The definitions of the commands specified here will not be copied.

NAME = <structured-name 1..30 with-wild> /**list-poss(2000): <structured-name 1..30>**

The definitions of the named commands, or of those whose names match the wildcard search criteria, will be copied.

OBJECT = *PROGRAM (...)

The definitions of programs will be copied.

NAME = *ALL(...)

The definitions of all programs will be copied.

EXCEPT = *NONE / <structured-name 1..30 with-wild> /**list-poss(2000): <structured-name 1..30>**

The definitions of the programs specified here will not be copied.

NAME = <structured-name 1..30 with-wild> /**list-poss(2000): <structured-name 1..30>**

The definitions of the named programs, or of those whose names match the wildcard search criteria, will be copied.

FROM-FILE = <filename 1..54>

Syntax file containing the definitions to be copied.

ATTACHED-INFO =

Determines which of the definitions belonging to the specified object will be copied.

ATTACHED-INFO = *YES

The definition of the specified object will be copied together with the definitions of all objects assigned to the specified object. (In other words: domain with associated commands, program with associated statements, command or statement with associated operands.)

ATTACHED-INFO = *NO

The definition of the specified object will be copied without the definitions of the objects assigned to the specified object. (In other words: domain without associated commands, program without associated statements, command or statement without associated operands.)

ATTACHED-INFO = *ONLY

Only the definitions of the objects assigned to the specified object will be copied. The definition of the specified object itself will not be copied.

OVERWRITE-POSSIBLE =

Determines whether an object already defined in the opened syntax file will be copied.

OVERWRITE-POSSIBLE = *NO

SDF-U rejects the copy request if the object is already defined in the opened syntax file, and issues a message to this effect.

OVERWRITE-POSSIBLE = *EXTERNAL-ATTRIBUTES

SDF-U only copies the objects and not the definition of the objects. The definition of the "current" object is retained. This operand may be specified only for copying domains and programs (COPY OBJ=*DOMAIN... or OBJ=*PROGRAM...). The operand ATTACHED-INFO is given the value *NO, regardless of whether another value has already been specified.

OVERWRITE-POSSIBLE = *YES

SDF-U will copy the object even if it is already defined in the opened syntax file. If necessary, SDF-U will replace the definition already existing in the opened syntax file by the definition to be copied.

EDIT**Set current position to command in syntax file**

The EDIT statement can be used to declare a command or an operand value as the “current” object. In order to modify the definition of a command with the MODIFY-CMD statement, you must first make sure that the command is made the “current” object.

EDIT
<pre> OBJECT = *COMMAND(...) / *VALUE(...) *COMMAND(...) NAME = <structured-name 1..30> *VALUE(...) OPERAND-L1 = *<u>ABOVE-CURRENT</u> / <structured-name 1..20> ,VALUE-L1 = *<u>CURRENT</u> / <structured-name 1..30> ,ORIGIN = *<u>CURRENT</u> / *COMMAND(...) *COMMAND(...) NAME = <structured-name 1..30> </pre>

OBJECT = *COMMAND (...)

A command becomes the current object.

NAME = <structured-name 1..30

Name of the command.

OBJECT=*VALUE(...)

An operand value becomes the current object.

OPERAND-L1 = *ABOVE-CURRENT / <structured-name 1..20>

Specifies the operand to which the operand value that is to be declared as the current object belongs. *ABOVE-CURRENT means that a value belonging to OPERAND-L1 is the current object. <structured-name 1..30> must be a globally unique operand name within the command.

VALUE-L1= *CURRENT / <structured-name 1..30>

Specifies the operand value that is to be declared as the current object. *CURRENT means that VALUE-L1 is the current object. If VALUE-L1 is not the current object and is of the type *KEYWORD, then the particular value defined for it must be specified. Note that this particular value must always be specified without the prefixed asterisk. If the operand value is not of the type *KEYWORD, then the data type defined for it must be specified.

ORIGIN =

Selects the command in which the specified operand value becomes the current object.

ORIGIN = *CURRENT

The operand value belongs to a command which is itself the current object or which contains an operand or operand value that is the current object.

ORIGIN = *COMMAND(...)

The operand value belongs to the command specified under NAME.

NAME = <structured-name 1..30>

Name of the command.

END
Terminate program execution

The END statement terminates input to SDF-U and closes the last opened syntax file.

END

This statement has no operands.

MODIFY-CMD

Modify command definition

The MODIFY-CMD statement is used to modify the definition of a command in the syntax file being processed. This command must be the “current” object.

MODIFY-CMD

```

IMPLEMENTOR = *UNCHANGED / *PROCEDURE(...)
  *PROCEDURE(...)
    | NAME = *UNCHANGED / <c-string 1..54>

```

IMPLEMENTOR =

Type of command implementation.

IMPLEMENTOR = *UNCHANGED

No modification related to command implementation.

IMPLEMENTOR = ***PROCEDURE**(...)

The command is implemented by means of a command procedure. When the command is entered, the command procedure is called.

NAME = *UNCHANGED / <c-string 1..54>

Name of the file containing the procedure. Members of libraries can also be specified in the form 'library(member)'.

MODIFY-VALUE**Modify operand value definition**

The MODIFY-VALUE statement is used to modify the definition of an operand value in the syntax file being processed. The operand value must be the “current” object.

MODIFY-VALUE

```

VALUE = *UNCHANGED / list-poss(2000): <c-string 1..1800 with-low>(…)
    <c-string 1..1800 with-low>(…)
    | OUTPUT = *UNCHANGED / <c-string 1..1800>

```

VALUE =

Specifies the types of values allowed as input.

VALUE = *UNCHANGED

No changes are made with respect to the allowed input values.

VALUE = list-poss(2000): <c-string 1..1800 with-low>(…)

The value for the operand must be one of the specified values (mandatory for values of the type KEYWORD). If no list is specified here, the value given can be abbreviated on input by the user in contrast to the STANDARD-NAME and ALIAS-NAME. If the operand value is of the type KEYWORD, the specification of a list is not allowed.

OUTPUT =

Defines which value is passed on to the implementation.

OUTPUT = *UNCHANGED

No changes are made with respect to the passed value.

OUTPUT = <c-string 1..1800>

The value specified here is passed.

OPEN-SYNTAX-FILE

Open syntax file

The OPEN-SYNTAX-FILE statement opens a system or group syntax file for processing with SDF-U. It is the first statement (except for standard statements) that must be entered after calling SDF-U. Each subsequent OPEN-SYNTAX-FILE statement implicitly causes SDF-U to close the previously opened syntax file.

OPEN-SYNTAX-FILE

```

FILE = <filename 1..54>
, TYPE = *GROUP(...) / *SYSTEM
    *GROUP(...)
        | SYSTEM-DESCRIPTION = *CURRENT / *NO
, MODE = *UPDATE / *READ

```

FILE = <filename 1..54>

Name of the syntax file to be opened.

TYPE =

Type of syntax file to be opened.

TYPE = *GROUP(...)

A group syntax file is to be opened.

SYSTEM-DESCRIPTION = *CURRENT / *NO

Specifies whether SDF-U is to access the currently active system syntax file.

TYPE = *SYSTEM

A system syntax file is to be opened.

MODE =

Defines how the opened syntax file is to be processed.

MODE = *UPDATE

The contents of the syntax file may be both output and updated. The syntax file already exists. It must not be activated.

MODE = *READ

The contents of the syntax file may only be output but not updated (read-only access). The syntax file already exists. It may be active.

REMOVE

Delete objects from syntax file

The REMOVE statement deletes objects (i.e. domains, programs and commands) from the processed syntax file.

REMOVE
<p>OBJECT = *DOMAIN(...) / *COMMAND(...) / *PROGRAM(...)</p> <p>*DOMAIN(...)</p> <p> NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30></p> <p> *ALL(...)</p> <p> EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30></p> <p>*COMMAND(...)</p> <p> NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30></p> <p> *ALL(...)</p> <p> EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30></p> <p>*PROGRAM(...)</p> <p> NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30></p> <p> *ALL(...)</p> <p> EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30></p>

OBJECT =

Type of object to be deleted.

OBJECT = *DOMAIN (...)

Domains will be deleted. The commands assigned to these domains will be deleted, provided that they have not been assigned to at least one other domain, or have been defined by means of REMOVE-POSSIBLE = *NO.

NAME = *ALL(...)

All domains will be deleted.

EXCEPT = *NONE / <structured-name 1..30 with-wild> /
list-poss(2000): <structured-name 1..30>

The domains specified here will not be deleted.

NAME = <structured-name 1..30 with-wild> /

list-poss(2000): <structured-name 1..30>

The named domains, or those whose names match the wildcard search criteria, will be deleted.

OBJECT = *COMMAND(...)

Commands will be deleted. The associated operands will also be deleted.

NAME = *ALL(...)

All commands will be deleted.

EXCEPT = *NONE / <structured-name 1..30 with-wild> /

list-poss(2000): <structured-name 1..30>

The commands specified here will not be deleted.

NAME = <structured-name 1..30 with-wild> /

list-poss(2000): <structured-name 1..30>

The named commands, or those whose names match the wildcard search criteria, will be deleted.

OBJECT = *PROGRAM(...)

Programs will be deleted. The associated statements will also be deleted.

NAME = *ALL(...)

All programs will be deleted.

EXCEPT = *NONE / <structured-name 1..30 with-wild> /

list-poss(2000): <structured-name 1..30>

The programs specified here will not be deleted.

NAME = <structured-name 1..30 with-wild> /

list-poss(2000): <structured-name 1..30>

The named programs, or those whose names match the wildcard search criteria, will be deleted.

SET-GLOBALS

Modify global information

The SET-GLOBALS statement changes general definitions relating to command/ statement input and processing. These definitions are contained in the global information and become effective as soon as the syntax file is activated.

SET-GLOBALS

```

VERSION = *UNCHANGED / <alphanum-name 1..12> / <c-string 1..12>
, GUIDANCE = *UNCHANGED / *STD / *MAXIMUM / *MEDIUM / *MINIMUM / *NO / *EXPERT
, LOGGING = *UNCHANGED / *STD / *INPUT-FORM / *ACCEPTED-FORM / *INVARIANT-FORM
, PROCEDURE-DIALOGUE = *UNCHANGED / *STD / *YES / *NO
, UTILITY-INTERFACE = *UNCHANGED / *STD / *OLD-MODE / *NEW-MODE
, CONTINUATION = *UNCHANGED / *STD / *OLD-MODE / *NEW-MODE
, FUNCTION-KEYS = *UNCHANGED / *STD / *OLD-MODE / *STYLE-GUIDE-MODE / *BY-TERMINAL-TYPE
, INPUT-HISTORY = *UNCHANGED / *STD / *ON / *OFF
, NUMBER-OF-INPUTS = *UNCHANGED / *STD / <integer 1..100>

```

VERSION =

Defines the version number of the syntax file. It is used for documentation purposes only.

VERSION = *UNCHANGED

The version number remains unchanged.

VERSION = <alphanum-name 1..12> / <c-string 1..12>

The syntax file is given the specified version number. Blanks at the end of a c-string are ignored.

GUIDANCE =

Determines the degree of dialog guidance. (The definition does not apply to jobs which were started via OMNIS; *EXPERT is assumed in such cases.)

GUIDANCE = *UNCHANGED

The current dialog guidance specification in the global information remains valid.

GUIDANCE = *STD

Dialog guidance remains unchanged when the processed group syntax file is activated. For a system syntax file, *STD has the same effect as *NO.

GUIDANCE = *MAXIMUM

Menus with explanatory texts are displayed for the purpose of selecting domains, commands and statements. Forms are displayed for the purpose of operand specification. Each structure has its own forms. These forms contain help texts for the operands, the default values, all permissible operand entries and additional information on these entries.

GUIDANCE = *MEDIUM

Menus with explanatory texts are displayed for the purpose of selecting domains, commands and statements. Forms are displayed for the purpose of operand specification. Each structure whose initial value is defined with SIZE=LARGE (see the ADD-VALUE statement in the "SDF-A" manual [4]) has its own forms. These forms contain the default values and all permissible operand values.

GUIDANCE = *MINIMUM

Menus are displayed for the purpose of selecting domains, commands and statements. Forms are displayed for the purpose of operand specification. Each structure whose initial value is defined with SIZE=*LARGE (see ADD-VALUE in the "SDF-A" manual [4]) has its own forms. These forms contain only the default values.

GUIDANCE = *NO

Input is requested with the prompt %CMD: (%KDO:) or %STMT: (%ANW:). Two or more commands may be entered in succession in a block, the commands being separated by a "logical end of line". Correction options are offered if the entry contains an error.

GUIDANCE = *EXPERT

Input is requested with / or %//. Two or more commands may be entered in succession in a block, the commands being separated by a "logical end of line". Correction options are not offered if the entry contains an error.

LOGGING =

Determines how input is to be logged.

LOGGING = *UNCHANGED

The current logging specification in the global information remains valid.

LOGGING = *STD

Logging remains the same when the processed group syntax file is activated. For a system syntax file, *STD has the same effect as *INPUT-FORM.

LOGGING = *INPUT-FORM

In unguided dialog, input character strings are logged exactly as entered. Passwords are blanked out. In guided dialog or error dialog, logging takes place as with *ACCEPTED-FORM.

LOGGING = *ACCEPTED-FORM

The following are logged:

- all names in their unabbreviated form
- each input operand with its name and the specified value
- any updated versions resulting from corrections.

Passwords are blanked out on the screen. Entries made in guided dialog are concatenated to form a string.

LOGGING = *INVARIANT-FORM

The following are logged:

- all names in the form in which they are defined in the syntax file as STANDARD-NAME (i.e. the names specified in the manuals)
- each operand occurring in the entry, with its name and specified value
- all optional operands implicitly contained in the entry, with their default values
- any updated versions resulting from a correction dialog.

Passwords are blanked out. Entries made in guided dialog are concatenated to form a string.

PROCEDURE-DIALOGUE =

Determines whether the user is to be requested to correct errors interactively whenever syntax or semantic errors occur in connection with a procedure or SYSSTMT file in interactive mode.

PROCEDURE-DIALOGUE = *UNCHANGED

The current specification in the global information regarding interactive correction remains valid.

PROCEDURE-DIALOGUE = *STD

The regulation governing interactive correction remains unchanged when the processed group syntax file is activated. For a system syntax file, *STD has the same effect as *NO.

PROCEDURE-DIALOGUE = *YES

The user is requested to correct errors interactively.

PROCEDURE-DIALOGUE = *NO

The user is not request to correct errors interactively.

UTILITY-INTERFACE =

Sets a switch whose value can be interrogated by a program via the CMDSTA macro. This switch makes it possible to control the type of statement input for programs which can read their statements both with RDATA and via SDF with RDSTMT.

UTILITY-INTERFACE = *UNCHANGED

The current switch specification in the global information remains valid.

UTILITY-INTERFACE = *STD

The switch remains unchanged when the processed group syntax file is activated. For a system syntax file, *STD has the same effect as *NEW.

UTILITY-INTERFACE = *OLD-MODE

Programs are to read their statements with RDATA.

UTILITY-INTERFACE = *NEW-MODE

Programs are to read their statements via SDF with RDSTMT.

CONTINUATION =

Determines the column for command input (SYSCMD) in which the continuation character “-” is to be specified. For statement input (SYSSTMT) the continuation character can be specified in any column.

CONTINUATION = *UNCHANGED

The current continuation character specification in the global information remains valid.

CONTINUATION = *STD

The regulation governing the continuation character remains unchanged when the processed group syntax file is activated. For a system syntax file, *STD defaults to the specification made at system generation.

CONTINUATION = *OLD-MODE

The continuation character must be entered in column 72.

CONTINUATION = *NEW-MODE

The continuation character may be entered in any column from 2 through 72.

FUNCTION-KEYS =

Defines function key assignments. A detailed description of the different modes can be found in the [section “Function keys” on page 47](#). Unsupported function keys do nothing; they do not have the same effect as the `[DUE]` key.

FUNCTION-KEYS = *UNCHANGED

The function key assignments defined in the global information are not changed.

FUNCTION-KEYS = *STD

The existing setting for the option is retained when the processed group syntax file is activated. For a system syntax file, *STD has the same effect as *BY-TERMINAL-TYPE.

FUNCTION-KEYS = *OLD-MODE

Function keys assignments correspond to the old mode, which is supported by all terminal types. The following key assignments apply:

- `[K1]` Exit function
- `[K2]` Interrupt function
- `[K3]` Refresh function (only in guided dialog)
- `[F1]` Exit-all function
- `[F2]` Test function (only in guided dialog)
- `[F3]` Execute function (only in guided dialog)

FUNCTION-KEYS = *STYLE-GUIDE-MODE

Function keys are assigned in accordance with the style guide. The following key assignments apply:

- K2** Interrupt function
- F1** Help function
- F3** Exit function
- F5** Refresh function (only in guided dialog)
- F6** Exit-all function
- F7** Scroll backward (only in guided dialog)
- F8** Scroll forward (only in guided dialog)
- F9** Execute RESTORE-SDF-INPUT INPUT=*LAST
- F11** Execute function (only in guided dialog)
- F12** Cancel function

FUNCTION-KEYS = *BY-TERMINAL-TYPE

The assignment of function keys depends on the type of terminal. If the terminal type supports the more comprehensive functionality of the style guide, SDF selects the *STYLE-GUIDE-MODE setting; otherwise, it selects the *OLD-MODE.



The terminal type with which the terminal or terminal emulation was generated in the system is evaluated for this setting. If the generated terminal type differs from the actual terminal type, there is no guarantee that the setting will reflect the actually supported functionality. In the case of an emulation, the recognized terminal type will depend on the generation as well as the environment variables. See the description of the emulation program for more details.

INPUT-HISTORY =

Specifies whether the input buffer is to be turned on, turned off, or reset.

INPUT-HISTORY = *UNCHANGED

The currently valid global information setting for saving inputs is not changed.

INPUT-HISTORY = *STD

The existing setting for the option is retained when the processed group syntax file is activated. In the case of a system syntax file, *STD has the same effect as *ON.

INPUT-HISTORY = *ON

The input buffer is turned on, and SDF saves all syntactically correct inputs in it. SET-LOGON-PARAMETERS, RESTORE-SDF-INPUT and SHOW-INPUT-HISTORY are not saved.

Whether ISP commands are saved depends on the entry in the PASSWORD-PROTECTION operand (command/statement MODIFY-SDF-OPTIONS).

The user can output the saved inputs by means of the SHOW-INPUT-HISTORY statement. The RESTORE-SDF-INPUT command can be used to retrieve a particular input and then repeat it with or without modifications.



Values which are specified for “secret” operands and which correspond to neither the default value nor a value defined with SECRET=*NO are saved in the input buffer as a “^” or in plain text, depending on what is specified for the PASSWORD-PROTECTION operand.

Values specified for operands not defined as secret, by contrast, are saved as plain text. In some cases, such information (e.g. procedure parameters) may also be worth protecting from a user’s viewpoint. To prevent such inputs from being redisplayed on the screen by SHOW-INPUT-HISTORY or RESTORE-SDF-INPUT, the user can turn off the input buffer (i.e. the history feature) before making entries for which security is required and then turn it on again. Alternatively, if the inputs have already been saved, the input buffer can be purged with *RESET, in which case all saved inputs will be deleted.

INPUT-HISTORY = *OFF

The input buffer is turned off. Subsequent inputs are not stored; but inputs saved earlier remain accessible.

NUMBER-OF-INPUTS = *UNCHANGED / *STD / <integer 1..100>

Part of the INPUT-HISTORY operand; defines how many inputs are to be saved in the input buffer. The maximum possible number is 100. When the maximum number of inputs to be saved is reached, the buffer is cycled, i.e. the oldest input is deleted whenever a new input is saved.

SHOW

Output objects of syntax file

The SHOW statement can be used to output the contents of a syntax file to SYSOUT or SYSLST. The output can be interrupted, restarted, or aborted with the **[K2]** key.

(part 1 of 2)

SHOW
<pre> OBJECT = *GLOBAL-INFORMATION / *DOMAIN(...) / *COMMAND(...) / *PROGRAM(...) *DOMAIN(...) NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *ALL(...) EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *COMMAND(...) NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *ALL(...) EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *PROGRAM(...) NAME = *ALL(...) / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> *ALL(...) EXCEPT = *NONE / <structured-name 1..30 with-wild> / list-poss(2000): <structured-name 1..30> ,ATTACHED-INFORMATION = *YES / *NO / *IMMEDIATE ,SIZE = *MINIMUM / *MAXIMUM / *MEDIUM ,IMPLEMENTATION-INFO = *NO (...) *NO(...) FORM = *UNGUIDED / *GUIDED ,LANGUAGE = E / <name 1..1> </pre>

continued ➡

(part 2 of 2)

```
,LINES-PER-PAGE = *STD / *UNLIMITED(...) / <integer 1..200>
  *UNLIMITED(...)
    | OUTPUT-FORM = *STD / *FOR-INPUT
,OUTPUT = *SYSOUT / *SYSLST(...)
  *SYSLST(...)
    | SYSLST-NUMBER = *STD / <integer 1..99>
,LINE-LENGTH = *STD / <integer 72..132>
,PRIVILEGE = *ANY / list-poss(64): <structured-name 1..30>
```

OBJECT =

Type of object whose definition is to be output.

OBJECT = *GLOBAL-INFORMATION

The global information of the syntax file will be output.

OBJECT = *DOMAIN(...)

The definitions of domains will be output.

NAME = *ALL(...)

The definitions of all domains will be output.

```
EXCEPT = *NONE / <structured-name 1..30 with-wild> /
list-poss(2000): <structured-name 1..30>
```

The definitions of domains specified here will not be output.

NAME = <structured-name 1..30 with-wild> /

```
list-poss(2000): <structured-name 1..30>
```

The definitions of the named domain, or of those whose names match the wildcard search criteria, will be output.

OBJECT = *COMMAND(...)

The definitions of commands will be output.

NAME = *ALL(...)

The definitions of all commands will be output.

```
EXCEPT = *NONE / <structured-name 1..30 with-wild> /
list-poss(2000): <structured-name 1..30>
```

The definitions of commands specified here will not be output.

NAME = <structured-name 1..30 with-wild> /

```
list-poss(2000): <structured-name 1..30>
```

The definitions of the named commands, or of those whose names match the wildcard search criteria, will be output.

OBJECT = *PROGRAM(...)

The definitions of programs will be output.

NAME = *ALL(...)

The definitions of all programs will be output.

**EXCEPT = *NONE / <structured-name 1..30 with-wild> /
list-poss(2000): <structured-name 1..30>**

The definitions of programs specified here will not be output.

**NAME = <structured-name 1..30 with-wild> /
list-poss(2000): <structured-name 1..30>**

The definitions of the named programs, or of those whose names match the wildcard search criteria, will be output.

ATTACHED-INFORMATION =

Determines which of the definitions belonging to the specified object will be output.

ATTACHED-INFORMATION = *YES

The definition of the specified object will be output together with the definitions of all objects assigned to the specified object (in other words: domain with associated commands, program with associated statements, command with associated operands).

ATTACHED-INFORMATION = *NO

The definition of the specified object will be output without the definitions of the objects assigned to the specified object (in other words: domain without associated commands, program without associated statements, command without associated operands).

