1 Preface

The software product SDF (**S**ystem **D**ialog **F**acility) implements the BS2000 user interface. It supports the input of commands and program statements in dialogs, from procedures and in batch mode. SDF also has its own command language (commands and statements in the SDF format). SDF requires syntax descriptions to process input internally. These descriptions are stored in separate files called syntax files.

SDF V4.5 can be run on any BS2000 version as of BS2000/OSD-BC V3.0.

This “SDF Management” manual describes how SDF can be installed and administered by means of SDF commands and the utilities SDF-I, SDF-U and SDF-PAR.

The following topics are dealt with in this manual:

– syntax files used by SDF
– commands to control SDF
– installation and management of syntax files
– merging of syntax files using SDF-I
– modification of syntax files using SDF-U
– creation and processing of the SDF parameter file with SDF-PAR

1.1 Target group

This manual is intended for systems support staff and experienced BS2000 users. It assumes that the user is familiar with the operating system and the SDF dialog interface.

The following manuals are recommended as basic reference works for the installation of software products: “Introductory Guide to Systems Support” [2], “IMON” [6] and “Subsystem Management” [16].

1.2 Summary of contents

In the introductory sections on SDF, SDF-I and SDF-U, you will learn how SDF is installed and how syntax files can be merged and modified, and you will be presented with a brief introduction to the SDF dialog interface.

The commands for controlling SDF and the statements of the SDF-I, SDF-U and SDF-PAR utilities are described in separate chapters in alphabetical order.

General information on the SDF dialog interface can be found in the manual “Introductory Guide to the SDF Dialog Interface” [1].

The BS2000 commands, including privileged commands, are described in the “Commands” manual [3], volumes 1 to 5.

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

README file

Information on any functional changes and additions to the current product version described in this manual can be found in the product-specific README file.
You will find the README file on your BS2000 computer under the file name SYSRME.*product*.*version*.*language* and for SDF V4.5 under SYSRME.SDF.045.E.

The user ID under which the README file is cataloged can be obtained from your systems support staff. The full path name is also output using the following command:

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

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

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

1.3 Changes since the last version of the manual

The SDF V4.5A manual contains the following changes since the last version of the manual (SDF Management for SDF V4.1A published in June, 1996):

- LOGOFF procedures that start automatically can be defined to terminate tasks. The definition can be applied to a specific user or to the entire system. One call and one include procedure can be defined in the SDF parameter file for a system-wide LOGOFF procedure. A user-specific call and/or include procedure is provided via the standard file name, just like for the LOGON procedure.
- The installation files for use under BS2000/OSD-BC V3.0 and higher are described in the chapter “Installation of SDF”.
- Changes have been made to the following commands:

Command	Changes
MODIFY-SDF-PARAMETERS	New SYSTEM-LOGOFF-INCL operand New SYSTEM-LOGOFF-PROC operand
SHOW-SDF-PARAMETERS	New SYSTEM-LOGOFF-INCL operand New SYSTEM-LOGOFF-PROC operand Extensions to the S variables

- The standard statement SHOW-STMT displays the syntax of a statement (functions in the same manner as the SHOW-CMD command).
- The examples were created on a system running BS2000/OSD V5.0.

1.4 Notes on the user interface

Detailed information on various options provided by SDF can be found in the “Introductory Guide to the SDF Dialog Interface” [1].

Abbreviation of names

SDF permits inputs to be abbreviated in interactive and batch modes as well as in procedures, provided the abbreviations used are unambiguous within the related syntax environment. Note, however, that an abbreviation that is currently unambiguous could potentially become ambiguous at a later date, particularly if new functions are added to the product. For this reason, it is best to avoid abbreviations entirely, especially in procedures, or at the very least, to ensure that only guaranteed abbreviations be used. In the statement formats, these guaranteed abbreviations are shown in boldface.

Command and statement names, operands and keyword values may be abbreviated as follows:

– Complete name components may be omitted from right to left; the hyphen preceding the dropped name component is also omitted.