ATTACHED-INFORMATION = *IMMEDIATE

The definition of the specified object will be output together with the definitions of the objects immediately assigned to the specified object, i.e. application areas and programs are output with the associated commands and statements respectively, but without the associated operands and operand values. In the case of commands, *IMMEDIATE has the same effect as *YES.

SIZE =

Determines the scope of the output. The exact effect of the SIZE operand depends on the OBJECT operand. The SIZE operand has no effect on the output of global information. The following information is output for OBJECT=*COMMAND:

SIZE = *MINIMUM

Output will include the command name, all associated operand names, all associated default values, and the structure-initiating operand values. Additional information on the operand values and help texts will be omitted.

SIZE = *MAXIMUM

Output will include the command name, all associated operand names, all associated default values, and all other associated operand values. Additional information on the operand values and help texts will also be output.

SIZE = *MEDIUM

Output will include the command name, all associated operand names, all associated default values, and all other associated operand values. Additional information on the operand values and help texts will be omitted.

IMPLEMENTATION-INFO =

Determines how the output is edited.

IMPLEMENTATION-INFO = *NO(...)

The definitions of the specified objects are output in a manual-oriented layout. The operands with the value PRESENCE=*INTERNAL-ONLY are not listed here.

FORM =

Determines whether output is to include definitions of objects which are prohibited for guided dialog.

FORM = *UNGUIDED

The definitions will be included.

FORM = *GUIDED

The definitions will not be included.

LANGUAGE = E / <name 1..1>

Determines the language in which help texts are output (E=English, D=German). This operand has no effect for global information.

LINES-PER-PAGE = *STD / *UNLIMITED(...) <integer 1..200>

Determines the number of lines per page. The count does not include the two header lines that SDF-U generates for each new page. The header is only generated if OUTPUT=*SYSLST is specified.

LINES-PER-PAGE = *STD

The default value is 24 lines for screen output and 55 lines for output to a file.

LINES-PER-PAGE = *UNLIMITED(...)

No check by SDF-U (no header is created).

OUTPUT-FORM=

Defines which characters are output at the beginning of the line.

OUTPUT-FORM=*STD

The first character in each line is a blank.

OUTPUT-FORM=*FOR-INPUT

Two slashes (//) are output at the start of each line.

This entry can be used together with IMPLEMENTATION=*YES to create SDF-U statements which can be used to restore a syntax file or a syntax file object.

OUTPUT =

Determines the output medium for the desired information.

OUTPUT = *SYSOT

Output is directed to the logical system file, which is usually the screen in interactive mode.

OUTPUT = *SYSLST(...)

Output is directed to the logical system file SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>

Determines the number of the logical system file SYSLST. If *STD is specified, the logical system file SYSLST receives no number.

LINE-LENGTH = *STD / <integer 72..132>

Determines the output line length.

LINE-LENGTH = *STD

The default value is 74 characters for screen output and 72 characters for output to a file.

PRIVILEGE = *ANY / list-poss(64): <structured-name 1..30>

Only objects to which at least one of the privileges that appear in the list is assigned are output. If *ANY is specified, the objects are listed irrespective of their privileges.

SHOW-CORRECTION-INFORMATION

Output syntax file correction information

The SHOW-CORRECTION-INFORMATION statement outputs information on the corrections in the syntax file.

The statement is only provided for diagnostic purposes.

SHOW-CORRECTION-INFORMATION
<pre> CORRECTION-ID = *SOURCE (...) / *OBJECT(...) *SOURCE(...) PM-NUMBER = *ALL / list-poss(100): <alphanum-name 8..8> *OBJECT(...) PM-NUMBER = *ALL / <alphanum-name 8..8> ,JULIAN-DATE = *ANY / <integer 1..366> ,PRODUCT-NAME = *ANY / <structured-name 1..15>(…) <structured-name 1..15>(…) VERSION = *ANY / <product-version> </pre>

The operands are not described here as the statement is only provided for diagnostic purposes.

SHOW-STATUS

Display status of opened syntax file

The SHOW-STATUS statement causes SDF-U to output the name of the currently open syntax file and information on one or more reference syntax files assigned to it.

SHOW-STATUS

This statement has no operands.

STEP

Define checkpoint

The STEP statement is used to define a checkpoint for error handling in a sequence of SDF-U statements. This statement is valid only in procedures and batch runs.

STEP

The STEP statement has no operands.

If SDF-U detects a syntax error or a serious logical error, the following steps are initiated:

- an error message is output
- the running SDF-U statement is aborted
- the subsequent statements are skipped until STEP or END is reached.

If SDF-U reaches an END statement first, it generates an abnormal program termination (TERM UNIT=STEP,MODE=ABNORMAL) and activates the spin-off mechanism. If SDF-U reaches a STEP statement first, it continues with the statement following the STEP. If the error has no influence on correct execution of the running job, the SDF-U user must preempt abnormal program termination by inserting a STEP.

SDF-U corrects minor logical errors automatically and a warning message is output. The spin-off mechanism is not activated in such cases.

12 SDF-PAR

Version: SDF-PAR V1.1

Privileges: TSOS

SDF-PAR is a utility that can be used to create, modify and display a parameter file for the product SDF. It thus offers an alternative to the privileged commands MODIFY-SDF-PARAMETERS and SHOW-SDF-PARAMETERS described in the “Commands” manual [1]. One of the advantages of SDF-PAR is that the parameter file can be processed off-line.

The SDF-PAR program can be used as of SDF V1.4. The statements issued to SDF-PAR are defined in a user syntax file that is supplied with SDF-PAR.

Starting SDF-PAR

The following procedure, which is in the scope of delivery, is to be called to start SDF-PAR in BS2000/OSD:

```
/CALL-PROCEDURE $.SYSPRC.SDF-PAR.011.112
```

The procedure contains the necessary preparations for the program call (the SDF-PAR message and syntax file are activated) in addition to the actual SDF-PAR call.



Before starting SDF-PAR, the SDF-PAR message file must be assigned, and the SDF-PAR user syntax file must be activated. All required preparations for the program run are also executed by the SYSPRC procedure supplied with delivery.

Since no START-SDF-PAR command is included in the supplied software package, the SYSPRG file has to be started with the START-EXECUTABLE-PROGRAM (or START-PROGRAM) command.

Further notes on the installation of SDF-PAR can be found the release notice of the software product SDF.

12.1 SDF-PAR statements

12.1.1 Overview of functions

Creating, processing and displaying the SDF parameter file

END	Terminate SDF-PAR
OPEN-PARAMETER-FILE	Create SDF parameter file or open existing file for processing
MODIFY-SYNTAX-FILE	Modify entries for syntax files
MODIFY-SYSTEM-LOGON- INCLUDE	Modify entry for global LOGON include procedure
MODIFY-SYSTEM-LOGON- PROCEDURE	Modify entry for global LOGON call procedure
MODIFY-VERSION	Modify format of parameter file
SHOW-PARAMETER-FILE	Display contents of opened SDF parameter file

SDF standard statements

Detailed descriptions of these statements can be found in the [section “Standard statements” on page 146](#).

END	Terminate SDF-PAR
EXECUTE-SYSTEM-CMD	Execute system command while SDF-U is running
HELP-MSG-INFORMATION	Output system message to SYSOUT
HOLD-PROGRAM	Interrupt execution of SDF-PAR
MODIFY-SDF-OPTIONS	Activate or deactivate user syntax file and change SDF settings
REMARK	Write comments in statement sequence and document execution of program
RESET-INPUT-DEFAULTS	Reset task-specific default values
RESTORE-SDF-INPUT	Redisplay previously entered statements or commands on screen
SHOW-INPUT-DEFAULTS	Display task-specific default values
SHOW-INPUT-HISTORY	Display contents of input buffer

SHOW-SDF-OPTIONS	Display task-specific information on activated syntax files and options set for input and processing of statements
SHOW-STMT	Display the syntax of a statement
STEP	Define restart point in sequence of SDF-PAR statements
WRITE-TEXT	Output specific text to SYSOUT or SYSLST

12.1.2 Description of the statements

MODIFY-SYNTAX-FILE

Modify entries for syntax files

The MODIFY-SYNTAX-FILE statement is used to enter syntax file names in the parameter file or to modify or remove existing syntax file entries. The following entries are possible:

- **System syntax file:**
Every parameter file must contain one entry for the basic system syntax file. This entry may be modified, but cannot be removed. The basic system syntax file contains the syntax description of all syntax commands and applies to all currently logged-on user tasks.
- **Subsystem syntax file:**
Entries for subsystem syntax files are optional. Subsystem syntax files are system syntax files that must be activated in addition to the basic syntax file. Such system syntax files need not be part of a subsystem activated by DSSM.
- **Group syntax files:**
Entries for group syntax files are optional. A group syntax file contains the syntax description of all commands, programs and statements that are available exclusively to a group of users defined by system administration. Group syntax files are assigned via a profile ID.

MODIFY-SYNTAX-FILE

TYPE = *SYSTEM / *SUBSYSTEM(...) / *GROUP(...)

*SUBSYSTEM(...)

 | **SUBSYSTEM-NAME** = <structured-name 1..8>

*GROUP(...)

 | **PROFILE-ID** = <structured-name 1..30>

 | **,HIERARCHY** = *YES / *NO

,NAME = *UNCHANGED / *NONE / <filename 1..54 without-gen-vers>

TYPE =

Defines the type of the syntax file for which the entry is modified.

TYPE = *SYSTEM

The entry for the basic system syntax file is modified.

TYPE = *SUBSYSTEM(...)

The entry for a subsystem syntax file is modified.

SUBSYSTEM-NAME = <structured-name 1..8>

Name of the subsystem to which the subsystem syntax file belongs.

TYPE = *GROUP(...)

The entry for a group syntax file is modified.

PROFILE-ID = <structured-name 1..30>

Defines the profile ID to which the group syntax file was or is to be assigned.

HIERARCHY =

Specifies whether the SDF file hierarchy is to be retained for the syntax analysis of commands/statements of a user task with the specified profile ID, i.e. whether the system syntax files are to be used for syntax analysis.

HIERARCHY = *YES

The system syntax files are activated by default on creating the user task. The SDF file hierarchy is retained.

HIERARCHY = *NO

The system syntax files are deactivated immediately after LOGON processing. The definitions in the system syntax file are therefore irrelevant for users with the specified profile ID; only the definitions stored in the associated group syntax file apply. Group syntax files defined with HIERARCHY=*NO **must** contain at least the LOGOFF command in addition to global information, since this is the only way to terminate a user task to which the defined profile ID is assigned.

NAME =

Specifies the name of the syntax file. The type of the specified syntax file must be the same as the one defined by the TYPE entry.

NAME = *UNCHANGED

The entry is not changed.

NAME = *NONE

The syntax file entry is deleted from the parameter file. The entry for the system syntax file cannot be deleted.

NAME = <filename 1..54 without-gen-vers>

Specifies the name of the syntax file.

MODIFY-SYSTEM-LOGON-PROCEDURE

Modify entry for global LOGON call procedure

The MODIFY-SYSTEM-LOGON-PROCEDURE statement is used to enter the name of a global LOGON call procedure in the currently open parameter file. This procedure will then be called and executed automatically at every LOGON with the CALL-PROCEDURE command.

MODIFY-SYSTEM-LOGON-PROCEDURE
NAME = <u>*UNCHANGED</u> / *NONE / *STD / <filename 1..54 without-gen-vers>

NAME = *UNCHANGED / *NONE / *STD / <filename 1..54 without-gen-vers>

Defines whether a global LOGON call procedure is to be entered in the parameter file. This entry is optional.

NAME = *NONE

Deletes the entry for a global LOGON call procedure from the parameter file.

NAME = *STD

The default name '\$.SYS.SDF.LOGON.SYSPROC' is entered.

NAME = <filename 1..54 without-gen-vers>

Name of the new global LOGON call procedure.

MODIFY-SYSTEM-LOGON-INCLUDE

Modify entry for global LOGON include procedure

The MODIFY-SYSTEM-LOGON-INCLUDE statement is used to enter the name of a global LOGON include procedure in the currently open parameter file. This procedure will then be called and executed automatically at every LOGON with the INCLUDE-PROCEDURE command.

MODIFY-SYSTEM-LOGON-INCLUDE
NAME = <u>*UNCHANGED</u> / *NONE / *STD / <filename 1..54 without-gen-vers>

NAME = *UNCHANGED / *NONE / *STD / <filename 1..54 without-gen-vers>

Defines whether a global LOGON include procedure is to be entered in the parameter file. This entry is optional.

NAME = *NONE

Deletes the entry for a global LOGON include procedure from the parameter file.

NAME = *STD

The default name '\$.SYS.SDF.LOGON.SYSINCL' is entered.

NAME = <filename 1..54 without-gen-vers>

Name of the new global LOGON include procedure.

MODIFY-VERSION

Modify format of parameter file

The MODIFY-VERSION statement is used to modify the format of the open parameter file. The format of the parameter file is defined when the file is created by means of the OPEN-PARAMETER-FILE statement.



Format *V2 is the format which is currently supported for parameter files. Parameter files in format *V1 cannot be used in the current SDF versions. Consequently only conversion to format *V2 makes sense.

MODIFY-VERSION
VERSION = *<u>UNCHANGED</u> / *V2 / *V1

VERSION = *UNCHANGED / *V2 / *V1

Defines the format of the parameter file.

VERSION = *V2

The parameter file is converted to the currently supported *V2 format.

When a *V1 format is converted to *V2 format, the record length of the parameter file is extended to 67 bytes. The additional bytes are padded with X'00'. Records for group syntax file entries are extended with 'YES' and ten X'00's.

Every object of the parameter file is reconstructed and reinserted into the open parameter file, and a message is output for each such object.

VERSION = *V1

The operand value is supported for compatibility reasons only.

Parameter files with *V1 format can not be used in the current SDF version (conversion truncates the records to 54 bytes).

OPEN-PARAMETER-FILE

Create or open parameter file

The OPEN-PARAMETER-FILE statement is used to open an existing parameter file or to create a new one. The format of the parameter file is defined at the time it is created.



Format *V2 is the format which is currently supported for parameter files. Parameter files in format *V1 cannot be used in the current SDF versions. Consequently only format *V2 makes sense when creating a parameter file.

The names of syntax files can also be entered at the time of creating the parameter file. A parameter file in *V2 format is created by default, and the name '\$.SYS.SDF.SYSTEM.SYNTAX' is entered in it as the basic system syntax file.

OPEN-PARAMETER-FILE
<pre> NAME = <filename 1..54 without-gen-vers> , MODE = <u>*READ</u> / *UPDATE / *CREATE(...) *CREATE(...) VERSION = <u>*V2</u> / *V1 , SYSTEM-SYNTAX-FILE = <u>*STD</u> / <filename 1..54 without-gen-vers> , GROUP-SYNTAX-FILE = <u>*NONE</u> / *STD(...) / <filename 1..54 without-gen-vers>(…) *STD(...) PROFILE-ID = <u>SYS-TSOS</u> / <structured-name 1..30> , HIERARCHY = <u>*YES</u> / *NO <filename 1..54 without-gen-vers>(…) PROFILE-ID = <u>SYS-TSOS</u> / <structured-name 1..30> , HIERARCHY = <u>*YES</u> / *NO </pre>

NAME = <filename 1..54 without-gen-vers>

Name of the parameter file to be opened or created.

MODE =

Specifies whether the parameter file is to be opened as a read-only file, as a file to be updated, or as a file to be created.

MODE = *READ

The existing parameter file is opened for read access only.

MODE = *UPDATE

The existing parameter file is opened in read/write mode.

MODE = *CREATE(...)

The parameter file is created and opened in read/write mode. If a parameter file with the specified name already exists, the statement is rejected.

VERSION =

Defines the format of the parameter file.

VERSION = *V2

The parameter file is created in the currently supported *V2 format.

VERSION = *V1

The operand value is supported for compatibility reasons only.

Parameter files with *V1 format can not be used in the current SDF version.

SYSTEM-SYNTAX-FILE =

Determines the entry for the basic system syntax file.

SYSTEM-SYNTAX-FILE = *STD

The name '\$.SYS.SDF.SYSTEM.SYNTAX' is entered in the parameter file as the basic system syntax file.

SYSTEM-SYNTAX-FILE = <filename 1..54 without-gen-vers>

Name of the basic syntax file to be entered in the parameter file.

GROUP-SYNTAX-FILE =

Determines the group syntax file entry for the default user ID.

GROUP-SYNTAX-FILE = *NONE

No group syntax file is entered for the default user ID.

GROUP-SYNTAX-FILE = *STD

The group syntax file '\$.SYS.SDF.GROUP.SYNTAX' is entered for the default user ID.

PROFILE-ID = SYS-TSOS / <structured-name 1..30>

Defines the profile ID to which the group syntax file is assigned. The default value is SYS-TSOS, since SDF also uses this value for the user ID TSOS.

HIERARCHY =

Specifies whether the SDF file hierarchy is to be retained for the syntax analysis of commands/statements of a user task with the specified profile ID, i.e. whether the system syntax files are to be used for syntax analysis.

HIERARCHY = *YES

The system syntax files are activated by default on creating the user task. The SDF file hierarchy is retained.

HIERARCHY = *NO

The system syntax files are deactivated immediately after LOGON processing. The definitions in the system syntax files are therefore irrelevant for users with the specified profile ID; only the definitions stored in the associated group syntax file apply. Group

syntax files defined with HIERARCHY=*NO must contain at least the EXIT-JOB (or LOGOFF) command in addition to global information, since this is the only way to terminate a user task to which the defined profile ID is assigned.

GROUP-SYNTAX-FILE = <filename 1..54 without-gen-vers>

The group syntax file with the specified name is entered for the default user ID.

PROFILE-ID = SYS-TSOS / <structured-name 1..30>

Defines the profile ID to which the group syntax file is assigned. The default value is SYS-TSOS, since SDF also uses this value for the user ID TSOS.

HIERARCHY =

Specifies whether the SDF file hierarchy is to be retained for the syntax analysis of commands/statements of a user task with the specified profile ID, i.e. whether the system syntax files are to be used for syntax analysis.

HIERARCHY = *YES

The system syntax files are activated by default on creating the user task. The SDF file hierarchy is retained.

HIERARCHY = *NO

The system syntax files are deactivated immediately after LOGON processing. The definitions in the system syntax file are therefore irrelevant for users with the specified profile ID; only the definitions stored in the associated group syntax file apply. Group syntax files defined with HIERARCHY=*NO **must** contain at least the EXIT-JOB (or LOGOFF) command in addition to global information, since this is the only way to terminate a user task to which the defined profile ID is assigned.

SHOW-PARAMETER-FILE

Display parameter file

The SHOW-PARAMETER-FILE statement displays the contents of the currently open parameter file. The following entries are output if present:

- version format of the parameter file
- name of the entered global LOGON procedure
- name of the entered basic system syntax file
- name of the subsystem syntax file for each entered subsystem
- name of the assigned group syntax file for each entered profile ID. In the case of parameter files in *V2 format, the hierarchy setting is also shown.

SHOW-PARAMETER-FILE

This statement has no operands.

13 SDF-CONV

Version: SDF-CONV V3.0B

Privileges: STD-PROCESSING, TSOS

The SDF-CONV utility routine is able to perform two different types of procedure conversion:

- conversion of the command language (ISP into SDF commands)
- conversion of the procedure format (non-S into S format)

SDF-CONV converts ISP commands and their operands in files and library elements into SDF commands with the appropriate operands. Commands in SDF command records are expanded to their full command names.

Certain ISP commands cannot be converted into SDF format. Such commands are added unchanged to the output procedure. For example, ISP commands are not converted if the selection of the corresponding SDF command is dependent on the runtime behavior or on procedure parameters or the ISP command is no longer supported by BS2000.

All commands and operand configurations that cannot be converted by SDF-CONV are listed in the [section "Restrictions" on page 250](#).

SDF-CONV can be used to convert non-S procedures (ISP and SDF format) into S procedures.

An S procedure ("structured procedure") is structured in accordance with SDF-P rules. SDF-P is a block-oriented programming language. It enhances the command language used in BS2000 up to now by enabling structured programming similar to higher programming languages (see the manual "SDF-P" [\[5\]](#)).

The software product SDF-P includes the following two components:

- the non-chargeable subsystem SDF-P-BASYS containing the basic SDF-P components
- the chargeable subsystem SDF-P offering the full SDF-P functionality

Only the functionality of the non-chargeable SDF-P-BASYS is taken into account when converting a non-S procedure into an S procedure using SDF-CONV (see the "Commands" manual [\[1\]](#)).

The resulting S procedure is compatible but not optimized in comparison with the control structure mechanisms of an S procedure generated with SDF-P.

The following conversions of commands and procedure formats can be performed using SDF-CONV:

non-S procedure with ISP commands	→	non-S procedure with SDF commands
non-S procedure with ISP commands	→	S procedure with ISP commands
non-S procedure with ISP commands	→	S procedure with SDF commands
non-S procedure with SDF commands	→	S procedure with SDF commands
S procedure with ISP commands	→	S procedure with SDF commands

SDF-CONV writes the result of the conversion into an output procedure, which is stored as a SAM file or as an element in a PLAM library. A detailed conversion log is output to SYSLST.

13.1 Starting and terminating the program

The SDF-CONV utility routine is started using the command

```
START-SDF-CONV Alias: SDF-CONV
```

```
VERSION = *STD / <product-version>
,MONJV = *NONE / <filename 1..54 without-gen-vers>
,CPU-LIMIT = *JOB-REST / <integer 1..32767>
,PROGRAM-MODE = *ANY / 24
```

This command calls SDF-CONV as a subsystem. The alternative of starting the SDF-CONV conversion routine by means of the command START-PROGRAM SDF-CONV continues to be supported for reasons of compatibility.

Once the program has been started, the message CVR0060 is displayed.

Every time a conversion run is completed, a short error listing is output (see "[section "Output to SYSOUT" on page 248](#)"), followed once more by message CVR0060 to indicate that the program is ready and waiting for further statements. This makes it possible to start several conversion runs in succession without SDF-CONV having to be restarted each time.

SDF-CONV is terminated by means of the statement

```
//END
```

13.2 Statements

13.2.1 Overview of all statements

The SDF-CONV utility routine accepts the statements listed below.

CONVERT	Start the conversion of the command language and/or procedure format
END	Terminate SDF-CONV run
EXECUTE-SYSTEM-CMD	Execute system command while SDF-CONV is running
HELP-MSG-INFORMATION	Output system message to SYSOUT
HOLD-PROGRAM	Interrupt execution of SDF-CONV
MODIFY-SDF-OPTIONS	Activate or deactivate user syntax file and change SDF settings
REMARK	Write comments in statement sequence and document execution of program
RESET-INPUT-DEFAULTS	Reset task-specific default values
RESTORE-SDF-INPUT	Redisplay previously entered statements or commands on screen
SHOW-INPUT-DEFAULTS	Display task-specific default values
SHOW-INPUT-HISTORY	Display contents of input buffer
SHOW-SDF-OPTIONS	Display task-specific information on activated syntax files and options set for input and processing of statements
SHOW-STMT	Display the syntax of a statement
STEP	Define restart point in sequence of SDF-CONV statements
WRITE-TEXT	Output specific text to SYSOUT or SYSLST

CONVERT is the only program-specific statement. The remaining statements, are referred to as "SDF standard statements" (see the [section "Standard statements" on page 146](#)).