– Individual characters of a name component may be omitted from right to left.

– An asterisk (*) preceding a keyword value is not considered a valid abbreviation for that value. As of SDF V4.0A, keyword values are always represented with a leading asterisk. The asterisk may be omitted only if there is no possible alternative variable operand value with a value range that includes the name of the keyword value. This form of abbreviation may be restricted due to extensions in later versions. For compatibility reasons, operand values that were previously represented without an asterisk are still accepted without the asterisk.

Example of input

Unabbreviated command format:

```
/MODIFY-SDF-OPTIONS SYNTAX-FILE=*NONE,GUIDANCE=*MINIMUM
```

Abbreviated command format:

```
/MOD-SDF-OPT SYN-F=*NO,GUID=*MIN
```

The guaranteed abbreviations are only intended as recommendations for abbreviated input; they may not always be the shortest possible input in your syntax environment. They are, however, clear and easy to understand and are designed to remain unique in the long term.

In some cases, an additional abbreviation is documented in the manual next to the command or statement name. This abbreviation is implemented as an alias for the command or statement name and is guaranteed in the long term. The alias consists of a maximum of 8 letters (A...Z) that are derived from the command or statement name. Aliases cannot be abbreviated further.

Default values

Most operands are optional, i.e. need not be explicitly specified. Such operands are preset to a specific operand value, the so-called default value. The default value for each operand is shown in the syntax underscored. If an optional operand is not explicitly specified, its default value is automatically inserted when executing the command or statement.

Positional operands

SDF permits operands to be specified either as keyword operands or as positional operands. However, it is quite possible that the positions of operands may change in future versions of the product. It is therefore advisable to avoid the use of positional operands, especially in procedures.

XHCS support

Every terminal works with a particular character set. A coded character set (CCS) is the unique representation of the characters in a character set in binary form. Extended character sets can be used when working with the software product XHCS[1]. SDF interprets inputs in accordance with the standard code table **EBCDIC.DF.03** (e.g. when converting uppercase/ lowercase letters).
The coding of the following characters in an extended character set must be the same as the coding used in the standard code table:

$, # , @ , ! , ” , ? , ^ , = , : , / , * , - , (,) , [,] , < , > , comma, semicolon, single quotes

SDF does not interpret any additional characters from an extended character set unless they appear within the data types <c-string> and <text>. In other words, the conversion of uppercase/lowercase letters is handled using a code table supplied by XHCS for the extended character set. Any additional characters used within other data types are rejected as syntax errors.

When statements are entered, the CCS used is the one specified in the appropriate SDF macro (RDSTMT/CMDRST, TRSTMT/CMDTST or CORSTMT/CMDTST). If no CCS was specified, the one used for entering commands is assumed.

For further details on XHCS support, see the manuals “Introductory Guide to the SDF Dialog Interface” [1] and “XHCS” [15].

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

1.5 Metasyntax

1.5.1 Notational conventions

The following notational conventions are used in this manual:

for notes

for warnings

Runtime examples are shown in `typewriter script`. Inputs expected from the user are highlighted in **`bold`**.

In the text, references to other publications are given as abbreviated titles with numbers enclosed in square brackets [ ]. All references that follow refer to BS2000/OSD-BC V2.0. The complete title of each publication to which reference is made is listed under “Related publications” at the back of the manual. This is followed by information on how to order manuals.

1.5.2 SDF syntax description

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

Table 1: Notational conventions

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

Table 2: Data types

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

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

Table 3: Suffixes for data types

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

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

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

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

Metasyntax

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

Table 1: Metasyntax (part 1 of 2)

Representation	Meaning	Examples
|	A vertical bar identifies related operands within a structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical bars preceding an operand corresponds to the depth of the structure.	**SUP**PORT = ***TAPE**(...) ***TAPE(...)** | **VOL**UME = ***ANY**(...) | ***ANY**(...) | | ...
,	A comma precedes further operands at the same structure level.	**GUID**ANCE-**MODE** = ***NO** / ***Y**ES ,**SDF-COM**MANDS = ***NO** / ***Y**ES
list-poss(n):	The entry “list-poss” signifies that a list of operand values can be given at this point. If (n) is present, it means that the list must not have more than n elements. A list of more than one element must be enclosed in parentheses.	list-poss: ***SAM** / ***ISAM** list-poss(40): <structured-name 1..30> list-poss(256): ***OMF** / ***SYSLST**(...) / <filename 1..54>
Alias:	The name that follows represents a guaranteed alias (abbreviation) for the command or statement name.	**HELP-SDF** Alias: **HPSDF**

Table 1: Metasyntax (part 2 of 2)

Data types

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

Table 2: Data types (part 1 of 6)

Data type	Character set	Special rules
filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.] { file | file(no) | group | group { (*abs) | (+rel) | (-rel) } } :cat: optional entry of the catalog identifier; character set limited to A...Z and 0...9; maximum of 4 characters; must be enclosed in colons; default value is the catalog identifier assigned to the user ID, as specified in the user catalog. $user. optional entry of the user ID; character set is A…Z, 0…9, $, #, @; maximum of 8 characters; first character cannot be a digit; $ and period are mandatory; default value is the user's own ID. $. (special case) system default ID file file or job variable name; may be split into a number of partial names using a period as a delimiter: $name_1$[.$name_2$[...]] $name_i$ does not contain a period and must not begin or end with a hyphen; file can have a maximum length of 41 characters; it must not begin with a $ and must include at least one character from the range A...Z.

Table 2: Data types (part 2 of 6)

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

Table 2: Data types (part 3 of 6)

Data type	Character set	Special rules
partial-filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.][partname.] :cat: see filename $user. see filename partname optional entry of the initial part of a name common to a number of files or file generation groups in the form: $name_1$.[$name_2$.[...]] $name_i$ (see filename). The final character of “partname” must be a period. At least one of the parts :cat:, $user. or partname must be specified.
posix-filename	A...Z 0...9 special characters	String with a length of up to 255 characters; consists of either one or two periods or of alpha-numeric characters and special characters. The special characters must be escaped with a preceding \ (backslash); the / is not allowed. Must be enclosed within single quotes if alter-native data types are permitted, separators are used, or the first character is a ?, ! or ^. A distinction is made between uppercase and lowercase.
posix-pathname	A...Z 0...9 special characters structure identifier: slash	Input format: [/]$part_1$/.../$part_n$ where $part_i$ is a posix-filename; max. 1023 characters; must be enclosed within single quotes if alter-native data types are permitted, separators are used, or the first character is a ?, ! or ^.

Table 2: Data types (part 4 of 6)

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

Table 2: Data types (part 5 of 6)

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

Table 2: Data types (part 6 of 6)

Suffixes for data types

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

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

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

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

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

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

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with (contd.)