In expert mode, the CONVERT statement is preset. This means that entry of "?" causes the operand form of the CONVERT statement to be output instead of the statement menu.

13.2.2 CONVERT statement

The CONVERT statement converts ISP commands stored in files or library elements into SDF commands.

```

CONVERT

FROM-FILE = <filename 1..54> / *LIBRARY-ELEMENT(...)

  *LIBRARY-ELEMENT(...)
    LIBRARY = <filename 1..54 without-gen-vers>
  ,ELEMENT = *ALL(...) / <composed-name 1..64>(…) /
    <filename 1..64 without-cat-gen-vers with-wild>(…)
    *ALL(...)
      | VERSION = *HIGHEST-EXISTING / *ALL / <composed-name 1..24> /
      | <filename 1..24 without-cat-gen-vers with-wild>
    <composed-name>(…)
      | VERSION = *HIGHEST-EXISTING / *ALL / <composed-name 1..24> /
      | <filename 1..24 without-cat-gen-vers with-wild>
    <filename>(…)
      | VERSION = *HIGHEST-EXISTING / *ALL / <composed-name 1..24> /
      | <filename 1..24 without-cat-gen-vers with-wild>
    ,TYPE = J / <alphanum-name 1..8>

,EXPECT-CONTINUATION = *STD / *NEW-MODE / *OLD-MODE
,PARAMETER-LINES = *CONVERT / *COPY-ONLY
,TO-FILE = *STD / <filename 1..54> / *LIBRARY-ELEMENT(...)

  *LIBRARY-ELEMENT(...)
    LIBRARY = <filename 1..54 without-gen-vers>
  ,ELEMENT = *SAME(...) / <composed-name 1..64>(…)
    *SAME(...)
      | VERSION = *HIGHEST-EXISTING / *SAME / <composed-name 1..24>
    <composed-name>(…)
      | VERSION = *HIGHEST-EXISTING / *SAME / <composed-name 1..24>
    ,TYPE = *SAME / <alphanum-name 1..8>

```

continued ➡


```

,PRODUCE-CONTINUATION = *STD / *NEW-MODE / *OLD-MODE
,REPLACE-OLD-FILE = *YES / *NO
,SYSTEM-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.U-CMD / <filename 1..54>
,GROUP-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.S-CMD / <filename 1..54> / *NO
,UNCHANGED-CMD = *NONE / list-poss(30): <structured-name 1..30>
,DOCUMENTATION = *NO / *MINIMUM / *MAXIMUM
,TARGET-VERSION = *CURRENT / *V10 / *V11 / *OSD-V1 / *OSD-V2
,OUTPUT-FORM = *STD / *ACCEPTED-FORM / *INVARIANT-FORM(...)
  *INVARIANT-FORM(...)
    | CONTEXT = *ALL / list-poss(3): *BATCH / *BATCH-PROC / *DIALOG-PROC
,PROCEDURE-FORMAT = *SAME (...) / *S-PROCEDURE(...)
  *SAME(...)
    | BLOCK-INPUT = *NOT-ALLOWED / *ALLOWED
  *S-PROCEDURE(...)
    | CMD-FORMAT = *SDF / *SAME
    ,PROGRAM-INPUT = *DATA(...) / *STMT(...)
      *DATA(...)
        | EXCEPT-AFTER-CMD = *NONE / list-poss(30): <text 0..30>
        ,WARNING = *NO / *YES
      *STMT(...)
        | EXCEPT-AFTER-CMD = *NONE / list-poss(30): <text 0..30>
    ,BLOCK-INPUT = *NOT-ALLOWED / *ALLOWED

```

FROM-FILE =

Name(s) of the SAM or ISAM file or of the element(s) from a PLAM or OSM library (input procedure) which are to be converted.

The files/library elements may be located on NK4 disks.

FROM-FILE = <filename 1..54>

Name of the SAM or ISAM file to be converted.

FROM-FILE = *LIBRARY-ELEMENT(...)

Name of the library and its element(s).
Elements of an OMS library must be of type S.

LIBRARY = <filename 1..54 without-gen-vers>

Name of the library containing the element or elements to be converted.

ELEMENT =

Specifies the name(s) of the library element or elements.

ELEMENT = *ALL(...)

All elements of the specified library are to be converted. This operand value can only be specified for elements of a PLAM library.

VERSION =

Specifies the element version.

VERSION = *HIGHEST-EXISTING

The highest existing version is used by default.

VERSION = *ALL

All versions of all elements are converted. This operand value can only be specified for elements of a PLAM library.

VERSION = <composed-name 1..24>

Version number of the element.

VERSION = <filename 1..24 without-cat-gen-vers with-wild>

All versions which match the specified string are to be converted. This operand value can only be specified for SDF versions \geq V3.0 and only for elements of a PLAM library.

ELEMENT = <composed-name 1..64>(...)

Name of the library element.

If the library is an OSM library, the names of the elements it contains must not be longer than 8 characters. If more than 8 characters are entered for the element name, only the first 8 are used for selecting the element.

VERSION=

For possible specifications see the operand ELEMENT = *ALL(...).

ELEMENT = <filename 1..64 without-cat-gen-vers with-wild>(...)

All elements in the library which match the specified string are to be converted. This operand value can only be specified with SDF versions \geq V3.0 and for elements in a PLAM library.

VERSION=

For possible specifications see the operand ELEMENT = *ALL(...).

TYPE = J / <alphanum-name 1..8>

Determines the type of the element(s) of the input library.

TYPE=S must always be specified for elements of an OMS library.

EXPECT-CONTINUATION =

Position of the continuation character in the input procedure.

EXPECT-CONTINUATION = *STD

The position of the continuation character corresponds to the current SDF setting.

This setting can be displayed by means of the SHOW-SDF-OPTIONS statement and modified by means of MODIFY-SDF-OPTIONS CONTINUATION=...

EXPECT-CONTINUATION = *NEW-MODE

The position of the continuation character is expected to be behind the last character in the line.

Lines in S procedures containing commands and statements may have any length; the position of the continuation character is therefore variable.

In non-S procedures, the continuation character must be in a position preceding column 73.

EXPECT-CONTINUATION = *OLD-MODE

The position of the continuation character is expected to be in column 72 exactly.

PARAMETER-LINES =

Determines whether command records which contain procedure parameters or in which job variables are to be replaced are converted.

PARAMETER-LINES = *CONVERT

Command records which contain procedure parameters or in which job variables are to be replaced are converted. If errors occur during conversion, the commands affected are transferred unchanged to the output procedure.

PARAMETER-LINES = *COPY-ONLY

Command records which contain procedure parameters or in which job variables are to be replaced are transferred unchanged to the output procedure. These commands are not provided with comments, regardless of the value set for documentation (DOCUMENTATION operand).

TO-FILE =

Name(s) of the SAM file or the element(s) of a PLAM library to which the conversion result (output procedure) is to be written.

TO-FILE = *STD

If no file name or library element is specified, SDF-CONV automatically creates a SAM output file with the name SYSCVR.tsn.yyyy.mm.dd.hh.mm.ss. This name is made up of the prefix SYSCVR, the task sequence number, and the current date and time, and is therefore

unique. The file will not be overwritten by subsequently generated output files (regardless of the setting for the REPLACE-OLD-FILE operand). In guided dialog the generated standard file name is displayed.

TO-FILE = <filename 1..54>

Name of the output file to which the conversion result (output procedure) is to be written. If there is no file with this name in the catalog, a SAM file with this name is created. Any file with the same name in the catalog must be released for overwriting by means of the operand REPLACE-OLD-FILE=*YES.

TO-FILE = *LIBRARY-ELEMENT(...)

Name of the PLAM library and its element(s).

LIBRARY = <filename 1..54 without-gen-vers>

Name of the PLAM library which contains the element or elements to be converted.

ELEMENT =

Specifies the name(s) of the library element(s).

ELEMENT = *SAME(...)

The name of the library element matches the name of the file or element specified in the FROM-FILE operand.

VERSION =

Specifies the version of the element.

VERSION = *HIGHEST-EXISTING

By default the highest available version is used.

VERSION = *SAME

The version of the output library element matches that of the input library element or, if a SAM or ISAM file is to be converted, of the highest available version.

VERSION = <composed-name 1..24>

Version number of the element.

ELEMENT = <composed-name 1..64>(…)

Name of the library element.

VERSION =

For possible specifications see the operand ELEMENT = *SAME(...).

TYPE = *SAME / <alphanum-name 1..8>/

Determines the type of the element(s) of the output library.

If the name of an SAM or ISAM file is specified for the FROM-FILE operand, the elements of the output library are automatically of type J.

PRODUCE-CONTINUATION =

Position of the continuation character in the output procedure.

If the output procedure is to be an S procedure, a variable format is used. Continuation characters are written as described under PRODUCE-CONTINUATION=*NEW-MODE.

PRODUCE-CONTINUATION = *STD

The position of the continuation character corresponds to the current SDF setting.

This setting can be displayed by means of the SHOW-SDF-OPTIONS statement and modified by means of MODIFY-SDF-OPTIONS CONTINUATION=...

PRODUCE-CONTINUATION = *NEW-MODE

The continuation character is written behind the last character in the line.

PRODUCE-CONTINUATION = *OLD-MODE

The continuation character is written in column 72 exactly.

REPLACE-OLD-FILE = *YES / *NO

Specifies whether an existing output file or a library element may be overwritten (*YES) or not (*NO).

SYSTEM-SYNTAX-FILE =

Name of the (system) syntax file used to convert user commands.

SYSTEM-SYNTAX-FILE = \$.SYSSDF-CONV.030.USER.U-CMD

Preset (system) syntax file for converting user commands.

SYSTEM-SYNTAX-FILE = <filename 1..54>

Name of the (system) syntax file to be used to convert user commands.

GROUP-SYNTAX-FILE =

Name of the (group) syntax file used for converting privileged commands.

GROUP-SYNTAX-FILE = \$.SYSSDF-CONV.030.USER.S-CMD

Preset (group) syntax file used for converting privileged commands.

GROUP-SYNTAX-FILE = <filename 1..54>

Name of the (group) syntax file to be used for converting privileged commands.

GROUP-SYNTAX-FILE = *NO

The privileged commands are not to be converted. If there is no (group) syntax file for the user, conversion will be aborted unless *NO is specified.

UNCHANGED-CMD =

The conversion of one or more commands can be suppressed. If SDF-CONV encounters such a command, the command record is transferred to the output procedure unchanged. In addition, an error message with the weight `NOTE` is written to the conversion log. You should bear in mind that the only commands that will be recognized as commands that are to be excluded from conversion are those written exactly as specified in the `UNCHANGED-CMD` operand.

Example:

In order to exclude the `ERASE` command from conversion, all the various forms used (in the input procedure) must be specified. This could be, for example: `ERASE`, `ERAS`, `ER`.

UNCHANGED-CMD = *NONE

No commands are to be excluded from conversion.

UNCHANGED-CMD = list-poss(30): <structured-name 1..30>

The specified commands are excluded from conversion right from the outset.

Up to 30 commands or alternative command forms can be specified.

If the output procedure is to be an `S` procedure, the `PROCEDURE` command is converted accordingly even if specified in the `UNCHANGED-CMD` operand.

DOCUMENTATION =

Successful conversion of the command language causes inline comments in commands from the input procedure to be lost, with the exception of those inline comments written before the command name.

The output procedure identified by `TO-FILE` can now contain additional information in the form of comment lines. The scope of this information depends not only on the value of the `DOCUMENTATION` operand but also on the value set for the `UNCHANGED-CMD` operand (see Table 7, "Documentation level", [page 247](#)): no additional comment lines are generated for any commands specified with the `UNCHANGED-CMD` operand.

Procedure parameters contained in the additionally generated comment lines are not replaced.

DOCUMENTATION = *NO

Command records from the input procedure which have been identified as comment lines are not included in the output procedure.

DOCUMENTATION = *MINIMUM

Command records from the input procedure which contain inline comments or which consist of more than one input line are passed to the output procedure for documentation purposes. These records appear (as comment lines) in front of the lines containing the conversion result. If it was not possible to convert the command record because conversion is context-dependent or runtime-dependent, a conversion proposal is written in the comment line. The comment lines have the following format:

```
/"*****" REMARK <command record of input procedure or conversion proposal>
```

This rule does not apply to command records containing any of the following commands: BEGIN-PROCEDURE, PROCEDURE, REMARK (only if in non-S procedures), SET-PROCEDURE-OPTIONS, BEGIN-PARAMETER-DECLARATION, DECLARE-PARAMETER, END-PARAMETER-DECLARATION or their guaranteed abbreviations.

DOCUMENTATION = *MAXIMUM

All command records from the input procedure are also written (as comment lines) in front of the line containing the conversion result. If it was not possible to convert the command record because conversion is context-dependent or runtime-dependent, a conversion proposal is written in the comment line. The comment lines have the following format:

```
/"*****" REMARK <command record of input procedure or conversion proposal>
```

This rule does not apply to command records containing any of the following commands: BEGIN-PROCEDURE, PROCEDURE, REMARK (only if in non-S procedures), SET-PROCEDURE-OPTIONS, BEGIN-PARAMETER-DECLARATION, DECLARE-PARAMETER, END-PARAMETER-DECLARATION or their guaranteed abbreviations.

TARGET-VERSION =

Defines which BS2000 version the procedure is to be able to run under.

TARGET-VERSION = *CURRENT

The converted procedure is to be able to run under the BS2000 version which is loaded when conversion is performed.

TARGET-VERSION = *V10 / *V11 / *OSD-V1

The converted procedure is to be able to run under the corresponding Version of BS2000 or BS2000/OSD. In these cases the conversion of the some commands is subject to greater restrictions than are necessary for BS2000/OSD V2 and later versions.

These specifications are supported for compatibility reasons only.

TARGET-VERSION = *OSD-V2

The converted procedure is to be able to run in BS2000/OSD-BC as of Version V2.0. This specification is equivalent to TARGET-VERSION = *CURRENT.

OUTPUT-FORM =

Defines the way in which optional operands (i.e. operands with preset values) which are not specified explicitly in the command record are handled.

Regardless of what is specified for OUTPUT-FORM, all specified operands and operand values are converted to the corresponding SDF operands and operand values, provided this is not prevented by prevailing restrictions (e.g. by the UNCHANGED-CMD operand or any of the restrictions listed in [section "Restrictions" on page 250ff.](#)

OUTPUT-FORM = *STD

ISP operands and operand values that are not specified are ignored during conversion, i.e. SDF-CONV only converts explicitly specified ISP operands and operand values into the corresponding SDF formats.

When the converted procedure is executed, all operands which have not been specified are set to their default values.

Note

It is possible to modify default values in the syntax files. This may mean that the results of command execution differ.

OUTPUT-FORM = *ACCEPTED-FORM

All specifications of SDF operands and operand values are expanded to their complete names.

This operand value can only be specified with SDF versions ≥ 4.0 .

OUTPUT-FORM = *INVARIANT-FORM(...)

ISP operands and operand values that are not specified are replaced by the corresponding SDF forms in the course of conversion and assigned their default values.

In this context, users should bear in mind that not all SDF commands with their operands and default values can be executed independently of the task type and the method of execution (called either directly or within a (second) procedure). In such cases, it is vital to specify the environment (task type/execution type) in which the converted procedure is to run, as errors may result in the program being aborted (context-dependency).

This operand value can only be specified with SDF versions ≥ 3.0 .

CONTEXT =

This operand allows you to specify an environment in which the converted procedure is to run. All SDF operands and default values which are guaranteed to run in the specified environment are generated.

Furthermore, it is possible to specify up to three different environments for which all matching operands and default values are to be replaced. Operands or default values that are not supported by all specified environments are not preset.

CONTEXT = *ALL

The commands are expanded to include the operands and default values guaranteed to run in all possible environments. In this case, that would be the following environments:

in interactive mode

- guided dialog
- unguided dialog
- procedure called from the terminal (DIALOG-PROC)

in batch mode

- in the ENTER file (BATCH)
- in a procedure called by the ENTER file (BATCH-PROC)

CONTEXT = *BATCH

The output procedure is to be able to run in (simple) batch mode.

CONTEXT = *BATCH-PROC

The output procedure is to be able to run in a procedure called in batch mode.

CONTEXT = *DIALOG-PROC

The output procedure is to be able to run in a procedure called from the terminal in interactive mode.

PROCEDURE-FORMAT =

Determines the type of conversion to be applied to the input procedure(s).

PROCEDURE-FORMAT = *SAME(...)

The output procedure is to have the same format as the input procedure.

The commands of input procedures are to be converted into SDF commands.

BLOCK-INPUT =

Determines how the SDF commands generated by SDF-CONV are to be written to the output procedure.

S procedures allow more than one command to be written continuously (as a block) into a single command line, individual command records being separated by semicolon, while in non-S procedures, each command must be written into a separate line since there are no command separators.

BLOCK-INPUT = *NOT-ALLOWED

SDF-CONV writes each command into a separate line in the output procedure.

BLOCK-INPUT = *ALLOWED

If SDF-CONV converts single ISP commands into more than one SDF command, these can be written continuously into a single command line.

Any specification of this operand value is ignored during the conversion of commands from non-S procedures.

Example

/ISP1	may be converted into /SDF11;SDF12
/ISP2;ISP3	may be converted into /SDF21;SDF22;SDF23;SDF3

PROCEDURE-FORMAT = *S-PROCEDURE(...)

The input procedure is to be converted into an S procedure. The following options can be specified: conversion of input procedure commands into SDF commands (CMD-FORMAT operand), conversion of data lines into statement lines (PROGRAM-INPUT operand), definition of the output procedure format (BLOCK-INPUT operand).

CMD-FORMAT =

Specifies whether the commands of the input procedure are to be converted into SDF commands or not.

CMD-FORMAT = *SDF

The commands of the input procedure are to be converted into SDF commands as specified with the OUTPUT-FORM operand.

If the input procedure is a non-S procedure and the PROCEDURE command is specified with the UNCHANGED-CMD operand, this specification is ignored.

CMD-FORMAT = *SAME

The commands of the input procedure are to be retained unchanged.

Nevertheless, the representations of procedure parameters are adapted to S procedure conventions.

Example

```

/FS &FILENAME          is converted into /FS &(FILENAME)
/REMARK replace &LIB  is converted into /REMARK 'replace &&(LIB)'
```

Notes

- Job variable expressions are adjusted as follows:
 &(jobvariable) becomes &(JV('jobvariable'))
 In command and statement lines the job variables must begin with the character '&'.
 In data lines they must also begin with '&', but specifying PROGRAM-INPUT=*STMT means that they are not interpreted as data but as statements.
- Procedure parameters are adjusted as follows:
 &(procpa) becomes &(procpa)
 The procedure parameters must comply with the following formation rules:
 - They must be declared in the procedure header ([BEGIN-]PROC line).
 - In command and statement lines they must begin with the character '&', and in data lines with the character specified for SUBDTA=.
 - The parameter name may be no more than 20 characters long (exception: positional parameters, which are automatically truncated by SDF-CONV).

PROGRAM-INPUT =

Determines whether data lines are to be converted into SDF statement lines during conversion of a non-S procedure into an S procedure.

In non-S procedures, statement lines can have the same format as data lines, i.e. they can be entered without leading slashes, while in S procedures, statement lines must

always start with two slashes.

In any case, the representations of procedure parameters are adapted to S procedure conventions (see example under CMD-FORMAT=*SAME).

PROGRAM-INPUT = *DATA(...)

Lines without leading slashes in the input procedure are written to the output procedure unchanged i.e. as data lines.

EXCEPT-AFTER-CMD =

Determines whether individual data lines are to be excepted from the general handling of data lines.

EXCEPT-AFTER-CMD = *NONE

All data lines are to be transferred to the output procedure unchanged.

EXCEPT-AFTER-CMD = list-poss(30): <text 1..30>

Permits exceptional conversion of individual data lines into SDF statement lines. Any data line following a command line containing at least one of the elements listed will then be converted into an SDF statement line; this exceptional handling will be continued until the next command line containing none of the elements listed.

WARNING = *NO / *YES

Determines whether a WARNING message is to be issued (*YES) or not (*NO) each time SDF-CONV detects a data line during conversion.

PROGRAM-INPUT = *STMT(...)

Lines without leading slashes in the input procedure are converted into SDF statement lines by prefixing two slashes to each of these lines.

EXCEPT-AFTER-CMD =

Determines whether individual data lines are to be excepted from the general handling of data lines.

EXCEPT-AFTER-CMD = *NONE

All data lines are to be converted into SDF statement lines.

EXCEPT-AFTER-CMD = list-poss(30): <text 1..30>

Permits individual data lines to be excepted from conversion into SDF statement lines i.e. are to be written to the output procedure unchanged. Any data line following a command line containing at least one of the elements listed will then be transferred unchanged; this exceptional handling will be continued until the next command line containing none of the elements listed.

BLOCK-INPUT =

Determines how the SDF commands generated by SDF-CONV are to be written to the output procedure.

S procedures allow more than one command to be written continuously (as a block) into

a single command line, individual command records being separated by semicolon, while in non-S procedures, each command must be written into a separate line since there are no command separators.

BLOCK-INPUT = *NOT-ALLOWED

SDF-CONV writes each command into a separate line in the output procedure.

BLOCK-INPUT = *ALLOWED

If SDF-CONV converts single ISP commands into more than one SDF command, these can be written continuously into a single command line.

Any specification of this operand value is ignored during the conversion of commands from non-S procedures.

For an example see `PROCEDURE-FORMAT=*SAME(BLOCK-INPUT=*ALLOWED)`.

13.2.3 Control functions

The following inputs are used for dialog guidance control. They are accepted in the statement part of SDF-CONV and executed immediately.

Functions	Brief description
<statement> ?	SDF-CONV switches to (temporarily) guided dialog
<statement> .. <operand> = ?	SDF-CONV provides information about the function of the operand and the possible operand values
K1 key	returns from (temporarily) guided dialog to unguided dialog
K2 key	switches to command level; the RESUME-PROGRAM command is used to return to SDF-CONV

Table 6: Control functions in the statement part of SDF-CONV

A full overview and a description of all possible inputs in a (temporarily) guided and an unguided dialog are given in the [chapter “Input of commands and statements” on page 33](#).

13.3 Outputs

13.3.1 Output procedure

The TO-FILE operand of the CONVERT statement defines the name of the output procedure or, if two or more input procedures stored in library elements are to be converted, the names of the output procedures.

In this context, it is vital to remember that the FROM-FILE and TO-FILE values must always be different. Regardless of the format of the input procedure, a cataloged SAM file or PLAM library element is always created for the output procedure. The operand REPLACE-OLD-FILE = *YES can be used to specify that an existing file or a library element with the same name can be overwritten. No empty files or library elements are generated. This means that in the event of aborting the conversion before writing any command record to the output procedure file specified with TO-FILE, no output procedure is generated.

The output procedure contains the converted commands or the ISP command records taken over unchanged from the input procedure.