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

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

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

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

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with-wild-
constr(n)
(contd.)</td><td>Examples:
<table>
<tr><th>Selector</th><th>Selection</th><th>Constructor</th><th>New name</th></tr>
<tr><td>A//*</td><td>AB1
AB2
A.B.C</td><td>D<3><2></td><td>D1
D2
D.CB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<3>.XY<2></td><td>G.A.D.XYA
G.A.D.XYB
G.B.F.XYA
G.B.F.XYB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<2>.XY<2></td><td>G.A.A.XYA
G.A.B.XYB
G.B.A.XYA
G.B.B.XYB</td></tr>
<tr><td>A//B</td><td>ACDB
ACEB
AC.B
A.CB</td><td>G/XY/</td><td>GCXYD
GCXYE
GCXY. [1]
G.XYC</td></tr>
</table>
[1] The period at the end of the name may violate naming conventions (e.g. for fully-qualified file names).</td></tr>
<tr><td>without</td><td>Restricts the specification options for a data type.</td></tr>
<tr><td>-cat</td><td>Specification of a catalog ID is not permitted.</td></tr>
<tr><td>-corr</td><td>Input format: [[C]’][V][m]m.na[’]
Specifications for the data type product-version must not include the correction status.</td></tr>
<tr><td>-gen</td><td>Specification of a file generation or file generation group is not permitted.</td></tr>
<tr><td>-man</td><td>Input format: [[C]’][V][m]m.n[’]
Specifications for the data type product-version must not include either release or correction status.</td></tr>
<tr><td>-odd</td><td>The data type x-text permits only an even number of characters.</td></tr>
<tr><td>-sep</td><td>With the data type “text”, specification of the following separators is not permitted: ; = ( ) < > ␣ (i.e. semicolon, equals sign, left and right parentheses, greater than, less than, and blank).</td></tr>
<tr><td>-temp-
file</td><td>Specification of a temporary file is not permitted (see #file or @file under filename).</td></tr>
</table>

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

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

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

1.5.3 Metasyntax for describing output in S variables

Starting with SDF V4.0 and SDF-P V2.0, the output of the SHOW command can be redirected to S variables. Output of this type is described by using the tabular format shown below:

Output information	**Name of the S variable**	**Type**	**Contents**
(Brief description of the output information)	(e.g. var.MENU-LOG, where var must be a declared S variable)	Type of S variable: S: STRING I: INTEGER	(Possible values that can be assigned to the S variable)

The name of the S variable may include the suffix (*LIST) to indicate that a list variable is involved. The (*LIST) suffix appears when the contents of the S variable are output by the SHOW-VAR command. It must not be specified by the user for assignments.

Special features and notes related to the specific information that is output are provided at the end of the table.

The “Commands, Vol. 6” manual [4] contains information on working with S variable output. The S variables of the SHOW command from “Commands, Vol. 4 and 5” [3] are also described there.

2 How SDF works

The command processor SDF (**S**ystem **D**ialog **F**acility) supports the input of BS2000 commands and program statements in SDF format. SDF format is the default command language as of BS2000 V10.0. SDF cannot be deactivated.

Syntax files

SDF requires the following data, which is made available in syntax files, for command and statement processing:

- information on the syntax of commands
- information on how these commands are implemented in BS2000, e.g.
  - the names of the system entries or procedures via which command execution is initiated
  - the definitions for parameter transfer to the executing system modules
- Information on privileges that the calling user must have
- information on the programs reading their statements via SDF, and the syntax of these statements
- general and command-related definitions for dialog guidance
- explanatory texts for the commands and statements and their operands (help texts).

Whenever SDF processes a command or statement input, it fetches the necessary information from the activated syntax files on the basis of a defined hierarchy. Syntax files are generated using the software product SDF-A (see the “SDF-A” manual [7]).

Syntax file types

There are three types of syntax files:

- System syntax files contain the syntax of commands and statements available to all users. As of SDF V3.0, multiple system syntax files may be active simultaneously. One of the active files must be the basic system syntax file; the basic file may be accompanied by a number of subsystem syntax files.
- Group syntax files are assigned to user IDs. They may restrict or (in particular for the privileged user ID TSOS) extend the scope of the system syntax files.
- User syntax files can be created and assigned by users themselves. They may restrict the scope of the system syntax files and the group syntax file (if any). Extensions and other functional changes are limited to statements issued to user programs and to commands implemented by means of procedures. As of SDF V4.1, a number of user syntax files may be activated simultaneously per user job.

The following distinction is also made for system and group syntax files:

- Software unit syntax files (SUSFs) contain the syntax of a specific product, such as SDF itself or BS2CPfor the basic configuration of BS2000. SUSFs are supplied by Fujitsu Siemens Computers.
- Installation syntax files (INSFs) consist of SUSFs that are merged with the user's own syntax files and can be used by SDF as syntax files during an ongoing session. An INSF that is individually customized to a computer center can be generated by means of the SDF-I utility before SDF is first installed.

The syntax of the commands and programs belonging to BS2000 is supplied by Fujitsu Siemens Computers in SUSFs designed for the respective products.

Security aspects

Privileges restricting access to specific objects can be assigned to objects in all three types of syntax files. In order to access an object, a user ID must then have at least one of the privileges assigned to that object. A user ID may also have more than one privilege. By evaluating the privileges, all objects to which all user IDs have access can be defined in one syntax file.

When the software package SECOS is being used, specific syntax file events, such as the activation of syntax files, can be logged (see the "SECOS" manual [13]).

Utility routines for managing and processing syntax files

The system and group syntax files created by the user with SDF-A (see [7]) or supplied with the software product can be merged into a system or group syntax file with the utility routine **SDF-I** . Syntax files from products not needed anymore can also be unmerged. Syntax files with different file formats can also be converted with SDF-I , and information on file formats and contents can also be output.

The utility routine **SDF-U** provides capabilities for modifying syntax files. The routine uses a subset of the functions and statements of the SDF-A software product to do this.
The following modifications were made:

- add new or modified command/program definitions to a syntax file (COPY statement)
- change default values (predefined settings) for the SDF options in a syntax file (SET-GLOBALS statement)
- output the contents of a syntax file to SYSLST or SYSOUT (SHOW statement)
- replace the procedure by which a command is implemented (MODIFY-CMD statement)

More comprehensive ways of manipulating and processing syntax files and creating user-specific syntax files are offered by the software product SDF-A, which is described in the “SDF-A” manual [7].

The **SDF-PAR** utility can be used to create, modify and display an SDF parameter file.

System and group syntax files generated using SDF-A, SDF-I or SDF-U can be activated and deactivated by means of the MODIFY-SDF-PARAMETERS command or the SDF-PAR utility. User syntax files are activated and deactivated by means of the MODIFY-SDF-OPTIONS command.

Command return codes

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

Command return codes consist of three parts:

- the main code, a message code that can be specified with the HELP-MSG-INFORMATION command in order to obtain detailed information.
- subcode1, which assigns the error situation to an error class that indicates how serious an error is. Subcode1 has a decimal value.
- subcode2, which can contain additional information (value is not zero). Subcode2 has a decimal value. In the event of an error (subcode1 is not zero), there are no rules regarding the use of subcode2. The value of subcode2 may be zero, 1 or 2 if no error has occurred. Subcode2 with the value 1 indicates that the requested function was already available before the command was issued. Subcode2 with the value 2 indicates an exception situation and should be classified as a warning.

A detailed description of all command return codes can be found in the manual “Introductory Guide to the SDF Dialog Interface” [1] and in “Commands Vol. 1” [3]. The command return code can only be evaluated using SDF-P resources in S procedures and dialog blocks (see the “SDF-P” manual [14] or the description of the IF-BLOCK-ERROR command in the “Commands Vol. 2” manual [3]).

Representation of command return codes

Command return codes are represented in tabular form in the order: subcode2, subcode1, main code, and meaning. A subcode2 with the value zero (i.e. no additional information present) is not listed in the table.
If guaranteed messages are available for a particular return code, the “Meaning” column in the table is provided with the supplementary heading “Guaranteed messages” (separated by a slash) and the message codes are listed after the text explaining the meanings. Help on these message codes can be obtained by the user with the HELP-MSG command.

Example:

(SC2)	SC1	Main code	Meaning / Guaranteed messages
	0	CMD0001	Command executed normally
1	32	CMD0500	Syntax description in current syntax file invalid Guaranteed message: CMD0500
	64	CMD0554	Command not executed successfully Guaranteed messages: CMD0300, CMD0302, CMD0490, CMD0508, CMD0509, CMD0552, CMD0554, CMD0555, CMD0579

General command return codes

Command return codes that may be passed by BS2000 commands on execution are an integral part of each command description. The descriptions do not include the general return codes that are passed by SDF automatically when:

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

The general command return codes are described in the manual “Introductory Guide to the SDF Dialog Interface” [1] and in “Commands Vol. 1” [3].