If SDF command records form part of the input procedure, the specified commands are expanded to their full command names.

Depending on the values of the DOCUMENTATION and UNCHANGED-CMD operands, the output procedure may also contain additionally added comment lines (see the table [7 on page 247](#)). These additionally inserted comment lines may be useful to the user in the following cases:

1. For manually post-editing the converted procedure (output procedure): certain ISP commands (e.g. JOIN) cannot be converted automatically as they have various SDF equivalents, the appropriate one being determined by the context. In the comment line, SDF-CONV makes a suggestion as to the appropriate conversion. This comment line is written in front of the command line with the unconverted command. The output procedure then contains the following lines:


```
/"*****" REMARK <sdf-cmd = conversion proposal>
/<isp-cmd>
```
2. For retaining inline comments in an ISP command record: If SDF-CONV encounters a command record with inline comments, the command, the operands and the operand values are converted but the comment is not transferred. Insertion of the ISP command record ensures that the information contained in the comment is not lost and can be incorporated in the command record manually if required. The output procedure contains the following lines:


```
/"*****" REMARK <isp-cmd "with inline comment">
/sdf-cmd
```

The following table shows the scope of the output procedure depending on the values set for the UNCHANGED-CMD and DOCUMENTATION operands. This is illustrated by way of an example based on two ISP commands:

```

/SYSFILE SYSLST=( )           → /ASSIGN-SYSLST TO-FILE=*PRIMARY
/ERASE OTTO "DELETE"         → /DELETE-FILE FILE-NAME=OTTO

```

DOC	UNCHANGED-CMD = *NO ¹	UNCHANGED-CMDS = isp-cmd ²
*NO	<pre> /sdf-cmd ... The command is converted. /ASSIGN-SYSLST TO-FILE=*PRIMARY /DELETE-FILE FILE-NAME=OTTO </pre>	<pre> /isp-cmd ... The command is not converted. /SYSFILE SYSLST=() /ERASE OTTO "DELETE" </pre>
*MIN	<pre> /REMARK isp-cmd ... (not always) /sdf-cmd ... The command is converted. If the command is longer than one line or contains an inline comment, SDF-CONV additionally writes this record to an inserted comment line. /ASSIGN-SYSLST TO-FILE=*PRIMARY /REMARK ERASE OTTO "DELETE" /DELETE-FILE FILE-NAME=OTTO </pre>	<pre> /REMARK sdf-cmd ... /isp-cmd ... The command is not converted, but SDF- CONV writes a conversion proposal in an inserted comment line. /REMARK ASSIGN-SYSLST TO-FILE=*PRIMARY /SYSFILE SYSLST=() /REMARK DELETE-FILE FILE-NAME=OTTO /ERASE OTTO "DELETE" </pre>
*MAX	<pre> /REMARK isp-cmd ... /sdf-cmd ... The command is converted. In addition, SDF-CONV writes the command record to an inserted comment line. /REMARK SYSFILE SYSLST=() /ASSIGN-SYSLST TO-FILE=*PRIMARY /REMARK ERASE OTTO "DELETE" /DELETE-FILE FILE-NAME=OTTO </pre>	<pre> /REMARK sdf-cmd ... /isp-cmd ... The command is not converted, but SDF-CONV writes a conversion proposal to an inserted comment line. /REMARK ASSIGN-SYSLST TO-FILE=*PRIMARY /SYSFILE SYSLST=() /REMARK DELETE-FILE FILE-NAME=OTTO /ERASE OTTO "DELETE" </pre>

Table 7: Documentation level in the output procedure depending on the values set for the UNCHANGED-CMD and DOCUMENTATION operands

¹ Provided conversion of the ISP command isp-cmd is not subject to any restrictions.

² Or: the ISP command cannot be converted because its meaning is context-dependent or runtime-dependent.

13.3.2 Output to SYSOUT

Following every conversion run, SDF-CONV outputs an overview of the number of errors, warnings and comments detected in the course of conversion. This overview has the following format (i = number):

```
% CVR1000 CONVERSION TERMINATED WITH:
% CVR1001          FAILURES:      i
% CVR1002          ERRORS   :      i
% CVR1003          WARNINGS:      i
% CVR1004          NOTES    :      i
```

FAILURES and ERRORS denote fatal (DMS) errors. When such errors occur, the error message is output both to SYSLST (see below) and also to SYSOUT. The SDF-CONV conversion run may be aborted after error messages of this type. SDF-CONV outputs the overview and waits for further statements after issuing the message CVR0060.

Details of the format of the messages can be found under [section "SDF-CONV messages" on page 298](#).

13.3.3 Conversion log to SYSLST

In addition to generating the output procedure, SDF-CONV writes a conversion log and outputs it automatically to SYSLST. The log is in three parts:

- Part 1. List of the operand values currently valid for the SDF-CONV run performed.
- Part 2. List of all records in the input file and of all messages relating to errors that occurred during conversion.
- Part 3. List detailing the number of errors and notes (this is the same as the list output to SYSOUT).

The second part of the log records messages in the following cases:

- there is no equivalent SDF command for the ISP command
- the ISP command cannot be unequivocally assigned to an SDF command
- when specified with certain operands, some ISP commands have no SDF equivalent
- a syntax error has been found

Command records containing such errors are not converted; instead they are copied unchanged to the output procedure. If an ERROR or FAILURE is reported before conversion starts, the second part of the log is not generated. The error messages are listed under "SDF-CONV messages" in the appendix.

The system file SYSLST can be assigned to a cataloged file by means of the ASSIGN-SYSLST command or copied using the COPY-SYSTEM-FILE command.

13.3.4 Program monitoring using monitoring job variables

SDF-CONV can supply a program-monitoring job variable with the following values:

'\$T 0000'	Program terminated normally. Conversion without notes.
'\$T 0001'	Program terminated normally. Conversion with notes.
'\$T 1002'	Program terminated normally. Conversion with warnings.
'\$T 1003'	Program terminated normally. Conversion with errors.
'\$A 3004'	Program terminated abnormally due to serious error, or CPU time limit exceeded.

The program-monitoring job variable must be declared in the START-SDF-CONV command (or, for the compatible call, in the START-EXECUTABLE-PROGRAM-SDF-CONV command). For further information see the manuals "Commands" [1] or "Job Variables" [9].

13.4 Restrictions

SDF-CONV cannot convert all ISP commands and old SDF commands and their operands into SDF commands with corresponding operands. Certain commands, operands and operand configurations are excluded from conversion right from the start. Moreover, there are certain so-called "conditional conversions" which are dependent on the operating system version. The conversion of procedure parameters as operands can also be subject to restrictions, as well as the conversion of non-S procedures into S procedures. The command records concerned are transferred unchanged to the output procedure (SDF-CONV outputs an error message to SYSLST) or, if the condition regarding version dependency was fulfilled, are converted.

13.4.1 Basic preconditions and restrictions

- Error-free procedures in ISP format are a basic prerequisite for producing error-free procedures in SDF format.
- SDF-CONV only converts ISP commands that were defined in BS2000 versions \leq V10.
- The FROM-FILE and TO-FILE operands must always be assigned different values. This applies even if the operand REPLACE-OLD-FILE = *YES is specified.
- Every command in the input procedure that is not described in one of the syntax files is copied unchanged to the output procedure. An error message is output to SYSLST.
- All commands which refer to card readers are copied unchanged to the output procedure, because such devices ceased to be supported by BS2000 with the release of V10.0.
- Operands to which no explicit values have been assigned can be generated for SDF commands. These operands implicitly receive the default value specified in the operand description of the relevant command (see the the "Commands" manual [1]).
- Procedure parameters must not be part of a command or operand and must not themselves be either a command or an operand.

Example

```
/PROC N,(&PARAM1 = 'SHOW', &PARAM2 = 'FILE')  
/...  
/&PARAM1-SDF-OPTIONS  
/CREATE-FILE &PARAM2 = OTTO.DAT  
/...  
/ENDP
```

- Commands containing procedure parameters can be converted to SDF format only if their conversion is not dependent on the parameters. In all other cases no one-to-one assignment can be made, and so the commands are not converted (cf. [page 256ff](#)).

Example

```

/PROC N,(&USE1)
/...
/RFD USE=&USE1
/...
/ENDP

```

```

&USE1 = NO      is converted in SDF to: /STOP-DISKETTE-INPUT
&USE1 = INPUT   is converted in SDF to: /START-DISKETTE-INPUT

```

- If the input procedure contains the SDF command MODIFY-SDF-OPTIONS, the value of the CONTINUATION operand must not be a procedure parameter.
- If the input procedure contains the SDF command ENDP-RESUME and the output procedure is to be an S procedure, the command is converted into EXIT-PROCEDURE RESUME=YES.
If the output procedure is to be a non-S procedure, the command is not converted since the procedure's spin-off mechanism is different in this case.
- All names beginning with an asterisk and containing no other wildcards are interpreted as keywords.
- The old label format (.label) is not converted into the new label format (label:).
- SDF-CONV does not recognize nested procedure parameters or nested replaceable job variables when converting ISP commands from S procedures into SDF commands, so that the command records concerned are not converted.
- Successful conversion of the command language causes inline comments in commands from the input procedure to be lost. This can be avoided by specifying the CONVERT statement operand DOCUMENTATION=*MIN/*MAX, which causes inline comments to be transferred to the output procedure.

13.4.2 Restrictions on the conversion into S procedures

- ENTER files cannot be converted into S procedures by means of SDF-CONV.
- SDF-CONV reports an error if the name of a procedure parameter exceeds 20 characters. While the names of procedure parameters in non-S procedures may be up to 255 characters long, a maximum length of 20 characters applies to S procedures. This restriction does not apply to positional operands; these are automatically abbreviated by SDF-CONV, if required.
- Command lines in non-S input procedures are only evaluated up to the 72nd character; all characters from column 73 onward are ignored.
- Conversion may affect the logging control. While in non-S procedures, logging is determined exclusively by the called procedure, in S procedures, it is also dependent on the relevant setting for the calling environment.
- No conversion takes place if more than one job variable has to be replaced in any value of any procedure parameter.
- If the first significant character following a command name in a non-S input procedure is an equal sign, SDF-CONV issues a WARNING, except in the event of command records containing the REMARK command.
- SDF-P does not support recursive replacement of procedure parameters.
- Procedure parameters and job variables in S procedures can contain further expressions which are to be replaced within the parentheses which enclose them, e.g. &(JOBVAR1&(JOBVAR2)). This feature is not supported by SDF-CONV.

Example

1. Non-S procedure:

```

/BEGIN-PROC A,PAR=YES(PROC-PAR=(&TEST=))
/MOD-JV JV(JV-NAME=JOBVAR),VALUE='&&TEST'
/SHO-JV JV(JV-NAME=JOBVAR)
/REMARK &TEST
/WR-TEXT '&(JOBVAR)'
/EXIT-PROC

```

Trace listing of the non-S procedure:

```

/CALL-PROC EXA-NON-S
%/BEGIN-PROC A,PAR=YES(PROC-PAR=(&TEST=))
%/MOD-JV JV(JV-NAME=JOBVAR),VALUE='&TEST'
%/SHO-JV JV(JV-NAME=JOBVAR)
%&TEST
%/REMARK &TEST
%&TEST=hallo
%/REMARK HALLO

```

```

%/WR-TEXT 'HALLO'
HALLO
%/EXIT-PROC

```

The job variable **JOBVAR** is replaced in two steps.
JOBVAR contains the value of the procedure parameter **TEST** i.e. **HALLO**.

2. S procedure:

```

/SET-PROC-OPTIONS LOG=YES
/BEGIN-PAR-DEC
/DECLARE-PAR NAME=TEST (INIT=*PROMPT)
/END-PAR-DEC
/MOD-JV JV(JV-NAME=JOBVAR),VALUE='&&TEST'
/SHO-JV JV(JV-NAME=JOBVAR)
/REMARK &TEST
/WR-TEXT '&(JV('JOBVAR'))'

```

Trace listing of the S procedure:

```

/CALL-PROC EXA-S,LOG=YES
% 1 1 /SET-PROCEDURE-OPTIONS LOGGING-ALLOWED=YES,INTERRUPT-
% ALLOWED=YES,DATA-ESCAPE-CHAR=NONE,SYSTEM-FILE-CONTEXT=OWN,-
% DATA-ERROR-HANDLING=NO
% 1 1 /BEGIN-PARAMETER-DECLARATION
% 1 1 /DECLARE-PARAMETER NAME=TEST,TYPE=ANY,INIT=*PROMPT
% 1 1 /END-PARAMETER-DECLARATION
% 6 1 /MOD-JV JV(JV-NAME=JOBVAR),VALUE='&&TEST'
% 7 1 /SHOW-JV JV(JV-NAME=JOBVAR)
%&TEST
%TEST: hallo
% 8 1 /REMARK 'HALLO'
% 9 1 /WRITE-TEXT '&TEST'
&TEST
% 11 1 /IF-BLOCK-ERROR
% 11 1 /END-IF
% 11 1 /EXIT-PROCEDURE

```

The job variable **JOBVAR** is replaced in a single step.
JOBVAR contains **&TEST**.

13.4.3 Commands that cannot be converted

- **Commands for non-privileged users**

ISP command	Explanation
ACCOUNT	This command is no more available
AUDIT	No one-to-one assignment is possible (conversion is runtime-dependent)
DATA	The command has no SDF equivalent
DIAG	This is an IDA command, not an ISP command.
DO	The command may collide with global parameters (conversion is context-dependent); a conversion proposal can be found in a comment line of the output procedure, provided DOCUMENTATION = *MIN/*MAX is set
END	This command has no SDF equivalent
PARAMETER	This command has no SDF equivalent

Table 8: Non-convertible commands for non-privileged users

- **Commands for system administration**

ISP command	Explanation
JOIN	No one-to-one assignment is possible (conversion is context-dependent); a conversion proposal can be found in a comment line of the output procedure of the output procedure, provided DOCUMENTATION = *MIN/*MAX is set
LOADAID	The command has no SDF equivalent
LOADAIDSYS	The command has no SDF equivalent
RCARD	Card readers are no longer supported
SDVC	No one-to-one assignment possible (conversion is context-dependent)

Table 9: Non-convertible commands for system administration

- **Commands for the operator at the console**

ISP command	Explanation
AGOGO ASR ASTOP BCACT BCAPPL BCASP BCCONN BCCONP BCDAC BCDISCON BCDISP BCEND BCGEN BCIN BCLOSE BCMAP BCMOD BCMOFF BCMON BCOPTION BCOUT BCSET BCSHOW BCSWP BCTIMES BCXAF BROADCAST CONSOLE DADM DCDIAG DCSTART MESSAGE MRSEND MRSMOD MRSSTART SHUTDOWN TURN	These operator commands can only be entered at the console, not at a terminal.

Table 10: Non-convertible commands for the operator at the console

13.4.4 Non-convertible operand configurations and conditional conversions

The following ISP commands are not converted, or only converted if certain conditions are fulfilled, when the listed operands are specified.

- Commands for non-privileged users

Command to be converted	Non-convertible operands or operands which restrict conversion	Explanation
FILE	LINK *DUMMY, LINK, TSET together	the command is not converted when: the LINK operand was specified and other operands apart from the file name were specified or these three operands were specified in a FILE command
MODIFY-JV-CONDITIONALLY	SET-VALUE	due to the changes in job variable syntax in JV V11.2 (as of BS2000/OSD-BC V2.0) errors may occur in the interpretation of the link name
PRINT	(file1 , file2 , ...) CONTROL=12LPI PNAME=	Format3 (= multiple files) is converted only if the files are specified with a fully or partially qualified name or with their <eam-file-numbers>; specification of more than one *SYSLSTnn file is not supported the operand is no longer supported any hyphen contained in a name is deleted without substitution during conversion into PRINT-FILE SPOOLOUT-NAME=...

Table 11: Non-convertible operand configurations and conditional conversions (non-privileged users)

Command to be converted	Non-convertible operands or operands which restrict conversion	Explanation
PRINT-FILE	LAYOUT-CONTROL CHARACTER-SETS CHARS-MODIFICATION CONTROL-CHARACTERS =PHYSICAL(LINE-SPACING=...) =PHYSICAL(... , PAGE-CONTROL=..) =TRANSPARENT() PRINTER-RESSOURCES =APA(..., TABLE-REFERENCE- CHAR=...) TRAY-NUMBER ROTATION DELETE-FILE	<p>this operand causes problems if specified at the same time as PRINTER-RESOURCES=APA(TABLE-REFERENCE-CHAR)</p> <p>procedure parameters are not accepted as operand values</p> <p>procedure parameters are not accepted as operand values</p> <p>procedure parameters are not accepted as operand values; the PRINT-DOCUMENT command differentiates between SPOOL and RSO; as a result the default value is ignored</p> <p>procedure parameters are not accepted as operand values</p> <p>the PRINT-DOCUMENT command differentiates between SPOOL and RSO; as a result the default value is ignored</p> <p>procedure parameters are not accepted as operand values</p> <p>the operand causes problems if specified at the same time as PRINTER-RESOURCES=LP65((INPUT-TRAY-NUMBER=...).</p> <p>Specification of a list is not supported.</p> <p>procedure parameters are accepted during conversion, but any value specified for ERROR-PROCESSING=*PAR (TRUNCATION=...) is ignored</p>
	DEVICE-NAME =...(DESTINATION=...) =...(DEVICE-TYPE=...)	<p>procedure parameters are not accepted as operand values</p> <p>the operand causes problems if specified at the same time as PRINTER-RESOURCES=LP65/APA.</p>

Table 11: Non-convertible operand configurations and conditional conversions (non-privileged users)

Command to be converted	Non-convertible operands or operands which restrict conversion	Explanation
PRINT-FILE (continued)	ERROR-PROCESSING =*PAR(TRUNCATION=...) FILE-NAME =*SYSLST-NUMBER() =*LIB-ELEM(..., CREATION-DATE=...) ---- FILE-PART =SECTION(SECTION-ID=... ,FIRST-RECORD=... ,LAST-RECORD=...) RECORD-PART =PAR(FIRST-CHAR=... ,LAST-CHAR=...) REPEAT SPOOL-OUT-NAME START-SPOOL =NO	procedure parameters are not accepted as operand values procedure parameters are not accepted as operand values specification of *EAM() and *SYSLST-NUMBER() in list form is not supported this suboperand is not available in the command PRINT-DOCUMENT procedure parameters are accepted as operand values, but if the procedure parameter is an integer (e.g. a SYSLST file number), the conversion result will be incorrect procedure parameters are not accepted as operand values procedure parameters are not accepted as operand values If the operand value is zero, it is converted to 1. procedure parameters are not accepted as operand values procedure parameters are not accepted as operand values procedure parameters are not accepted as operand values the operand value is no longer supported
PUNCH	(file1 , file2 , ...) <eam-file-number> TAPE=	Format3 (= multiple files) has no SDF equivalent file names consisting solely of digits are not permitted pooler tapes are not supported as of BS2000 V11
RESUME	with operands	is an IDA command
RESUME-PROCEDURE	MODE=C	the operand is not supported as C can stand for COMMAND/CMD as well as for CURRENT
SECURE	WORK=...	the operand has no SDF equivalent

Table 11: Non-convertible operand configurations and conditional conversions (non-privileged users)

Command to be converted	Non-convertible operands or operands which restrict conversion	Explanation
SYSFILE	FILE=...	the operand is too complex, one-to-one assignment is not possible; conversion is carried out only if the file name is specified in the following form: FILE=<filename-without-gen-vers>
TCHNG	CORR=C'a'	the operand is no longer supported

Table 11: Non-convertible operand configurations and conditional conversions (non-privileged users)

- Commands for the system administration

ISP command	Non-convertible operands or operands which restrict conversion	Explanation
EXCAT	QUIET	the operand is no longer supported
IMCAT	FORM=CAT RESET= BUFNUM=	the operand is no longer supported; a conversion proposal can be found in the comment line of the output procedure provided DOCUMENTATION = *MIN/*MAX is set the operands are no longer supported
RFD	USERID=userid OWNERID='ownerid'	the keywords must be specified

Table 12: Non-convertible operand configurations and conditional conversions (system administration)

13.4.5 Restrictions on procedure parameters as operands

The following examples illustrate the difficulties that arise when ISP commands containing procedure parameters as operands are to be converted to SDF format.

1. The SDF command is determined by the ISP operand value

```
/CATJV ... ,STATE=UPDATE    → /MODIFY-JV-ATTRIBUTES ...
/CATJV ... ,STATE=NEW       → /CREATE-JV ...
/CATJV ... ,STATE=&OP1      → / ?
```

2. The SDF operand value is determined by the ISP operand value

```
/EXECUTE ... ,SYMTEST=NO    → /START-PROGRAM ... ,TEST-OPTIONS=NONE
/EXECUTE ... ,SYMTEST=ALL   → /START-PROGRAM ... ,TEST-OPTIONS=AID
/EXECUTE ... ,SYMTEST=&OP2  → /START-PROGRAM ... ,TEST-OPTIONS= ?

/GETJV <name>               → /SHOW-JV JV-ID=JV-NAME(JV-NAME=<name>)
/GETJV *<name>              → /SHOW-JV JV-ID=JV-LINK(JV-LINK=<name>)
/GETJV &OP3                 → /SHOW-JV JV-ID= ?
```

Note

Command records containing procedure parameters, or in which job variables are to be replaced, can be excluded from conversion by specifying the operand `PARAMETER-LINES = *COPY-ONLY` in the `CONVERT` statement. Instead of converting the command records concerned, SDF-CONV writes them unchanged to the output procedure. No messages are issued in such cases.

The following tables provide an overview of the commands that are not converted if the listed positional and/or keyword operands are specified as procedure parameters.

`<i>` means that the *i*-th positional operand must not be a parameter.
`OPERAND=` means that the positional operand value must not be a parameter.