2.1 SDF parameters

The names for system and group syntax files are only fixed when SDF is first activated. The names of the system and group syntax files (see page 31f) to be activated and those of the automatically processed LOGON and LOGOFF procedures (see page 34) are stored in an SDF parameter file. This file can subsequently be changed with the MODIFY-SDF-PARAMETERS command or the SDF-PAR utility program.

When SDF is started, the information from the parameter file is read into memory. The syntax files are then activated on the basis of this information. The MODIFY-SDF-PARAMETERS command can be used by systems support to change the names in the parameter file as well as in memory and thus determines how long the changes remain valid, i.e. whether they are permanent or apply to the current session only.

A parameter file with the default name SYSPAR.SDF.041 is supplied as part of SDF; a parameter file with a freely selectable name can be generated by means of the MODIFY-SDF-PARAMETERS command or the SDF-PAR utility.

Two SDF commands are provided for controlling SDF and assigning syntax files:

- MODIFY-SDF-OPTIONS
  - defines criteria local to a task for command and statement input and processing
  - activates or deactivates user syntax files
  - applies to the user job in question.
- MODIFY-SDF-PARAMETERS
  - changes the current and/or permanent definitions for system and group syntax files
  - changes the current and/or permanent definitions for system-wide LOGON and LOGOFF procedures
  - applies throughout the entire system.

When switching syntax files, the following points should be kept in mind:

- The system syntax files are switched immediately for all user jobs on a system-wide level.
- The group syntax file that was valid when the job was started remains valid until the job is terminated. Jobs started after the switch work with the new group syntax file.
- A user syntax file is switched immediately.

2.2 System syntax files

2.2.1 Basic system syntax file

A basic system syntax file (INSF) is activated automatically immediately after SDF is loaded. The file defined in the parameter file is activated. If no parameter file is specified in the DSSM declaration for SDF, then $TSOS.SYSPAR.SDF is used as the parameter file. If the parameter file does not have any valid contents, then a new name is requested via console message CMD0691. When “*STD“ is the reply, $TSOS.SYSSDF.SDF.045 is activated as the basic system syntax file (and $TSOS.SYSSDF.BS2CP.<bs2vers> is activated as the subsystem syntax file).

The activated basic system syntax file can be switched during the session by system administration (see the MODIFY-SDF-PARAMETERS command). The basic system syntax file activated during a session with MODIFY-SDF-PARAMETERS may have any name.

2.2.2 Subsystem syntax files

A basic distinction must be drawn between the following ways of activating subsystem syntax files:

1. Subsystem syntax files activated implicitly when a subsystem is created (see START-SUBSYSTEM in the “Subsystem Management” manual [16]).
2. Local subsystem syntax files activated implicitly when a subsystem is started (see the START-LOCAL-SUBSYSTEM command in the “Subsystem Management” manual [16]).
3. Subsystem syntax files activated explicitly. Like the basic system syntax file, these files are activated automatically if their names are included in the SDF parameter file. Note, too, that system administration can activate or switch subsystem syntax files during an ongoing session with the aid of the MODIFY-SDF-PARAMETERS command. The SCOPE operand defines whether a switch of subsystem syntax files applies to all existing or future tasks.

SUSFs for subsystems loaded prior to “SYSTEM READY” (see the subsystem start attribute CREATION-TIME=*BEFORE-DSSM-LOAD/*AT-DSSM-LOAD) must be entered as a subsystem syntax file in the SDF parameter file. You must make sure that the subsystem syntax file can be accessed during system initialization (i.e. make sure that it can be found on the home pubset, for example).Subsystem syntax files must be cataloged with USER-ACCESS=*SPECIAL.
During installation IMON automatically ensures that the entries are correct.