- Commands for non-privileged users

ISP command	No parameter as positional operand ...	No parameter as a value of the keyword operand ...
CANCEL	<2> ,<3>	
CATJV		SHARE=, STATE=
COPY		DIALOG=, WRITE=
DELON	<1>	
ENTER		LOG=, TIME=
ERAJV		all operands
ERASE	<1> ,<2>	CR=, EX=, LA=, LIST=, POS=
EXECUTE	<1>	CLASSII=, IDA=, SYMTEST=
FSTATUS	<2>	CRDATE=, EXDATE=, EXTENDS=, FREESIZE=, FROM=, LADATE=, LIST=, PASS=, SHARE=, SIZE=, SORT=, SUPPORT=, STATE=, TYPE=, VTOC=
GETJV	<1>	
HELP		INF=
IMPORT		LIST=
LOAD	<1>	CLASSII=, IDA=, SYMTEST=
LOGOFF	<1> ,<2>	
OPTION		MSG=, TESTPRIV=
PASSWORD		REL=
PRINT	<1> ,<2>	BINARY=, CHARS-POOL=, CONTROL=, COPIES=, DEFER=, DELETE-FILE=, DEVICE=, DEVIN=, FROM=, LOOP=, ROTATION=, SPACE=, TAPE=, TO=, TRANSLATION-TABLE=, VOLUME=
PRIORITY	<2>	
PUNCH	<1> ,<2>	ACCESS=, BY-PASS=, DEVICE=, DEVIN=, FDTYPE=, SKEL=, TAPE=, VOLUME=, BLKTYPE=, WRITEPR=
RDTFT	<1>	LINK=
RELEASE	<2>	
RESTART	<2> ,<3>	CHECK=
SECURE		FILE=, VOLUME=
SETJV	<1> ,<2>	
SETSW		INVERT=, OFF=, ON=

Table 13: Restrictions affecting procedure parameters as operands (non-privileged users)

ISP command	No parameter as positional operand ...	No parameter as a value of the keyword operand ...
SET-SS-OPTIONS		SS-NAME=
SKIP	<1>	
SPARAM		COMPRESS=
SQUC		REL=, SUSP=, TYPE=
STAJV	<1> ,<2>	
STATUS	<1> ,<2>	DISP=, IDENT=, INTYPE=, TIMEREQ=, TYPE=
SYSFILE		SYSDTA=, SYSIPT=, SYSLST=, ..., TASKLIB=
SYSTATUS	<1>	
TCHNG		OFLOW=, READ=, TCHAR=
VERIFY		REPAIR=, SUPPORT=

Table 13: Restrictions affecting procedure parameters as operands (non-privileged users)

- Commands for the system administration

ISP command	No parameter as positional operand ...	No parameter as a value of the keyword operand ...
CATM		BUFCLS=, STATE=, WAIT=
EXCAT	<2>	
IMCAT		ACNTNUM=, BUFCLS=, FORM=
MSGCONTROL		FILE=
RFD		USE=
SEVER	<2>	
SPMGT		SECONDARY=
SQUC		REL=, SUSP=, TYPE=
STAM	<1>	HOST=, REF=

Table 14: Restrictions affecting procedure parameters as operands (system administration)

13.5 Components and installation of SDF-CONV

The utility routine SDF-CONV can run with BS2000/OSD.

SDF-CONV is supplied as a subsystem but can, for reasons of compatibility, also be started as a simple program. SDF-CONV converts ISP commands which were defined in a BS2000 version \leq V10.

The SDF-CONV utility routine consists of the following components:

- the module library SYSLNK.SDF-CONV.030
for calling the utility routine as a subsystem by means of the START-SDF-CONV command
- the conversion routine SYSPRG.SDF-CONV.030
which is available for reasons of compatibility for calling the utility routine by means of the command START-EXECUTABLE-PROGRAM.
- the syntax files
 - SYSSDF.SDF-CONV.030, (system) syntax file containing the syntax for the SDF-CONV program statements,
 - SYSSDF.SDF-CONV.030.USER.U-CMD, (system) syntax file, and
 - SYSSDF.SDF-CONV.030.USER.S-CMD (group) syntax file containing the command syntax for the commands to be converted, split up according to functional range for system administration (group syntax file) and non-privileged users (system syntax file)
- the declaration file
 - SYSSSC.SDF-CONV.030.112
- the message file
 - SYSMES.SDF-CONV.030
- the REP file SYSREP.SDF-CONV.030 for object corrections.

Subsystem installation

SDF-CONV is a BS2000 subsystem and is managed by DSSM. It cannot be loaded by DSSM until it has been declared in the subsystem catalog. The subsystem declarations required to do this are contained in the file SYSSDF.SDF-CONV.030.112.

The system administrator has to proceed as follows to install the subsystem:

The subsystem declaration SYSSSC.SDF-CONV.030.112 must be added to the subsystem catalog by means of SCCM. The START-SUBSYSTEM command is used to activate the SDF-CONV subsystem. This means that the message file, the syntax file SYSSDF.SDF-CONV.030 and the REP file also belong to the SDF-CONV subsystem and are activated automatically.

By default, the two other syntax files (SYSSDF.SDF-CONV.030.USER.U-CMD and SYSSDF.SDF-CONV.030.USER.S-CMD) are stored under the standard system ID "\$". In order to install them under another ID, the preset path names in the operands SYSTEM-SYNTAX-FILE and GROUP-SYNTAX-FILE of the CONVERT statement must be modified accordingly.

These syntax files must not be merged into the current BS2000 system or group syntax files.

Before SDF-CONV can be called with the START-EXECUTABLE-PROGRAM command, the following command must be issued:

```
/MODIFY-FILE-ATTRIBUTES SYSPRG.SDF-CONV.030, NEW-NAME=SDF-CONV,-  
/ USER-ACCESS=*ALL
```

Generation of the subsystem catalog and the SSCM routine is described in the manual "Introductory Guide to Systems Support" [2], management of group and system syntax files is described in the [chapter "SDF syntax files" on page 137](#), the SDF-I utility routine is described in the [chapter "SDF-I" on page 165](#), and UGEN is described in the manual "System Installation" [3].

Note:

If optional REPs are used in the system, preset operand values in commands such as CATALOG can be modified. SDF-CONV ignores these changes unless the system administrator enters them in the syntax files using the SDF-A utility routine.

13.6 Examples

This chapter contains examples of the conversion of non-S procedures into S procedures and of ISP commands into SDF commands.

In each of the examples, the input and output procedures as well as the logs produced by SDF-CONV are reproduced and annotated.

Example 1: The procedure DO.REORG reorganizes storage space occupied by ISAM files. The conversion into SDF commands as well as into S procedure format is not subject to any restrictions, SDF-CONV terminates without errors.

Input procedure: DO.REORG
Logging file: LST.REORG
Output procedure: SDF.SDFP.DO.REORG

Example 2: The procedure DO.LIB.CONTENTES causes the most important element data to be output for all elements in a library. The conversion of the PRINT and DO commands is subject to certain restrictions. SDF-CONV issues a NOTE for each.

The procedure format (non-S) is retained unchanged.

Input procedure: DO.LIB.CONTENTES
Logging file: LST.LIB.CONTENTES
Output procedure: SDF.DO.LIB.CONTENTES

Example 3: The procedure PROC.ASSXT is a small auxiliary procedure that is to speed up processing (system file assignment, assembler call) in the event of frequent access to the ASSEMBH assembler.

The procedure was written as a non-S procedure using the SDF command language. No restrictions apply to its conversion into an S procedure. SDF-CONV terminates without errors.

Input procedure PROC.ASSXT
Logging file: LST.ASSXT
Output procedure: SDFP.PROC.ASSXT

Example 1:

The procedure DO.REORG reorganizes the storage space of an ISAM file. The conversion into SDF commands as well as into S procedure format is not subject to any restrictions, SDF-CONV terminates without errors.

Input procedure DO.REORG

```

/PROC N,(&FNAME),SUBDTA=&
/DCLJV #NEWIMAG
/SETJV #NEWIMAG,X'0C'
/SETSW ON=(4,5), OFF=(15)
/SYSFILE SYSDTA=(SYSCMD)
/WR-TEXT ' '
/WR-TEXT '*****'
/WR-TEXT '***      Reorganization of ISAM files      ***'
/WR-TEXT '*****'
/WR-TEXT ' '
/WR-TEXT 'Name of file to be reorganized:'
/REMARK &FNAME
/EXEC $EDT
@GET '&FNAME' N
@CON
@IF N:@IF N D:@SY'COPY &FNAME,&FNAME..BEFORE,SAME'
@CON
@IF N : @IF N D : @SAVE 0
@CON
@IF E : @SETSW ON=15
@IF D : @SETSW ON=15
@HALT
/SKIP .ERR, ON=(15)
/TCHNG OFLOW=NO
/WR-TEXT '&(#NEWIMAG)'
/TCHNG OFLOW=ACK
/WR-TEXT ' '
/WR-TEXT 'File &FNAME has been reorganized.'
/WR-TEXT ' '
/WR-TEXT 'Space required before:'
/FS &FNAME..BEFORE
/WR-TEXT ' '
/ERASE &FNAME..BEFORE
/WR-TEXT 'Space required now:'
/FS &FNAME
/SKIP .EOP, OFF=(15)
/.ERR REMARK
/WR-TEXT ' '
/WR-TEXT '***** Error during file processing *****'

```

```

/WR-TEXT '** REORGANIZATION HAS NOT BEEN PERFORMED ! **'
/WR-TEXT ' '
/.EOP STEP
/SETSW OFF=(4,5,15)
/SYSFILE SYSDTA=(PRIMARY)
/ENDP

```

SDF-CONV run

```

/start-sdf-conv _____ (1)
% BLS0517 MODULE 'CVRMAIN' LOADED
% CVR0001 SDF-CONV VERSION 'V03.0B80' STARTED
% CVR0060 SDF-CONV VERSION 'V03.0B80' READY
//convert from-file=do.reorg,to-file=sdf.sdfp.do.reorg,doc=*min,- _____ (2)
//procedure-format=*s-proc(prog-inp=*data,block-inp=*allowed)
% CVR1000 CONVERSION TERMINATED WITH:
% CVR1001 FAILURES: 0
% CVR1002 ERRORS : 0
% CVR1003 WARNINGS: 0
% CVR1004 NOTES : 0
//end _____ (3)

```

- (1) Start of the SDF-CONV utility routine.
- (2) The CONVERT statement assigns the input and output procedures as well as defining the documentation level. The input procedure is to be converted into an S procedure. Block input of generated SDF commands is to be allowed. All other operands of the CONVERT statement receive the default value. Once processing has been completed, SDF-CONV reports that the conversion was performed without warnings or errors. This listing is output to SYSOUT.
- (3) SDF-CONV is terminated with the END statement.

Conversion log (SYSLST)

```

***** BS2000 COMMAND CONVERTER : SDF-CONV V3 ***** page 1

% CVR1010 OPTIONS IN EFFECT: _____ (1)
% CVR1011 FROM-FILE = DO.REORG
% CVR1012 EXPECT-CONTINUATION = *NEW-MODE
% CVR1013 PARAMETER-LINES = *CONVERT
% CVR1014 TO-FILE = SDF.SDFP.DO.REORG
% CVR1015 PRODUCE-CONTINUATION = *NEW-MODE
% CVR1016 REPLACE-OLD-FILE = *YES
% CVR1017 SYSTEM-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.U-CMD
% CVR1018 GROUP-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.S-CMD
% CVR1019 UNCHANGED-CMD = *NONE
% CVR1020 DOCUMENTATION = *MINIMUM
% CVR1021 TARGET-VERSION = *OSD-V2
% CVR1022 OUTPUT-FORM = *STD
% CVR1023 PROCEDURE-FORMAT = *S-PROCEDURE(CMD-
FORMAT=*SDF,PROGRAM-INPUT=*DATA(EXCEPT-AFTER-CMD=*NONE,WARNING=*NO),BLOCK-
INPUT=*ALLOWED)

```

```

***** BS2000 COMMAND CONVERTER : SDF-CONV V3 ***** page 2

/PROC N,(&FNAME),SUBDTA=& _____ (2)
/DCLJV #NEWIMAG
/SETJV #NEWIMAG,X'OC'
/SETSW ON=(4,5),OFF=(15)
/SYSFILE SYSDTA=(SYSCMD)
/WR-TEXT '
/WR-TEXT '*****'
/WR-TEXT '*** Reorganization of ISAM files ***'
/WR-TEXT '*****'
/WR-TEXT '
/WR-TEXT 'Name of file to be reorganized:'
/REMARK &FNAME
/EXEC $EDT
@GET '&FNAME' N
@CON
@IF N:@IF N D:@SY'COPY &FNAME,&FNAME..BEFORE,SAME'
@CON
@IF N : @IF N D : @SAVE 0
@CON
@IF E : @SETSW ON=15
@IF D : @SETSW ON=15
@HALT
/SKIP .ERR, ON=(15)
/TCHNG OFLOW=NO

```

```

/WR-TEXT '&(#NEWIMAG)'
/TCHNG OFLOW=ACK
/WR-TEXT ' '
/WR-TEXT 'File &FNAME has been reorganized.'
/WR-TEXT ' '
/WR-TEXT 'Space required before:'
/FS &FNAME..BEFORE
/WR-TEXT ' '
/ERASE &FNAME..BEFORE
/WR-TEXT 'Space required now:'
/FS &FNAME
/SKIP .EOP, OFF=(15)
/.ERR REMARK
/WR-TEXT ' '
/WR-TEXT '***** Error during file processing *****'
/WR-TEXT '** REORGANIZATION HAS NOT BEEN PERFORMED ! **'
/WR-TEXT ' '
/.EOP STEP
/SETSW OFF=(4,5,15)
/SYSFILE SYSDTA=(PRIMARY)
/ENDP

```

```

% CVR1000 CONVERSION TERMINATED WITH: _____ (3)
% CVR1001          FAILURES:    0
% CVR1002          ERRORS   :    0
% CVR1003          WARNINGS:    0
% CVR1004          NOTES    :    0

```

- (1) The operand values set for the SDF-CONV run are listed.
- (2) All command records in the input procedure are listed. If an error occurs during the conversion, the appropriate message is written beneath the command record concerned. This is not the case in the example, i.e. SDF-CONV was able to assign an SDF command to each ISP command as well as converting the input procedure into an S procedure without errors.
- (3) The final part of the log is identical to the listing output to SYSOUT.

Output procedure SDF.SDFP.DO.REORG

The conversion result is an S procedure.

```

/SET-PROCEDURE-OPTIONS LOGGING-ALLOWED=NO, INTERRUPT-ALLOWED=YES,- _____ (1)
/   DATA-ESCAPE-CHAR=STD, SYSTEM-FILE-CONTEXT=OWN, DATA-ERROR-HANDLING=-
/   NO; BEGIN-PARAMETER-DECLARATION; DECLARE-PARAMETER NAME=FNAME, TYPE=-
/   ANY, INIT=*PROMPT; END-PARAMETER-DECLARATION
/SET-JV-LINK   JV-NAME=#NEWIMAG
/MODIFY-JV   JV-CONTENTS=(JV-NAME=#NEWIMAG), SET-VALUE=X'0C'
/MODIFY-JOB-SWITCHES  ON=(4,5), OFF=15
/ASSIGN-SYSDTA  TO=*SYSCMD
/WRITE-TEXT ' ' _____ (2)
/WRITE-TEXT '*****'
/WRITE-TEXT '***      Reorganization of ISAM files      ***'
/WRITE-TEXT '*****'
/WRITE-TEXT ' '
/WRITE-TEXT 'Name of file to be reorganized:'
/REMARK '&(FNAME)'
/START-PROGRAM FROM-FILE=$EDT
@GET '&(FNAME)' N
@CON
@IF N:@IF N D:@SY'COPY &(FNAME),&(FNAME).BEFORE,SAME'
@CON
@IF N : @IF N D : @SAVE 0
@CON
@IF E : @SETSW ON=15
@IF D : @SETSW ON=15
@HALT
/SKIP-COMMANDS TO-LABEL=ERR,IF=JOB-SWITCHES(ON=15)
/MODIFY-TERMINAL-OPTIONS  OVERFLOW-CONTROL=NO-CONTROL
/WRITE-TEXT '&(JV('#NEWIMAG'))'
/MODIFY-TERMINAL-OPTIONS  OVERFLOW-CONTROL=USER-ACKNOWLEDGE
/WRITE-TEXT ' '
/WRITE-TEXT 'File &(FNAME) has been reorganized.'
/WRITE-TEXT ' '
/WRITE-TEXT 'Space required before:'
/SHOW-FILE-ATTRIBUTES  FILE-NAME=&(FNAME).BEFORE,SELECT=BY-ATTRIBUTES
/WRITE-TEXT ' '
/DELETE-FILE  FILE-NAME=&(FNAME).BEFORE,SELECT=BY-ATTRIBUTES
/WRITE-TEXT 'Space required now:'
/SHOW-FILE-ATTRIBUTES  FILE-NAME=&(FNAME),SELECT=BY-ATTRIBUTES
/SKIP-COMMANDS TO-LABEL=EOP,IF=JOB-SWITCHES(OFF=15)
/.ERR REMARK
/WRITE-TEXT ' '
/WRITE-TEXT '***** Error during file processing *****'
/WRITE-TEXT '** REORGANIZATION HAS NOT BEEN PERFORMED ! **'
/WRITE-TEXT ' '

```

```

/.EOP SET-JOB-STEP
/MODIFY-JOB-SWITCHES OFF=(4,5,15)
/ASSIGN-SYSDTA TO=*PRIMARY
/IF-BLOCK-ERROR;EXIT-PROCEDURE ERROR=YES;END-IF;EXIT-PROCEDURE

```

- (1) The SDF commands for the S procedure generated from the command PROC N,... are represented in block format, i.e. the commands are written continuously, separated by semicolon.
- (2) The WRITE-TEXT command is an SDF command and therefore need not be converted. However, SDF-CONV expands the input command to its full input format.

Example 2:

The procedure DO.LIB.CONTENTES lists all the elements contained in a library. The most important data for each element is written to a list. The conversion of the PRINT and DO commands in this procedure is subject to restrictions. SDF-CONV reports a NOTE for each of these commands. The PRINT command is not converted because its first operand is a procedure parameter; conversion of the DO command cannot be guaranteed due to the variety of context-dependent conversion possibilities. SDF-CONV enters conversion proposals for the DO commands in the output procedure.

Input procedure DO.LIB.CONTENTES

```

/PROC N,(&LIBNAME=,&LISTOUT=,&PRINT=,&INPUT),SUBDTA=&
/DCLJV #NEWIMAG
/SETJV #NEWIMAG,X'OC'
/WR-T '&(#NEWIMAG)'
/WR-T ' '
/WR-T ' '
/WR-T '*****'
/WR-T '**** Listing of elements stored in a library ****'
/WR-T '*****'
/WR-T ' '
/WR-T 'Name of the library : LIBNAME'
/REMARK input &LIBNAME
/SYSFILE SYSOUT=*DUMMY
/FS &LIBNAME
/SYSFILE SYSOUT=()
/SKIP .LINK
/STEP "Error with FSTATUS"
/SYSFILE SYSOUT=()
/WR-T ' '
/WR-T ' '

```

```

/WR-T 'Input error !'
/WR-T 'Library &LIBNAME does not exist/cannot be opened'
/SKIP .E
/.LINK REMARK Edit the library
/WR-T 'Name of the output list                : LISTOUT'
/WR-T 'If the output list already exists      '
/WR-T 'the data is added at the end of the file.      '
/WR-T 'Is the list to be printed (Y/N) ?        : PRINT  '
/REMARK Input &LISTOUT and &PRINT
/SETSW ON=(1)
/SYSFILE SYSDTA=(SYSCMD)
/SYSFILE SYSLST=(&LISTOUT,EXTEND)
/SYSFILE SYSOUT=*DUMMY
/FILE &LIBNAME,LINK=LIB001
/EXEC $LMS
PAR INFO=SUMMARY
PRT (LST)
LST* (1)
END
/STEP
/SYSFILE SYSOUT=( )
/SYSFILE SYSLST=( )
/SYSFILE SYSDTA=( )
/SETSW OFF=(1)
/REL LIB001
/WR-T ' '
/WR-T ' '
/WR-T '**** Editing completed.'
/WR-T '**** The result is contained in the list &LISTOUT.'
/SKIP "Branch dependent on the PRINT variables" .&PRINT
/WR-T ' '
/.J REMARK Print the list
/PRINT &LISTOUT,PNAME=INF,ENDNO=2048
/WR-T 'List &LISTOUT being printed.'
/.N REMARK
/WR-T 'Edit another library or terminate the procedure ?'
/SKIP "Input: A (edit) or E (terminate)" .&INPUT
/.A REMARK Call the procedure again
/DO DO.LIB.CONTENTS
/STEP
/.E REMARK End
/WR-T ' '
/WR-T '**** Procedure terminated'
/ENDP

```


SDF-CONV run

```
/start-sdf-conv _____ (1)
% BLS0517 MODULE 'CVRMAIN' LOADED
% CVR0001 SDF-CONV VERSION 'V03.0B80' STARTED
% CVR0060 SDF-CONV VERSION 'V03.0B80' READY
//convert from-file=do.lib.contents,to-file=sdf.do.lib.contents,doc=*min (2)
% CVR1000 CONVERSION TERMINATED WITH:
% CVR1001          FAILURES:    0
% CVR1002          ERRORS   :    0
% CVR1003          WARNINGS:    0
% CVR1004          NOTES    :    2
//end _____ (3)
```

- (1) The SDF-CONV utility routine is started.
- (2) The CONVERT statement assigns the input and output procedures, as well as defining the documentation level. All other operands of the CONVERT statement are given the default value. Once processing has been completed, SDF-CONV reports that conversion was terminated with two notes (NOTE). This overview is output to SYSOUT.
- (3) SDF-CONV is terminated with the END statement.

Conversion log (SYSLST)

***** BS2000 COMMAND CONVERTER : SDF-CONV V3 ***** page 1

```
% CVR1010 OPTIONS IN EFFECT: _____ (1)
% CVR1011 FROM-FILE = DO.LIB.CONTENTS
% CVR1012 EXPECT-CONTINUATION = *NEW-MODE
% CVR1013 PARAMETER-LINES = *CONVERT
% CVR1014 TO-FILE = SDF.DO.LIB.CONTENTS
% CVR1015 PRODUCE-CONTINUATION = *NEW-MODE
% CVR1016 REPLACE-OLD-FILE = *YES
% CVR1017 SYSTEM-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.U-CMD
% CVR1018 GROUP-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.S-CMD
% CVR1019 UNCHANGED-CMD = *NONE
% CVR1020 DOCUMENTATION = *MINIMUM
% CVR1021 TARGET-VERSION = *OSD-V2
% CVR1022 OUTPUT-FORM = *STD
% CVR1023 PROCEDURE-FORMAT = *SAME(BLOCK-INPUT=*NOT-ALLOWED)
```

***** BS2000 COMMAND CONVERTER : SDF-CONV V3 ***** page 2

```
/PROC N,(&LIBNAME=,&LISTOUT=,&PRINT=,&INPUT),SUBDTA=& _____ (2)
/DCLJV #NEWIMAG
/SETJV #NEWIMAG,X'OC'
/WR-T '&(#NEWIMAG) '
/WR-T ' '
/WR-T ' '
/WR-T '*****'
/WR-T '**** Listing of elements stored in a library ****'
/WR-T '*****'
/WR-T ' '
/WR-T 'Name of the library : LIBNAME'
/REMARK input &LIBNAME
/SYSFILE SYSOUT=*DUMMY
/FS &LIBNAME
/SYSFILE SYSOUT=()
/SKIP .LINK
/STEP "Error with FSTATUS"
/SYSFILE SYSOUT=()
/WR-T ' '
/WR-T ' '
/WR-T 'Input error !'
/WR-T 'Library &LIBNAME does not exist/cannot be opened'
/SKIP .E
/.LINK REMARK Edit the library
/WR-T 'Name of the output list : LISTOUT'
/WR-T 'If the output list already exists '

```

```

/WR-T 'the data is added at the end of the file.          '
/WR-T 'Is the list to be printed (Y/N) ?                : PRINT '
/REMARK Input &LISTOUT and &PRINT
/SETSW ON=(1)
/SYSFILE SYSDDTA=(SYSCMD)
/SYSFILE SYSLST=(&LISTOUT,EXTEND)
/SYSFILE SYSOUT=*DUMMY
/FILE &LIBNAME,LINK=LIB001
/EXEC $LMS
PAR INFO=SUMMARY
PRT (LST)
LST* (1)
END
/STEP
/SYSFILE SYSOUT=()
/SYSFILE SYSLST=()
/SYSFILE SYSDDTA=()
/SETSW OFF=(1)
/REL LIB001
/WR-T ' '
/WR-T ' '
/WR-T '**** Editing completed.'
/WR-T '**** The result is contained in the list &LISTOUT.'
/SKIP "Branch dependent on the PRINT variables" .&PRINT
/WR-T ' '
/.J REMARK Print the list
/PRINT &LISTOUT,PNAME=INF,ENDNO=2048
% CVRF040 INPUT COMMAND NOT CONVERTED _____ (3)
/WR-T 'List &LISTOUT being printed.'
/.N REMARK
/WR-T 'Edit another library or terminate the procedure ?'
/SKIP "Input: A (edit) or E (terminate)" .&INPUT
/.A REMARK Call the procedure again
/DO DO.LIB.CONTENTS
% CVRF061 COMMAND CAN NEVER BE CONVERTED TO A SECURED SDF FORM; ORIGINAL
INPUT REMAINS VALID _____ (4)
/STEP
/.E REMARK End
/WR-T ' '
/WR-T '**** Procedure terminated'
/ENDP

% CVR1000 CONVERSION TERMINATED WITH: _____ (5)
% CVR1001          FAILURES:    0
% CVR1002          ERRORS   :    0
% CVR1003          WARNINGS:    0
% CVR1004          NOTES    :    2

```

- (1) The operand values set for the SDF-CONV run are listed.
- (2) All command records in the input procedure are output.
- (3) SDF-CONV issues a NOTE pointing out that the PRINT command could not be converted as its first positional operand is a procedure parameter. For further details see [section "Restrictions" on page 250](#).
- (4) SDF-CONV issues a NOTE pointing out that the DO command could not be converted for reasons of context-dependency. For further details see [section "Restrictions" on page 250](#).
- (5) The last part of the log is identical with the output to SYSOUT.