2.3 Group syntax file

If a group syntax file is to be automatically activated after LOGON processing, this file must have been assigned to the user ID via a profile ID.

This is done as follows:

- A profile ID is defined for the appropriate user ID via the PROFILE-ID operand of the MODIFY-USER-ATTRIBUTES or ADD-USER command.
- The MODIFY-SDF-PARAMETERS command is used to assign a group syntax file to the profile ID.

After LOGON, SDF first looks for the profile ID assigned to the user ID in the user catalog entry. If a profile ID exists, SDF uses the group syntax file currently assigned to this profile ID.

If no file has been assigned to the profile ID, the user job is started without a group syntax file.

If no profile ID has been assigned to the user ID, no group syntax file is activated.

This approach has the following advantage: If a group syntax file is assigned to more than one user ID, it is not necessary to change several user catalog entries when the group syntax file is switched. Instead, these user IDs are allocated a single profile ID whose assignment is then changed centrally in the SDF parameter file.

With the exception of the system administration group syntax files, all group syntax files must be shareable. If the files are protected by means of a basic ACL, ACL or GUARDS, the assigned user IDs must have at least execute privileges. If a group syntax file exists for the user maintenance ID, it must be cataloged with USER-ACCESS=*SPECIAL.

The user catalog entries of certain system IDs (e.g. TSOS) are preset with standard profile IDs. Corresponding default values are also defined for the standard parameter file. To avoid conflicting names, user-defined profile IDs must not start with the string “SYS-”.

A special group syntax file for systems support (TSOS user ID) is not needed since the expanded set of commands is defined in the system syntax files through the assignment of the TSOS system privilege. The privileged commands and operands are therefore only available to user IDs with the TSOS system privilege. SDF automatically checks which system privilege is assigned to a user ID based on the entries in the user catalog.

If a TSOS group syntax file is used, then the following situations can arise:

- A TSOS group syntax file is not declared in SDF parameter file. The TSOS group syntax file may be added at any time by means of the MODIFY-SDF-PARAMETERS command.
- A TSOS group syntax file is declared in the SDF parameter file. If this group syntax file does not exist, system administration receives message CMD0300 and processing continues. The declaration of the TSOS group syntax file may be removed from the SDF parameter file at any time by means of the MODIFY-SDF-PARAMETERS command.

2.4 LOGON and LOGOFF procedures

Procedures can be started automatically during LOGON and LOGOFF processing in the dialog mode and in batch mode. LOGON and LOGOFF procedures can each be provided system-wide or for specific users as call and include procedures. LOGON procedures are supported in SDF V1.4 and higher, LOGOFF procedures in SDF V4.4 and higher.

- The user-specific procedure files are created by the user himself. The user can create one call and one include procedure each for LOGON and LOGOFF processing. They are started automatically when the procedures have the following standard names:
  - $userid.SYS.SDF.LOGON.USERPROC (call procedure for LOGON)
  - $userid.SYS.SDF.LOGON.USERINCL (include procedure for LOGON)
  - $userid.SYS.SDF.LOGOFF.USERPROC (call procedure for LOGOFF)
  - $userid.SYS.SDF.LOGOFF.USERINCL (include procedure for LOGOFF)
- 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 the next LOGON in the current session without changing any entries in the parameter file.

Calling order

If there are system-wide procedure files as well as user-specific procedure files created for a user, then the system-wide LOGON procedures are executed first during LOGON processing, and during LOGOFF processing the user-specific LOGOFF procedures are executed first.

An include procedure is always started before the corresponding call procedure.

Restrictions

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

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 and LOGOFF procedures are called for this user job. Only the user-specific LOGON and LOGOFF procedures, when present, are called in this case.

LOGON and LOGOFF procedures are ignored without warning in the following cases:

– The procedure file is cataloged but does not use any storage space.
– The task is an RFA task.
– The task does not have any privileges other than the HARDWARE-MAINTENANCE, SECURITY-ADMINISTRATION, SAT-FILE-MANAGEMENT and SAT-FILE-EVALUATION privileges.
– For LOGOFF procedures only:
  The task was cancelled with CANCEL-JOB or FORCE-JOB-CANCEL.