Output procedure SDF.DO.LIB.CONTENTS

```

/BEGIN-PROCEDURE LOGGING=NO,PARAMETERS=YES(PROCEDURE=PARAMETERS=(
/&LIBNAME=,-
/&LISTOUT=,-
/&PRINT=,-
/&INPUT),ESCAPE-CHARACTER='&')
/SET-JV-LINK JV-NAME=#NEWIMAG
/MODIFY-JV JV-CONTENTS=(JV-NAME=#NEWIMAG),SET-VALUE=X'OC'
/WRITE-TEXT '&(#NEWIMAG)' _____ (1)
/WRITE-TEXT ' '
/WRITE-TEXT ' '
/WRITE-TEXT '*****'
/WRITE-TEXT '**** Listing of elements stored in a library ****'
/WRITE-TEXT '*****'
/WRITE-TEXT ' '
/WRITE-TEXT 'Name of the library : LIBNAME'
/REMARK input &LIBNAME
/ASSIGN-SYSOUT TO=*DUMMY
/SHOW-FILE-ATTRIBUTES FILE-NAME=&LIBNAME,SELECT=BY-ATTRIBUTES
/ASSIGN-SYSOUT TO=*PRIMARY
/SKIP-COMMANDS TO-LABEL=LINK
/"*****" REMARK '/STEP "Error with FSTATUS"' _____ (2)
/SET-JOB-STEP
/ASSIGN-SYSOUT TO=*PRIMARY
/WRITE-TEXT ' '
/WRITE-TEXT ' '
/WRITE-TEXT 'Input error !'
/WRITE-TEXT 'Library &LIBNAME does not exist/cannot be opened'
/SKIP-COMMANDS TO-LABEL=E
/.LINK REMARK Edit the library
/WRITE-TEXT 'Name of the output list : LISTOUT'
/WRITE-TEXT 'If the output list already exists '
/WRITE-TEXT 'the data is added at the end of the file. '

```

```

/WRITE-TEXT 'Is the list to be printed (Y/N) ?           : PRINT '
/REMARK Input &LISTOUT and &PRINT
/MODIFY-JOB-SWITCHES ON=1
/ASSIGN-SYSDTA TO=*SYSCMD
/ASSIGN-SYSLST TO=&LISTOUT,OPEN-MODE=EXTEND
/ASSIGN-SYSOUT TO=*DUMMY
/CREATE-FILE FILE-NAME=&LIBNAME,SUPPRESS-ERRORS=*FILE-EXISTING
/ADD-FILE-LINK LINK-NAME=LIB001,FILE-NAME=&LIBNAME,SUPPORT=*DISK
/START-PROGRAM FROM-FILE=$LMS
PAR INFO=SUMMARY
PRT (LST)
LST* (1)
END
/SET-JOB-STEP
/ASSIGN-SYSOUT TO=*PRIMARY
/ASSIGN-SYSLST TO=*PRIMARY
/ASSIGN-SYSDTA TO=*PRIMARY
/MODIFY-JOB-SWITCHES OFF=1
/REMOVE-FILE-LINK LINK-NAME=LIB001
/WRITE-TEXT ' '
/WRITE-TEXT ' '
/WRITE-TEXT '**** Editing completed.'
/WRITE-TEXT '**** The result is contained in the list &LISTOUT.'
/"*****" REMARK '/SKIP "Branch dependent on the PRINT variables" .&PRIN- (3)
/T'
/SKIP-COMMANDS TO-LABEL=&PRINT
/WRITE-TEXT ' '
/.J REMARK Print the list
/PRINT &LISTOUT,PNAME=INF,ENDNO=2048 _____ (4)
/WRITE-TEXT 'List &LISTOUT being printed.'
/.N REMARK
/WRITE-TEXT 'Edit another library or terminate the procedure ?'
/"*****" REMARK '/SKIP "Input: A (edit) or E (terminate)" .&INPUT' _____ (5)
/SKIP-COMMANDS TO-LABEL=&INPUT
/.A REMARK Call the procedure again
/"*****" REMARK '/CALL-PROCEDURE NAME=DO.LIB.CONTENTS' _____ (6)
/DO DO.LIB.CONTENTS
/SET-JOB-STEP
/.E REMARK End
/WRITE-TEXT ' '
/WRITE-TEXT '**** Procedure terminated'
/EXIT-PROCEDURE

```

- (1) The WRITE-TEXT command is an SDF command and therefore need not be converted. However, SDF-CONV expands the input command to its full input format.
- (2) The command record with the inline comment is written to a comment line preceding the command line with the conversion result. This ensures that the information in the inline comment is not lost during conversion.
(In this case, DOC=*MIN/*MAX must be set.)
- (3) see (2).
- (4) The command record with the unconverted PRINT command is taken over from the input procedure.
- (5) see (2).
- (6) The command record with the unconverted DO command is taken over from the input procedure. At the same time, however, SDF-CONV suggests what the conversion of this DO command could look like. This proposal is written as a comment line preceding the unconverted command record.
(In this case, DOC=*MIN/*MAX must be set.)

Example 3

The procedure PROC.ASSXT is a small auxiliary procedure that is to speed up processing (system file assignment, assembler call) in the event of frequent access to the ASSEMBH assembler. The procedure was written as a non-S procedure using the SDF command language. No restrictions apply to its conversion into an S procedure.

Input procedure PROC.ASSXT

```
/BEGIN-PROC A,P-P=(&FILE),ESC-CHAR='&'
/REMARK **** Call ASSEMBH ****
/DEL-FI LST.ASEMBH
/SET-JOB-STEP
/ASS-SYSDTA *SYSCMD
/DEL-SYS-FI "from previous ASSEMBH calls" *OMF
/START-PROG $ASSEMBH
//COMPILE SOURCE=&FILE,LISTING=PAR(OUTPUT=LST.ASEMBH),-
//TEST-SUPPORT=YES
//END
/SET-JOB-STEP
/ASS-SYSDTA *P
/END-PROC
```

SDF-CONV run

```
/start-sdf-conv _____ (1)
% BLS0517 MODULE 'CVRMAIN' LOADED
% CVR0001 SDF-CONV VERSION 'V03.0B80' STARTED
% CVR0060 SDF-CONV VERSION 'V03.0B80' READY
//convert from-file=proc.assxt,to-file=sdfp.proc.assxt,doc=*min,- _____ (2)
//procedure-format=*s-proc(prog-inp=*data)
% CVR1000 CONVERSION TERMINATED WITH:
% CVR1001 FAILURES: 0
% CVR1002 ERRORS : 0
% CVR1003 WARNINGS: 0
% CVR1004 NOTES : 0
//end _____ (3)
```

- (1) The SDF-CONV utility routine is started.
- (2) The CONVERT statement assigns the input and output procedures, as well as defining the documentation level. The procedure is to be converted into an S procedure. All other operands of the CONVERT statement retain their default value. Once processing has been completed, SDF-CONV reports that conversion was terminated without warnings or errors. This overview is output to SYSOUT.
- (3) SDF-CONV is terminated with the END statement.

Conversion log (SYSLST)

***** BS2000 COMMAND CONVERTER : SDF-CONV V3 ***** page 1

```
% CVR1010 OPTIONS IN EFFECT: _____ (1)
% CVR1011 FROM-FILE = PROC.ASSXT
% CVR1012 EXPECT-CONTINUATION = *NEW-MODE
% CVR1013 PARAMETER-LINES = *CONVERT
% CVR1014 TO-FILE = SDFP.PROC.ASSXT
% CVR1015 PRODUCE-CONTINUATION = *NEW-MODE
% CVR1016 REPLACE-OLD-FILE = *YES
% CVR1017 SYSTEM-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.U-CMD
% CVR1018 GROUP-SYNTAX-FILE = $.SYSSDF.SDF-CONV.030.USER.S-CMD
% CVR1019 UNCHANGED-CMD = *NONE
% CVR1020 DOCUMENTATION = *MINIMUM
% CVR1021 TARGET-VERSION = *OSD-V2
% CVR1022 OUTPUT-FORM = *STD
% CVR1023 PROCEDURE-FORMAT = *S-PROCEDURE(CMD-
FORMAT=*SDF,PROGRAM-INPUT=*DATA(EXCEPT-AFTER-CMD=*NONE,WARNING=*NO),BLOCK-
INPUT=*NOT-ALLOWED)
```

***** BS2000 COMMAND CONVERTER : SDF-CONV V3 ***** page 2

```
/BEGIN-PROC A,P-P=(&FILE),ESC-CHAR='&' _____ (2)
/REMARK **** Call ASSEMBH ****
/DEL-FI LST.ASEMBH
/SET-JOB-STEP
/ASS-SYSDTA *SYSCMD
/DEL-SYS-FI "from previous ASSEMBH calls" *OMF
/START-PROG $ASSEMBH
//COMPILE SOURCE=&FILE,LISTING=PAR(OUTPUT=LST.ASEMBH),-
//TEST-SUPPORT=YES
//END
/SET-JOB-STEP
/ASS-SYSDTA *P
/END-PROC
```

```
% CVR1000 CONVERSION TERMINATED WITH: _____ (3)
% CVR1001 FAILURES: 0
% CVR1002 ERRORS : 0
% CVR1003 WARNINGS: 0
% CVR1004 NOTES : 0
```

- (1) The operand values set for the SDF-CONV run are listed.
- (2) All command records in the input procedure are output.

- (3) The last part of the log is identical with the output to SYSOUT.

Output procedure SDFP.PROC.ASSXT

```

/SET-PROCEDURE-OPTIONS LOGGING-ALLOWED=YES, INTERRUPT-ALLOWED=YES,-
/      DATA-ESCAPE-CHAR=STD, SYSTEM-FILE-CONTEXT=OWN, DATA-ERROR-HANDLING=-
/      NO
/BEGIN-PARAMETER-DECLARATION
/DECLARE-PARAMETER NAME=FILE, TYPE=ANY, INIT=*PROMPT
/END-PARAMETER-DECLARATION
/REMARK '**** CALL ASSEMBH ****'
/DELETE-FILE LST.ASEMBH
/SET-JOB-STEP
/ASSIGN-SYSDTA TO=*SYSCMD
/"*****" REMARK '/DEL-SYS-FI "from previous ASSEMBH calls" *OMF' ----- (1)
/DELETE-SYSTEM-FILE "from previous ASSEMBH calls" *OMF
/START-PROGRAM $ASSEMBH
//COMPILE SOURCE=&(FILE), LISTING=PAR(OUTPUT=LST.ASEMBH), TEST-SUPPORT=-
//      YES
//END
/SET-JOB-STEP
/ASSIGN-SYSDTA TO=*P
/IF-BLOCK-ERROR; END-IF
/END-PROCEDURE

```

- (1) The command record with the inline comment is written to a comment line preceding the command line with the conversion result. This ensures that the information in the inline comment is not lost during conversion.
(In this case, DOC=*MIN/*MAX must be set.)

13.7 Assignment of ISP to SDF commands

13.7.1 User commands

ISP command	SDF command
ABEND	EXIT-JOB
ABORT	CANCEL-PROCEDURE
ADD-ISAM-POOL-LINK	ADD-ISAM-POOL-LINK
APPLICATION	SET-DCAM-APPLICATION-LINK
AUDIT	RESUME-HARDWARE-AUDIT
AUDIT	SHOW-HARDWARE-AUDIT
AUDIT	START-HARDWARE-AUDIT
AUDIT	STOP-HARDWARE-AUDIT
BCNTRL	MODIFY-MSG-OPTIONS
BREAK	HOLD-PROGRAM
CALL	CALL-PROCEDURE
CANCEL	CANCEL-JOB
CATALOG	CREATE-FILE
CATALOG	CREATE-FILE-GROUP
CATALOG	MODIFY-FILE-ATTRIBUTES
CATALOG	MODIFY-FILE-GROUP-ATTRIBUTES
CATJV	CREATE-JV
CATJV	MODIFY-JV-ATTRIBUTES
CHANGE	CHANGE-FILE-LINK
CONNECTION	REMOVE-DCAM-CONNECTION-LINK
CONNECTION	SET-DCAM-CONNECTION-LINK
COPY	COPY-FILE
CREATE-ISAM-POOL	CREATE-ISAM-POOL
DCLJV	SET-JV-LINK
DELETE-ISAM-POOL	DELETE-ISAM-POOL
DELON	REMOVE-CJC-ACTION
DROP	UNLOCK-FILE-LINK
ENDON	END-CJC-ACTION

Table 15: Assignment of ISP to SDF user commands

ISP command	SDF command
ENDP	END-PROCEDURE
ENDP	EXIT-PROCEDURE
ENTER	ENTER-JOB
EOF	EOF
ERAJV	DELETE-JV
ERASE	DELETE-FILE
ERASE	DELETE-FILE-GENERATION
ERASE	DELETE-FILE-GROUP
ERASE	DELETE-SYSTEM-FILE
ERASE	EXPORT-FILE
ESCAPE	HOLD-PROCEDURE
EXECUTE	START-PROGRAM
FILE	CREATE-FILE
FILE	CREATE-FILE-GENERATION
FILE	CREATE-TAPE-SET
FILE	DELETE-TAPE-SET
FILE	DELETE-TAPE-SET
FILE	EXTEND-TAPE-SET
FILE	IMPORT-FILE
FILE	MODIFY-FILE-ATTRIBUTES
FILE	MODIFY-FILE-GENERATION-SUPPORT
FILE	SET-FILE-LINK
INTR	SEND-MESSAGE
LOAD	LOAD-PROGRAM
LOGOFF	EXIT-JOB
LOGOFF	LOGOFF
LOGON	SET-LOGON-PARAMETERS
MODIFY-JOB	MODIFY-JOB
MODIFY-JV-CONDITIONALLY	MODIFY-JV-CONDITIONALLY
MODIFY-MSG-ATTRIBUTES	MODIFY-MSG-ATTRIBUTES
MRSSTA	SHOW-MRS-CONNECTIONS
MSGCONTROL	MODIFY-MSG-FILE-ASSIGNMENT

Table 15: Assignment of ISP to SDF user commands

ISP command	SDF command
ON	ADD-CJC-ACTION
OPTION	MODIFY-JOB-OPTIONS
OPTION	MODIFY-TEST-OPTIONS
PARAMETER	PARAMETER
PASSWORD	ADD-PASSWORD
PASSWORD	REMOVE-PASSWORD
PAUSE	SEND-MESSAGE
PRINT	PRINT-FILE
PRINT	WRITE-SPOOL-TAPE
PRIORITY	CHANGE-TASK-PRIORITY
PRIORITY	MODIFY-JOB
PROCEDURE	BEGIN-PROCEDURE
PSWORD	MODIFY-USER-PROTECTION
PUNCH	WRITE-DISKETTE
PUNCH	WRITE-POOLER-TAPE
RDIR	REDIRECT-REMOTE-OUTPUT
RDTFT	SHOW-FILE-LINK
RELEASE	REMOVE-FILE-LINK
RELEASE-SUBSYSTEM-SPACE	RELEASE-SUBSYSTEM-SPACE
REMARK	REMARK
REMOVE-ISAM-POOL-LINK	REMOVE-ISAM-POOL-LINK
REMOVE-JV-LINK	REMOVE-JV-LINK
REMOVE-SPOOL-CHARACTER-SET	REMOVE-SPOOL-CHARACTER-SET
REMOVE-SPOOL-DEVICE	REMOVE-SPOOL-DEVICE
REMOVE-SPOOL-FORM	REMOVE-SPOOL-FORM
RESTART	RESTART-PROGRAM
RESUME	RESUME-PROGRAM
RFAEND	REMOVE-RFA-CONNECTION
RFASTART	SET-RFA-CONNECTION
RTI	RESUME-PROCEDURE
SDVC	MODIFY-PRINTER-OUTPUT-STATUS
SDVC	START-PRINTER-OUTPUT

Table 15: Assignment of ISP to SDF user commands

ISP command	SDF command
SDVC	STOP-PRINTER-OUTPUT
SECURE-RESOURCE-ALLOCATION	SECURE-RESOURCE-ALLOCATION
SHOW-DISK-DEFAULTS	SHOW-DISK-DEFAULTS
SHOW-DISK-STATUS	SHOW-DISK-STATUS
SHOW-FILE	SHOW-FILE
SHOW-JOB-CLASS	SHOW-JOB-CLASS
SHOW-JV-LINK	SHOW-JV-LINK
SHOW-MOUNT-PARAMETER	SHOW-MOUNT-PARAMETER
SHOW-MSG-DEFAULTS	SHOW-MSG-FILE-ASSIGNMENT
SHOW-PCS-OPTION	SHOW-PCS-OPTION
SHOW-RESOURCE-ALLOCATION	SHOW-RESOURCE-ALLOCATION
SHOW-SPOOL-CHARACTER-SETS	SHOW-SPOOL-CHARACTER-SETS
SHOW-SPOOL-DEVICES	SHOW-SPOOL-DEVICES
SHOW-SPOOL-FORMS	SHOW-SPOOL-FORMS
SHOW-SPOOL-PARAMETERS	SHOW-SPOOL-PARAMETERS
SHOW-SS-STATUS	SHOW-SUBSYSTEM-STATUS
SHOW-TAPE-STATUS	SHOW-TAPE-STATUS
SHOW-USER-ATTRIBUTES	SHOW-USER-ATTRIBUTES
SKIP	SKIP-COMMANDS
SKIPJV	SKIP-COMMANDS
SKIPUS	SKIP-COMMANDS
SPARAM	MODIFY-SPOOL-OUT-OPTIONS
SQUC	HOLD-SPOOL-OUT
SQUC	RESUME-SPOOL-OUT
STAJV	SHOW-JV-ATTRIBUTES
STAM	SHOW-MASTER-CATALOG-ENTRY
STATUS	SHOW-JOB-STATUS
STATUS	SHOW-SYSTEM-STATUS
STATUS	SHOW-USER-STATUS
STEP	SET-JOB-STEP
SYSFILE	ASSIGN-SYSDTA
SYSFILE	ASSIGN-SYSIPT

Table 15: Assignment of ISP to SDF user commands

ISP command	SDF command
SYSFILE	ASSIGN-SYSLST
SYSFILE	ASSIGN-SYSOPT
SYSFILE	ASSIGN-SYSOUT
SYSFILE	REMOVE-TASKLIB
SYSFILE	SET-TASKLIB
SYSTATUS	SHOW-SYSTEM-FILE-ASSIGNMENTS
TCHNG	MODIFY-TERMINAL-OPTIONS
TYPE	SEND-MESSAGE
VERIFY	CHECK-FILE-CONSISTENCY
VERIFY	REMOVE-FILE-ALLOCATION-LOCKS
VERIFY	REPAIR-DISK-FILES
WAIT	WAIT-EVENT
WHEN	WAIT-EVENT
WRITE-ACCOUNTING-RECORD	WRITE-ACCOUNTING-RECORD

Table 15: Assignment of ISP to SDF user commands

13.7.2 System administration commands

ISP command	SDF command
ADD-SPOOL-DEVICE	ADD-SPOOL-DEVICE
ADD-SPOOL-FORM	ADD-SPOOL-FORM
ADD-SUBSYSTEM	ADD-SUBSYSTEM
AUDIT	HOLD-HARDWARE-AUDIT
AUDIT	RESUME-HARDWARE-AUDIT
AUDIT	SHOW-HARDWARE-AUDIT
AUDIT	START-HARDWARE-AUDIT
AUDIT	STOP-HARDWARE-AUDIT
BIAS	MODIFY-SYSTEM-BIAS
CATALOG	CREATE-FILE
CATALOG	CREATE-FILE-GENERATION
CATALOG	CREATE-FILE-GROUP
CATALOG	MODIFY-FILE-ATTRIBUTES
CATALOG	MODIFY-FILE-GENERATION-SUPPORT
CATALOG	MODIFY-FILE-GROUP-ATTRIBUTES
CATEGORY	MODIFY-TASK-CATEGERIES
CATM	ADD-MASTER-CATALOG-ENTRY
CATM	MODIFY-MASTER-CATALOG-ENTRY
CHANGE-ACCOUNTING-FILE	CHANGE-ACCOUNTING-FILE
CHANGE-CONSLOG	CHANGE-CONSLOG-FILE
CHANGE-SERSLOG	CHANGE-SERSLOG-FILE
ERAM	REMOVE-MASTER-CATALOG-ENTRY
ERASE	DELETE-FILE
ERASE	DELETE-FILE-GENERATION
ERASE	DELETE-FILE-GROUP
ERASE	EXPORT-FILE
EXCAT	EXPORT-CATALOG
EXCAT	EXPORT-PUBLIC-VOLUME-SET
EXCAT	FORCE-CATALOG-EXPORT
FILE	CREATE-FILE
FILE	CREATE-FILE-GENERATION

Table 16: Assignment of ISP to SDF commands for system administration

ISP command	SDF command
FILE	CREATE-TAPE-SET, EXTEND-TAPE-SET
FILE	IMPORT-FILE
FILE	MODIFY-FILE-ATTRIBUTES
FILE	MODIFY-FILE-GENERATION-SUPPORT
FILE	SET-FILE-LINK
HOLD-JOB	HOLD-JOB
HOLD-JOB-CLASS	HOLD-JOB-CLASS
HOLD-PCS	HOLD-PCS
HOLD-SS	HOLD-SUBSYSTEM
IMCAT	IMPORT-PUBLIC-VOLUME-SET
IMPORT	CHECK-IMPORT-DISK-FILE
IMPORT	IMPORT-FILE
JOIN	ADD-USER
JOIN	MODIFY-USER-ATTRIBUTES
LOADAID	LOADAID
MODIFY-ACCOUNTING-PARAMETERS	MODIFY-ACCOUNTING-PARAMETERS
MODIFY-JOB-CLASS	MODIFY-JOB-CLASS
MODIFY-JOB-STREAM	MODIFY-JOB-STREAM
MODIFY-PCS-OPTION	MODIFY-PCS-OPTIONS
MODIFY-SPOOL-FORM	MODIFY-SPOOL-FORM
MODIFY-SPOOL-PARAMETERS	MODIFY-SPOOL-PARAMETERS
MSGCONTROL	MODIFY-MSG-FILE-ASSIGNMENT
MSGCONTROL	RESET-MSG-FILE-ASSIGNMENT
NCHOLD	HOLD-TASK
NCREL	RESUME-TASK
RDIR	REDIRECT-REMOTE-OUTPUT
RELEASE-JOB	RESUME-JOB
RELEASE-JOB-CLASS	RESUME-JOB-CLASS
RELEASE-JOB-STREAM	RESUME-JOB-STREAM
REMOVE-SPOOL-CHARACTER-SET	REMOVE-SPOOL-CHARACTER-SET
REMOVE-SPOOL-DEVICE	REMOVE-SPOOL-DEVICE
REMOVE-SPOOL-FORM	REMOVE-SPOOL-FORM

Table 16: Assignment of ISP to SDF commands for system administration

ISP command	SDF command
RESUME-PCS	RESUME-PCS
RESUME-SS	RESUME-SUBSYSTEM
RFD	START-DISKETTE-INPUT
RFD	STOP-DISKETTE-INPUT
SDVC	MODIFY-DISKETTE-OUTPUT-STATUS
SDVC	MODIFY-PRINTER-OUTPUT-STATUS
SDVC	MODIFY-TAPE-OUTPUT-STATUS
SDVC	START-DISKETTE-OUTPUT
SDVC	START-PRINTER-OUTPUT
SDVC	START-TAPE-OUTPUT
SDVC	START-TAPE-REPLAY
SDVC	STOP-DISKETTE-OUTPUT
SDVC	STOP-PRINTER-OUTPUT
SDVC	STOP-TAPE-OUTPUT
SDVC	STOP-TAPE-REPLAY
SET-DSSM-OPTIONS	SET-DSSM-OPTIONS
SET-PUBSET-ATTRIBUTES	SET-PUBSET-ATTRIBUTES
SET-REPLUG-READ-MARK	SET-REPLUG-READ-MARK
SEVER	LOCK-USER
SEVER	REMOVE-USER
SEVER	UNLOCK-USER
SHARE	ADD-SHARED-PROGRAM
SHOW-ACCOUNTING-STATUS	SHOW-ACCOUNTING-STATUS
SHOW-CONSLOG	SHOW-CONSLOG-ATTRIBUTES
SHOW-DISK-STATUS	SHOW-DISK-STATUS
SHOW-JOB-STREAM	SHOW-JOB-STREAM
SHOW-PCS-OPTION	SHOW-PCS-OPTIONS
SHOW-PUBSET-ATTRIBUTES	SHOW-PUBSET-ATTRIBUTES
SHOW-RESOURCE-REQUESTS	SHOW-RESOURCE-REQUESTS
SHOW-SERSLOG	SHOW-SERSLOG-STATUS
SHOW-SS-STATUS	SHOW-SUBSYSTEM-STATUS
SHOW-TRACE-STATUS	SHOW-TRACE-STATUS

Table 16: Assignment of ISP to SDF commands for system administration

ISP command	SDF command
SHOW-USER-ATTRIBUTES	SHOW-USER-ATTRIBUTES
SPMGT	SET-SPACE-SATURATION-LEVEL
SQUC	HOLD-SPOOLOUT
STAM	SHOW-MASTER-CATALOG-ENTRY
START-ACCOUNTING	START-ACCOUNTING
START-JOB-STREAM	START-JOB-STREAM
START-PCS	START-PCS
START-SERSLOG	START-SERSLOG
START-SS	START-SUBSYSTEM
START-TRACE	START-TRACE
STATUS	SHOW-JOB-STATUS
STATUS	SHOW-SYSTEM-STATUS
STATUS	SHOW-USER-STATUS
STOP-ACCOUNTING	STOP-ACCOUNTING
STOP-JOB-STREAM	STOP-JOB-STREAM
STOP-PCS	STOP-PCS
STOP-SERSLOG	STOP-SERSLOG
STOP-SS	STOP-SUBSYSTEM
STOP-TRACE	STOP-TRACE

Table 16: Assignment of ISP to SDF commands for system administration

13.8 Conversion of ISP and old SDF commands by SDF-CONV

Commands for conversion to SDF format are classified into user commands and system administration commands. The tables list the commands for conversion in alphabetical order, along with the equivalent SDF commands generated by SDF-CONV. In cases where no conversion is performed by SDF-CONV, the reason is outlined. Certain operand configurations cannot be converted. The commands concerned are identified by a reference to the page in the [section “Restrictions” on page 250ff](#), where the exceptions are explained.

13.8.1 User commands

Command for conversion	Converted to SDF command	Reason why not converted or notes
ABEND	EXIT-JOB	
ABORT	CANCEL-PROCEDURE	
ACCOUNT	-----	No one-to-one assignment possible
APPLICATION LINK=... (as only operand) other configurations	REMOVE-DCAM-APPLICATION-LINK SET-DCAM-APPLICATION-LINK	
AUDIT	-----	Conversion is runtime-dependent
BCNTRL	MODIFY-MSG-OPTIONS	
BREAK	HOLD-PROGRAM	
CALL	CALL-PROCEDURE	
CANCEL	CANCEL-JOB	(See page 261)
CATALOG	CREATE-FILE CREATE-FILE-GENERATION CREATE-FILE-GROUP IMPORT-FILE MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GROUP-ATTRIBUTES	(See page 264)
CATJV STATE=NEW STATE=UPDATE	CREATE-JV MODIFY-JV-ATTRIBUTES	(See page 261)
CHANGE	CHANGE-FILE-LINK	
CONNECTION LINK=... (as only operand) other configurations	REMOVE-DCAM-CONNECTION-LINK SET-DCAM-CONNECTION-LINK	
COPY	COPY-FILE	(See page 261)

Table 17: Conversion of user commands by SDF-CONV

Command for conversion	Converted to SDF command	Reason why not converted or notes
DATA	-----	No SDF equivalent
DCLJV	SET-JV-LINK	
DELON	REMOVE-CJC-ACTION	(See page 261)
DIAG	-----	Not an ISP command
DO	-----	Conversion is context-dependent; no automatic conversion, but conversion (CALL-PROCEDURE) in the output procedure (See page 254)
DROP	UNLOCK-FILE-LINK	
END	-----	No SDF equivalent
ENDON	END-CJC-ACTION	
ENDP	EXIT-PROCEDURE	
ENTER	ENTER-JOB	(See page 261)
EOF	EOF	
ERAJV	DELETE-JV	(See page 261)
ERASE only file name * file or *SYSfile POS=... VOLUME=...	DELETE-FILE DELETE-SYSTEM-FILE DELETE-FILE-GENERATION EXPORT-FILE	(See page 261)
ESCAPE (without operands)	HOLD-PROCEDURE	
EXECUTE	START-PROGRAM	(See page 261)
FILE	CREATE-FILE CREATE-FILE-GENERATION CREATE-TAPE-SET DELETE-TAPE-SET EXTEND-TAPE-SET IMPORT-FILE MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GENERATION-SUPPORT SET-FILE-LINK	(See page 256)
FSTATUS VTOC=YES	SHOW-FILE-ATTRIBUTES IMPORT-FILE SUPPORT=BY-FILE-NAME	(See page 261)
GETJV	SHOW-JV	(See page 261)
GETUS	SHOW-USER-SWITCHES	

Table 17: Conversion of user commands by SDF-CONV

Command for conversion	Converted to SDF command	Reason why not converted or notes
HELP INF=CD oder INF=C	HELP-MESSAGE-INFORMATION HELP-MESSAGE-INFORMATION INFORMATION=MAXIMUM	(See page 261) There is no SDF equivalent for these operand values, but they are converted nevertheless
HOLD	LOCK-FILE-LINK	
IMPORT	IMPORT-FILE SUPPORT=DISK(...)	(See page 261)
INTR	SEND-MESSAGE TO=PROGRAM	
JAC	WRITE-ACCOUNTING-RECORD USER-DATA=(...)	
LOAD	LOAD-PROGRAM	(See page 261)
LOGOFF	EXIT-JOB	(See page 261)
LOGON	SET-LOGON-PARAMETERS	
MARGIN	MODIFY-TERMINAL-OPTIONS	
MRSSTA	SHOW-MRS-CONNECTIONS	
ON	ADD-CJC-ACTION	
OPTION MSG=... MAXLST=... MAXOPT=... DUMP=... TESTPRIV=...	MODIFY-JOB-OPTIONS MODIFY-JOB-OPTIONS MODIFY-JOB-OPTIONS MODIFY-TEST-OPTIONS MODIFY-TEST-OPTIONS	(See page 261)
PARAMETER	-----	No SDF equivalent
PASSWORD nur Kennwort REL=YES	ADD-PASSWORD REMOVE-PASSWORD	(See page 261)
PAUSE	SEND-MESSAGE TO=OPERATOR (WAIT-RESPONSE=YES)	
PRINT TAPE=NO TAPE≠NO	PRINT-DOCUMENT WRITE-SPOOL-TAPE	(See pages 256 and 261)
PRINT-FILE	PRINT-DOCUMENT	
PRIORITY <1..9> <30..255> (<30..255>,EXPRESS)	MODIFY-JOB CHANGE-TASK-PRIORITY MODIFY-JOB ...(START=IMMEDIATE) CHANGE-TASK-PRIORITY	(See page 261)
PROCEDURE	BEGIN-PROCEDURE	
PSWORD	MODIFY-USER-PROTECTION	

Table 17: Conversion of user commands by SDF-CONV

Command for conversion	Converted to SDF command	Reason why not converted or notes
PUNCH DEVICE=CENTRAL, TAPE=NO DEVICE=CENTRAL, TAPE=POOLER DEVICE=DISKETTE DEVIN=... FORM=... HREC=... TRLNUM=..., HDRNUM=...	PUNCH-FILE ----- WRITE-DISKETTE WRITE-DISKETTE WRITE-DISKETTE WRITE-DISKETTE WRITE-DISKETTE TRAILER-EXIT-NUMBER	(See pages 258 and 261) Pooler tapes are not supported as of BS2000 V11 The operands refer to card readers, which are no longer supported; however, they are still converted
RDTFT	SHOW-FILE-LINK	(See page 261)
RELEASE	REMOVE-FILE-LINK	(See page 261)
REMARK	REMARK	
RESTART	RESTART-PROGRAM	(See page 261)
RESUME with operands without operands	----- RESUME-PROGRAM	Not an ISP command (see page 258)
RFAEND	REMOVE-RFA-CONNECTION	
RFASTART	SET-RFA-CONNECTION	
RTI	RESUME-PROCEDURE	
SECURE	SECURE-RESOURCE-ALLOCATION	(See pages 258 and 261)
SETJV	MODIFY-JV	(See page 261)
SETSW with operands without operands	MODIFY-JOB-SWITCHES SHOW-JOB-SWITCHES	(See page 261)
SETUS	MODIFY-USER-SWITCHES	
SET-FILE-LINK	ADD-FILE-LINK	
SET-SS-OPTIONS	SET-SUBSYSTEM-OPTIONS	(See page 262)
SKIP	SKIP-COMMANDS IF=NO-CONDITION	(See page 262)
SKIPJV	SKIP-COMMANDS IF=JOB-SWITCHES(...)	
SKIPUS	SKIP-COMMANDS IF=USER-SWITCHES(...)	
SPARAM	MODIFY-SPOOL-OUT-OPTIONS	(See page 262)
SQUC REL=	HOLD-SPOOL-OUT RESUME-SPOOL-OUT	(See page 262)
STAJV	SHOW-JV-ATTRIBUTES	(See page 262)

Table 17: Conversion of user commands by SDF-CONV

Command for conversion	Converted to SDF command	Reason why not converted or notes
STAM	SHOW-MASTER-CATALOG-ENTRY	
STATUS ENVIR / LIST / PROG BIAS / CATEGORY / SUMMARY other operands	SHOW-USER-STATUS SHOW-SYSTEM-STATUS SHOW-JOB-STATUS	(See page 262)
STEP	SET-JOB-STEP	
SYSFILE SYSDTA=... SYSIPT=... SYSLST=... SYSLSTn=... SYSOPT=... SYSOUT=... TASKLIB=... TASKLIB=(NO)	ASSIGN-SYSDTA TO= ASSIGN-SYSIPT ASSIGN-SYSLST TO= ASSIGN-SYSLST TO=*SYSLST-NUMBER(n) ASSIGN-SYSOPT ASSIGN-SYSOUT TO= SET-TASKLIB REMOVE-TASKLIB	(See pages 259 and 262)
SYSTATUS	SHOW-SYSTEM-FILE-ASSIGNMENTS	(See page 262)
TCHNG	MODIFY-TERMINAL-OPTIONS	(See pages 259 and 262)
TYPE	SEND-MESSAGE TO=OPERATOR (WAIT-RESPONSE=NO)	
VERIFY REPAIR=YES REPAIR=ABS REPAIR=NO	REPAIR-DISK-FILES FILE-STATUS=OPEN(...) REPAIR-DISK-FILES FILE-STATUS=ANY(...) REMOVE-FILE-ALLOCATION-LOCKS	(See page 262)
WAIT	WAIT-EVENT UNTIL=JV(...)	
WHEN	WAIT-EVENT UNTIL=USER-SWITCHES(...)	

Table 17: Conversion of user commands by SDF-CONV

13.8.2 System administration commands

Command for conversion	Converted to SDF command	Reason why not converted or notes
BIAS	MODIFY-SYSTEM-BIAS	
CANCEL , KILL	FORCE-JOB-CANCEL	
CATEGORY	MODIFY-TASK-CATEGORIES	
CATM STATE=NEW STATE=UPDATE	ADD-MASTER-CATALOG-ENTRY MODIFY-MASTER-CATALOG-ENTRY	(See page 262)
CREATE-SS	START-SUBSYSTEM	
DELETE-SS	STOP-SUBSYSTEM	
ERAM	REMOVE-MASTER-CATALOG-ENTRY	
EXCAT CANCEL FORCE END	CANCEL-CATALOG-EXPORT FORCE-CATALOG-EXPORT EXPORT-PUBLIC-VOLUME-SET	(See pages 259 and 262)
HOLD-SS	HOLD-SUBSYSTEM	
IMCAT	IMPORT-PUBLIC-VOLUME-SET IMPORT-CATALOG	(See pages 259 and 262)
JOIN	-----	Conversion is context-dependent; no automatic conversion, but conversion proposal (ADD-USER or MODIFY-USER) in the comment line (see page 254)
LOADAID	-----	No SDF equivalent
LOADAIDSYS	-----	No SDF equivalent
MSGCONTROL FILE=STD FILE≠STD	RESET-MSG-FILE-ASSIGNMENT MODIFY-MSG-FILE-ASSIGNMENT	(See page 262)
MODIFY-MSG-DEFAULTS	MODIFY-MSG-FILE-ASSIGNMENT	
MSGCONTROL		
NCHOLD	HOLD-TASK	
NCREL	RESUME-TASK	
RCARD	-----	Card readers are no longer supported
RDIR	REDIRECT-REMOTE-OUTPUT	
RELEASE-JOB	RESUME-JOB	
RELEASE-JOB-CLASS	RESUME-JOB-CLASS	
RELEASE-JOB-STREAM	RESUME-JOB-STREAM	
RESUME-SS	RESUME-SUBSYSTEM	

Table 18: Conversion of commands for system administration by SDF-CONV

Command for conversion	Converted to SDF command	Reason why not converted or notes
RFD USE=INPUT USE=NO	START-DISKETTE-INPUT STOP-DISKETTE-INPUT	(See pages 259 and 262)
SDVC	-----	No one-to-one assignment possible
SEVER RESET REMOVE SET	UNLOCK-USER REMOVE-USER LOCK-USER	(See page 262)
SHARE	ADD-SHARED-PROGRAM	
SHOW-SS-STATUS	SHOW-SUBSYSTEM-STATUS	
SPMGT	SET-SPACE-SATURATION-LEVEL	(See page 262)
SQUC	HOLD-SPOOLOUT	(See page 262)
STAM	SHOW-MASTER-CATALOG-ENTRY	(See page 262)
START-SS	START-SUBSYSTEM	
STATUS BIAS / CATEGORY / SUMMARY ENVIR / LIST / PROG other operands	SHOW-SYSTEM-STATUS SHOW-USER-STATUS SHOW-JOB-STATUS	
STOP-SS	STOP-SUBSYSTEM	

Table 18: Conversion of commands for system administration by SDF-CONV

13.9 SDF-CONV messages

SDF-CONV distinguishes between NOTE, WARNING, ERROR and FAILURE.

If an ERROR or FAILURE occurs, the error message is output to SYSOUT as well as to SYSLST. The SDF-CONV conversion run is then aborted.

If a NOTE or WARNING occurs, the error message appears only in the log (output to SYSLST). After the error message has been issued, SDF-CONV processes the next command line of the input procedure.

The SDF-CONV messages have the following format: CVRxxxx <message-text>

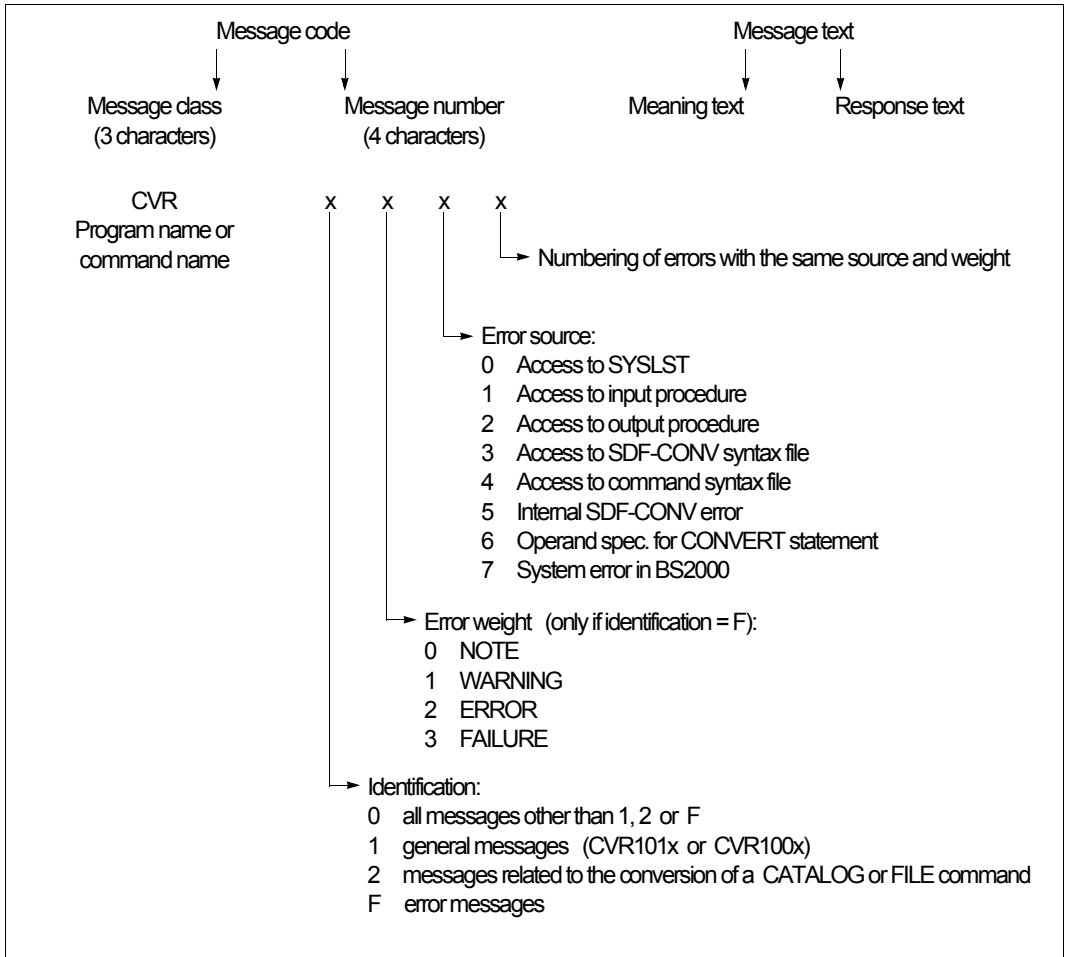


Figure 50: Structure of SDF-CONV messages

In a few cases, a message will be output followed by one or more other error messages that are intended to provide a more precise description of the cause of the error or its impact. Messages of this explanatory nature can have the format `CMDxxxx <message-text>`.

More detailed information concerning the cause of the error or possible remedial action can be obtained with the aid of the error code (`CVRxxx` or `CMDxxxx`) and the HELP-MSG-INFORMATION standard statement or command (see the “Commands” manual [1]).

14 DISPLAY

Version: **DISPLAY V1.1**

Privileges: **TSOS**

Depending on a user's privileges, many SHOW commands in BS2000/OSD only display the user's own data or the data of all users.

However, it can be undesirable to assign additional privileges to a user because these privileges as a rule include more extensive rights, e.g. the executability of commands.

The DISPLAY subsystem enables nonprivileged users to access the full data set of selected SHOW commands without the users having to be granted additional privileges. Only a few SHOW commands which are relevant to security are excluded from this.

To ensure security and compatibility, the standard configuration supplied contains only the basic procedures required. No function commands for accessing privileged data are supplied.

14.1 Product use

It is the task of the system administration to decide which users are to be granted access to which SHOW commands in accordance with the applicable requirements. The CREATE-DISPLAY-CMD command is used for this purpose. The system administration can use this command to create DISPLAY commands. A DISPLAY command is by default a copy of the SHOW command with the corresponding name, e.g. the DISPLAY-FILE-ATTRIBUTES command is a copy of the SHOW-FILE-ATTRIBUTES command. However, for the DISPLAY command the system administration can define which privileges permit its use and thus the output of the entire data set. By default DISPLAY commands are entered in the syntax file SYSSDF.DISPLAY.nnn which is supplied with the DISPLAY subsystem, but the system administration can also specify any other syntax file.

As DISPLAY commands are generally copies of SHOW commands, they must be created again each time the corresponding SHOW command is modified, e.g. after a change of version or a change to the command with SDF-A. You are therefore recommended to save the commands for generating DISPLAY commands as a procedure. This facilitates reconstruction after a version update.

Behavior of a DISPLAY command (DISPLAY-...) compared to that of the corresponding SHOW command (SHOW-...):

- Text output to SYSOUT or SYSLST is identical to the output which the corresponding SHOW command supplies when it is called under a user ID with the privileges TSOS and PRINT-SERVICE-ADMINISTRATION.
- During normal execution the return code and the spin-off behavior are the same as with the corresponding SHOW command. If internal errors occur during execution, the following command return code is generated:

(SC2)	SC1	Maincode	Meaning
	32	DSP0001	Internal error

- A DISPLAY command generates no output to the S variable stream SYSINF. The structured output to an S variable must be requested explicitly using the /EXECUTE-CMD CMD=(DISPLAY-...), STRUCT-OUT=... command.

Modifying DISPLAY commands

The system administration can process DISPLAY commands retroactively with SDF-A. This permits the functionality to be restricted compared to the corresponding SHOW command or operands or values to be released whose use is restricted in the SHOW command.

The following actions are permitted:

- **Deleting optional operands**
Optional operands of a DISPLAY command can be deleted completely. Internally the operands are assigned the default value which applies for the SHOW command.
- **Changing the privileges of operands and operand values:**
Individual privileges can be assigned to individual operands and operand values to restrict their use to particular users.
As long as a permitted input option is retained for all users, the standard privileges of an operand or operand value can be restricted as required. However, they may only be extended if they were previously assigned one of the privileges TSOS or PRINT-SERVICE-ADMINISTRATION.
- **Changing the default values**
As long as syntax rules are observed, the default values for DISPLAY commands can be deleted (i.e. entry of the operand concerned becomes mandatory), modified or (re)introduced (i.e. entry of the operand which was previously mandatory becomes optional).
- **Restricting the value range:**
The permitted value range of an operand can be restricted by deleting or changing the default options (e.g. wildcards not permitted in the file name).

In particular, general data types can be replaced by a list of values which are actually allowed (e.g. data type <name 1..8> for a user ID by the user IDs X, Y and Z for which the command is to be permitted).

14.1.1 Restrictions

The following restrictions apply for using the DISPLAY subsystem:

- In accordance with the F2/Q3 specifications, the DISPLAY subsystem may not be used in a security system.
- The product SDF-A is required to modify DISPLAY commands.
- The CREATE-DISPLAY-CMD command cannot be applied to commands which were defined with CMD-ALLOWED = NO.
- For security reasons the CREATE-DISPLAY-CMD command may not be used with:
 - the SHOW-USER-ATTRIBUTES command
 - commands which were defined with CMD-ALLOWED=YES (UNLOAD=YES)
- When other commands which are relevant to security, such as SHOW-FILE (the file can contain passwords) or SHOW-JV-ATTRIBUTES (output of passwords with INF=*ALL), are released, it is emphatically recommended that access should be restricted by modifications (see [“Modifying DISPLAY commands” on page 302](#)).

14.2 Commands

14.2.1 Overview of all commands

The DISPLAY subsystem accepts the commands listed in the table below.

Statement	Brief description
CREATE-DISPLAY-CMD	Create a DISPLAY command
DELETE-DISPLAY-CMD	Delete a DISPLAY command

Table 19: Commands accepted by DISPLAY

14.2.2 CREATE-DISPLAY-CMD

The CREATE-DISPLAY-CMD command creates a DISPLAY command. This enables the functionality of the corresponding SHOW command to be made available to nonprivileged users.

CREATE-DISPLAY-CMD	Alias: CRDPCMD
<p>CMD = <structured-name 1..30></p> <p>,FROM-SYNTAX-FILE = *TASK-HIERARCHY / <filename 1..54></p> <p>,INTERNAL-NAME = *AUTOMATIC / <alphanum-name 1..8></p> <p>,PRIVILEGE = *ALL / *EXCEPT(...) / list-poss (64): <structured-name 1..30></p> <p style="padding-left: 2em;">*EXCEPT(...)</p> <p style="padding-left: 4em;"> EXCEPT-PRIVILEGE = list-poss (64): <structured-name 1..30></p> <p>,TO-SYNTAX-FILE = *SYSSDF / <filename_1..54></p> <p>,WRITE-MODE = *NEW / *EXCHANGE</p>	

CMD = <structured-name 1..30>

Full, NOT ABBREVIATED, name of the SHOW command for which a corresponding DISPLAY command is to be created.

FROM-SYNTAX-FILE = ***TASK-HIERARCHY** / <filename 1..54>

Syntax file in which the SHOW command is defined.

INTERNAL-NAME = ***AUTOMATIC** / <alphanum-name 1..8>

Internal name of the DISPLAY command. By default one is created automatically. In case of ambiguity the name must be explicitly specified.

PRIVILEGE = ***ALL** / ***EXCEPT(...)** / list-poss (64): <structured-name 1..30>

Privileges of the DISPLAY command. The syntax and default correspond to those of the PRIVILEGE operand of the SDF-A ADD-CMD statement.

PRIVILEGE = *EXCEPT(...)**EXCEPT-PRIVILEGE = list-poss (64): <structured-name1..30>**

Exceptions list of privileges for the DISPLAY command. Execution of the DISPLAY command is permitted with every privilege except those specified here. This also applies for privileges which will only be introduced in the future.

TO-SYNTAX-FILE = *SYSSDF / <filename_1..54>

Syntax file in which the DISPLAY command is to be entered. This syntax file must exist and must contain the DISPLAY domain. The default is the installed syntax file of the DISPLAY subsystem.

WRITE-MODE = *NEW / *EXCHANGE

Defines whether an already existing entry in the syntax file is to be overwritten:

WRITE-MODE = *NEW

No entry for the command which is to be created may exist yet. If one does, command processing is aborted with an error.

WRITE-MODE = *EXCHANGE

Any existing entry for the command which is to be created is replaced. If no entry exists yet, command processing is aborted with an error.

14.2.3 DELETE-DISPLAY-CMD

The DELETE-DISPLAY-CMD command deletes a DISPLAY command.

DELETE-DISPLAY-CMD	Alias: DLDP CMD
CMD = <structured-name_1..30>	
,FROM-SYNTAX-FILE = *SYSSDF / <filename_1..54>	

CMD = <**structured-name_1..30**>

Full, not abbreviated, name of the DISPLAY command to delete.

FROM-SYNTAX-FILE = ***SYSSDF** / <**filename_1..54**>

Syntax file in which the DISPLAY command to delete is defined.

14.3 Installation

Detailed information on installing the DISPLAY subsystem and on hardware and software requirements is provided in the Release Notice for the software product DISPLAY.

The files for DISPLAY V1.1A are supplied using the SOLIS delivery procedure.

The following files are supplied:

File	Comment
SKMLNK.DISPLAY.011	Subsystem module library for operation on SX servers
SPMLNK.DISPLAY.011	Subsystem module library for operation on SQ servers
SYSFGM.DISPLAY.011.D	Release Notice (German)
SYSFGM.DISPLAY.011.E	Release Notice (English)
SYSLNK.DISPLAY.011	Subsystem module library for operation on S servers
SYSTEMS.DISPLAY.011	Subsystem message file
SYSPRC.DISPLAY.011	Procedure for creating entries in the syntax file
SYSRMS.DISPLAY.011	Delivery scope for DISPLAY
SYSSDF.DISPLAY.011	Subsystem syntax file
SYSSII.DISPLAY.011	Structure and installation information for IMON
SYSSSC.DISPLAY.011	DSSM declarations

The valid file and volume attributes are listed in the SOLIS2 delivery note.

Related publications

The manuals are available as online manuals, see <http://manuals.ts.fujitsu.com>, or in printed form which must be paid and ordered separately at <http://manualshop.ts.fujitsu.com>.

- [1] **BS2000/OSD-BC
Commands**
User Guide
- [2] **BS2000/OSD-BC
Introductory Guide to Systems Support**
User Guide
- [3] **BS2000/OSD-BC
System Installation**
User Guide
- [4] **SDF-A (BS2000/OSD)**
User Guide
- [5] **SDF-P (BS2000/OSD)**
Programming in the Command Language
User Guide
- [6] **POSIX (BS2000/OSD)**
POSIX Basics for Users and System Administrators
User Guide
- [7] **POSIX (BS2000/OSD)**
Commands
User Guide
- [8] **FDDRL (BS2000/OSD)**
User Guide
- [9] **JV (BS2000/OSD)**
Job Variables
User Guide

Related publications

- [10] **SECOS** (BS2000/OSD)
Security Control System - Access Control
User Guide
- [11] **SECOS** (BS2000/OSD)
Security Control System - Audit
User Guide
- [12] **XHCS** (BS2000/OSD)
8-Bit Code Processing in BS2000/OSD
User Guide
- [13] **BS2000/OSD**
Softbooks English
CD-ROM

Index

A

- abbreviated input 77
- abbreviation
 - name 77
- adding user-generated syntax files 176
- application domain 52
- assignment
 - group syntax file 139
 - input procedure 233
 - of output procedure 246
 - output procedure 232, 235
 - system syntax file 138
 - user syntax file 140
- assignment ISP - SDF
 - system administration commands 287
 - user commands 282

B

- basic system syntax file 163
- batch mode, input 69
- behavior in the event of errors 163
- blanking 35
- blocked input 37

C

- cancel function 37, 49
- card reader 250
- change password 20
- CHECKPOINT/RESTART 157
- CLIGET 67
- CLISSET 67
- command
 - application domain SDF 145
 - CREATE-DISPLAY-CMD 305
 - DELETE-DISPLAY-CMD 307

- disable 12
- format 14
- input example 99
- ISP format 12
- modify 12, 143
- reconstruct 18
- representation of syntax 14
- restrict 12
- syntax analysis 89
- test mode 89
- command (general)
 - rename 78
- command language
 - format 14
 - ISP 12
- command menu 53
- command procedure, user implementation 62
- command processor SDF 11
- commands
 - to be excluded from conversion 232, 238
- commands for system administration
 - assignment ISP - SDF 287
 - conversion ISP - SDF by SDF-CONV 296
- comment lines
 - output procedure 232, 238
- components of SDF-CONV 263
- compressed input 77
 - structures 81
- context dependency
 - output procedure 232
- continuation character
 - input procedure 235
 - New mode 62, 69
 - Old mode 62, 69
 - output procedure 232, 237

- continuation line 62, 69
 - New mode 62, 69
 - Old mode 62, 69
- control of dialog guidance 245
- control statements, menu 42, 45
- conversion ISP - SDF by SDF-CONV
 - system administration commands 296
 - user commands 291
- conversion log
 - output to SYSOUT 248
- conversion of command language (ISP to SDF) 229
 - examples 265
- conversion of procedure format (non-S to S) 229
 - examples 265
- conversion proposal
 - output procedure 239, 246
- conversion restrictions 250
 - non-convertible commands and operands 254, 255
 - procedure parameters as operands 260
- conversion result
 - output procedure 246
 - SYSOUT 248
- CONVERT statement 232
- CONVERT-SYNTAX-FILE statement 172
- COPY statement (SDF-U) 191
- correction dialog 11, 34
- CREATE-DISPLAY-CMD (command) 305
- current object 194

D

- data lines in input procedure 243
- default value 79
 - task-specific 93
- defining
 - checkpoint 215
 - restart point 215
- definition
 - modify for operand value 198
- DELETE-DISPLAY-CMD (command) 307
- deleting objects from syntax files 200
- delivery unit syntax file 144

- dialog guidance 99
 - control statements 42, 45
 - function keys 47
 - ISP commands 58
 - language selection 12
 - selection 34
- dialog guidance control 245
- dialog levels 39
- documentation
 - output procedure 232
- documentation level
 - output procedure 238, 246
- domain 52
- DSSM 161

E

- EDIT statement (SDF-U) 194
- END standard statement 146
- END statement 173
 - SDF-PAR 218
 - SDF-U 196
- entries
 - special 43
- error behavior 163
- error correction dialog 18
- error messages
 - SDF-CONV 298
- example
 - change password 20
 - LOGON procedure 24
 - output application domains 19
 - query task-specific SDF options 22
 - reconstruct a command 28
 - terminate an interactive task 29
- examples 265
- execute function 47, 48
- EXECUTE-SYSTEM-CMD
 - standard statement 147
- executing SDF-I 166
- exit function 36, 47, 48
- exit-all function 36, 47, 48
- EXPERT form dialog guidance 33
- expert mode 33

F

format

SDF command 14

SDF statement 14

function key assignment 47

function keys 36

G

generating the subsystem catalog 161

group syntax file 139

grouped input 37

guaranteed abbreviation 78

guided dialog 39

levels 39

operand form 54

screen format 41

special entries 56

temporary 58

H

handling of optional operands 240

help function 36, 48

HELP-MSG-INFORMATION

standard statement 148

hierarchy of syntax files 141

history 87

HOLD-PROGRAM

standard statement 148

I

implementing command procedures 62

information system 18

inline comments 246

input

abbreviate 17

as keyword operand 80

as positional operand 80

blanked 11

block 37

compression 77

continuation line 69

explicit 79

from a procedure 11, 62

from job variables 70

function keys 47

guided dialog 56

ignore 89

implicit 79

in batch mode 11, 69

in interactive mode 11

in line mode 17

in menu mode 17

ISP command 58

NEXT line 42, 45

operand form 54

restore 11

reuse 87

save 87

sensitive 87

unguided dialog 35

via function keys 47

input buffer 18, 87

input procedure

assignment to file or library element 233

inline comments 246

input procedure (procedure to be converted)

handling of data lines 243

procedure format 241

input serial number 87

INSF 165

installation

SDF 159

SDF-PAR 217

installation files 159

installation of SDF-CONV 264

interactive mode input 11

interactive task

start 29

terminate 29

interrupt

accept 67

reject 67

interrupt event 66

interrupt function 36, 47, 48

interruptibility

of procedures 65

of programs 66

ISP (Interactive Scanner and Processor) 12

- ISP command language 12
- ISP commands
 - assignment to SDF commands 282
 - conversion restrictions 250
 - conversion to SDF commands 291
 - dialog guidance 58
 - non-convertible 254, 255
- J**
- job variable replacement
 - conversion setting 235
- job variables
 - substitute 71
- K**
- K2 interrupt, procedures 65
- keyword
 - abbreviation 77
- L**
- language selection dialog guidance 12
- loading SDF 161, 163
- log 83
- logging
 - scope 83
- LOGOFF procedure 64
 - standard name 64
 - system-wide 64
 - user-specific 64
- LOGON procedure 63
 - standard name 63
 - system-wide 63
 - user-specific 63
- logon procedure 63
 - create 24
- M**
- macro calls, SDF 154
- masked input 35
 - log 83
- menu 42
 - application domain 52
 - command/statement 53
 - conditions for display 53
 - logging 83
- MERGE statement 174
- message file 298
 - installation 264
- MODIFY-CMD statement 197
- modifying
 - command definition 196
 - global information 202
 - operand value definition 198
- MODIFY-SDF-OPTIONS
 - standard statement 149
- MODIFY-SYNTAX-FILE (SDF-PAR) 220
- MODIFY-SYSTEM-LOGON-PROCEDURE (SDF-PAR) 222, 223
- MODIFY-VALUE statement (SDF-U) 198
- monitoring job variables 249
- N**
- New mode, continuation character 62, 69
- NEXT line 42, 45
- NO form, dialog guidance 34
- non-S procedures 229
- O**
- object, current 194
- Old mode
 - continuation character 62, 69
 - function key assignment 36, 47
- OMNIS 157
- OPEN statement 178
- OPEN-PARAMETER-FILE statement 225
- OPEN-SYNTAX-FILE statement 199
- operand
 - default value 79
 - optional 79
 - positional 80
- operand form 42, 54
 - conditions for display 54
 - switch to a different level 55

- operand value
 - explicit entry 35
 - implicit entry 35
 - operand value definition
 - modify 198
 - operating system dependency 239
 - operating system version dependency 232
 - optional operands in conversion 240
 - output
 - into PLAM library (output procedure) 246
 - into SAM file (output procedure) 246
 - monitoring job variables for program monitoring 249
 - of messages during data line conversion 243
 - syntax file name 214
 - to SYSOUT (overview) 248
 - output application domains 19
 - output procedure
 - assignment to file or library element 232, 235
 - comment lines 232
 - context dependency 232
 - conversion proposal 239, 246
 - documentation 232
 - documentation level 238, 246
 - version dependency 232
 - output procedure (converted procedure)
 - assignment to file or library element 246
 - contents 246
 - definition of command format 242
 - definition of procedure format 241, 242
 - output format 241, 243
- P**
- parameter file 138
 - positional operand 80
 - preset
 - operand value 79
 - procedure
 - correction dialog 62
 - for LOGOFF 64
 - for LOGON 63
 - interruption 65
 - logon 63
 - procedure dialog 65
 - procedure file 62
 - as input procedure 233
 - as output procedure 232, 235
 - procedure files
 - system-wide 63
 - procedure parameter
 - substitute 70
 - procedure parameters
 - conversion setting 235
 - restrictions 252
 - procedure parameters as operands 260
 - profile ID 139
 - program
 - interruption 66
 - program monitoring
 - monitoring job variables 249
 - Prozedurdateien
 - systemweite 63
- Q**
- quit functions 50, 60
- R**
- Readme file 15
 - reconstruct command 28
 - refresh function 47, 48
 - REMARK
 - standard statement 149
 - REMOVE statement 181, 200
 - REP file
 - installation 264
 - optional REPs 264
 - RESET-INPUT-DEFAULTS
 - standard statement 150
 - restart point
 - define 215
 - restore function 37, 48
 - restore input 11
 - RESTORE-SDF-INPUT
 - standard statement 150
 - restrictions
 - conversion 250
 - restrictions on conversion into S procedures 252

S

- S procedures 229
 - restrictions on conversion 252
- save input 87
- screen format 41
- scroll backward function 48
- scroll forward function 48
- SDF
 - application domain 145
 - CHECKPOINT/RESTART 157
 - command processor 11
 - correction dialog 11
 - dialog guidance control 245
 - format of the command language 14
 - installation 159
 - language of dialog guidance 12
 - load 161, 163
 - macros 154
 - OMNIS 157
 - parameter file 138, 139
 - profile ID 139
 - representation of syntax 14
 - standard statements 146
 - syntax description 137
 - syntax file 137
- SDF (System Dialog Facility) 11
- SDF commands
 - assignment to ISP commands 282
 - conversion from ISP commands 291
- SDF parameter file 217
 - create 217, 225
 - display 228
 - modify 217
 - modify entries 220, 222, 223
 - open 225
- SDF standard statements 169
 - overview 231
- SDF-A 137
 - command set modification 12
 - statement set modification 12
- SDF-CONV
 - installation 264
 - operation 229
 - overview of possible conversions 230
- SDF-CONV messages 298
- SDF-I
 - add software unit system files 174
 - assign input syntax file 178
 - assign output syntax file 178
 - convert syntax file 172
 - execution 166
 - merge syntax files 174
 - notes on input 169
 - start 166
 - syntax file compatibility 169, 170
 - terminate 173
 - unmerge software unit syntax files 181
 - unmerge syntax files 181
 - unmerge user-generated syntax files 181
 - work files 168
- SDF-I utility routine 137
- SDF-PAR 217
 - install 217
 - start 217
 - terminate 218
 - user syntax file 217
- SDF-U 187
 - interrupt 188
 - modify command definitions 197
 - resume 188
 - start 187
- SDF-U utility routine 137
- selection menu 85
 - ambiguous abbreviation 86
 - wildcards 85
- SET-GLOBALS statement (SDF-U) 202
- SHOW statement (SDF-U) 208
- SHOW-CORRECTION-INFORMATION statement 213
- SHOW-INPUT-DEFAULTS
 - standard statement 151
- SHOW-INPUT-HISTORY
 - standard statement 151
- SHOW-PARAMETER-FILE statement 228
- SHOW-SDF-OPTIONS
 - standard statement 152
- SHOW-STATUS statement 214

- SHOW-STMT
 - (standard statement) 152
- SHOW-SYNTAX-FILE statement (SDF-I) 183
- software unit syntax files
 - merge 174
 - unmerge 181
- special entries
 - guided dialog 56
 - unguided dialog 35, 43
- SSCM 161
- standard name
 - LOGOFF procedure 64
 - LOGON procedure 63
- standard names 78
- standard statement 146
 - END 146
 - EXECUTE-SYSTEM-CMD 147
 - HELP-MSG-INFORMATION 148
 - HOLD-PROGRAM 148
 - MODIFY-SDF-OPTIONS 149
 - REMARK 149
 - RESET-INPUT-DEFAULTS 150
 - RESTORE-SDF-INPUT 150
 - SHOW-INPUT-DEFAULTS 151
 - SHOW-INPUT-HISTORY 151
 - SHOW-SDF-OPTIONS 152
 - SHOW-STMT 152
 - STEP 153
 - WRITE-TEXT 153
- standard statements 169, 188
- starting
 - SDF 161
 - SDF-I 166
 - SDF-PAR 217
 - SDF-U 187
- starting SDF-CONV 230
- statement
 - CONVERT 232
 - format 14
 - representation of syntax 14
 - syntax analysis 89
 - test mode 89
- statement menu 53
- statements
 - overview 231
- STEP standard statement 153
- STEP statement 215
 - SDF-U 215
- structure
 - attribute NULL-ABBREVIATION 82
 - flat notation 81
 - input compression 81
 - STRUCTURE-IMPLICIT notation 81
- Styleguide mode
 - function key assignment 36, 47
- subdivision of the dialog screen 41
- substituting
 - job variables 71
 - procedure parameters 70
- substituting job variables 70
- subsystem catalog
 - generate 161
- subsystem SDF-CONV 263
- SUSF 165
- syntax analysis 89, 221, 226, 227
- syntax file 137
 - create 143
 - delivery unit 144
 - for user group 139
 - group 139
 - hierarchy 141
 - merge 144
 - modify 143
 - system 138
 - types 137
 - unmerge 144
 - user 140
- syntax files
 - add user-generated 176
 - assignment 232, 237
 - conversion 172
 - copy contents 191
 - delete objects 200
 - installation 264
 - merge 174
 - modify 187
 - open 199

- output name 214
- output objects 208
- position to a command 194
- unmerge 181

syntax representation 14

SYSOUT output 248

system administration commands

- conversion restrictions 250

system syntax file 138

system syntax files

- activate 221

- deactivate 221, 226, 227

- information on 183

systemweite LOGON-Prozeduren 63

system-wide logon procedure 63

system-wide LOGON procedures 63

T

task-specific default value 93

task-specific SDF options 22

temporarily guided dialog 34, 58

- termination 59

terminating

- program run 194

- SDF-PAR 218

terminating SDF-CONV 230

test function 47

test mode

- activate 89

- for commands 89

- for statements 89

U

unguided dialog 33

- correction 34

- special entries 35, 43

user commands

- assignment ISP - SDF 282

- conversion ISP - SDF by SDF-CONV 291

- conversion restrictions 250

user guidance

- maximum 39

- medium 40

- minimum 40

user syntax file 140

user syntax files

- for SDF-PAR 217

user-generated syntax files

- unmerge 181

user-specific logon procedure 63

utility routine

- SDF-I 137

- SDF-PAR 217

- SDF-U 137

V

version dependency

- operating system 232

W

wildcards

- in statement names 85

WRITE-TEXT

- standard statement 153