No LOGON and LOGOFF procedures are started for some system tasks, too (e.g. during system initialization).

Interruptibility

The LOGON procedures should be declared as uninterruptible with the INTERRUPT-ALLOWED=*NO operand of the BEGIN-PROCEDURE or SET-PROCEDURE-OPTIONS command. This is especially recommended when nested procedures are used. The DO command is not executed within a LOGON procedure. LOGON procedures cannot be terminated with the ENDP-RESUME command, either.

3 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.

3.1 SDF installation files

SDF V4.5 can be run under BS2000 as of BS2000/OSD-BC V3.0.

The following files are part of the standard product and are supplied along with SDF V4.5A:

File	Notes
SIPLIB.SDF.045	TPR macro library
SPMLNK.SDF.045	Load library for SDF V4.5A (for SPARC systems)
SRMLNK.SDF.045	Load library for SDF V4.5A (for RISC systems)
SYSFGM.SDF.045.D	Release notice for SDF in German
SYSFGM.SDF.045.E	Release notice for SDF in English
SYSLIB.SDF.045	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.045	Load library for SDF V4.5A (for /390 systems)
SYSMES.SDF.045	SDF message file
SYSREP.SDF.045	Patches for SDF V4.5A (dummy item)
SYSRMS.SDF.045	RMS for SDF V4.5A
SYSSDF.SDF.045	Syntax file for the commands from the SDF application area
SYSSII.SDF.045 [1)]	Installation information file for installation with IMON
SYSSSC.SDF.045.120	SSCM catalog declarations for the SDF subsystem (BS2000/OSD V3.0 and versions ≥ V4.0)
SYSSSC.SDF.045.121	SSCM catalog declarations for the SDF subsystem (BS2000/OSD V3.1)

File	Notes
SYSMSV.SDF-I.041	Complete system message file for SDF-I
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 75)
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 97)
SYSSDF.SDF-U.041	Syntax file containing the SDF-U statements
SYSLNK.SDF-U.041	Program modules of SDF-U
SYSMSV.SDF-U.041	Complete system message file for SDF-U
SYSMES.SDF-U.041	SDF-U message file
SYSREP.SDF-U.041	Patches for SDF-U (dummy item)
SYSRMS.SDF-U.041	RMS for SDF-U V4.1
SYSSII.SDF-U.041 [1)]	Installation information file for installation with IMON
SYSPRG.SDF-PAR.011	Program for creating, modifying and displaying the SDF parameter file (see chapter “SDF-PAR” on page 127)
SYSMSV.SDF-PAR.011	Complete system message file for SDF-PAR
SYSMES.SDF-PAR.011	SDF-PAR message file
SYSPRC.SDF-PAR.011.100	Startup procedure for SDF-PAR (BS2000/OSD < V2.0)
SYSPRC.SDF-PAR.011.112	Startup procedure for SDF-PAR (BS2000/OSD ≥ V2.0)
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.

3.2 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.045.120** (or with the suffix 121 for BS2000/OSD V3.1).

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.

3.3 Preparing SDF

3.3.1 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.045) as the basic syntax file. All other system syntax files, including SYSSDF.BS2CP.<bs2vers>, are activated as subsystem syntax files.

3.3.2 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.

4 Commands for SDF management

4.1 Overview of functions

Activating and deactivating syntax files

MODIFY-SDF-OPTIONS	Modify SDF settings and activate or deactivate user syntax files
MODIFY-SDF-PARAMETERS	Assign system, subsystem and group syntax files and activate or deactivate system-wide LOGON/ LOGOFF procedures

Display functions

SHOW-SDF-OPTIONS	Output task-specific information on activated syntax files and the definitions for statement input and processing
SHOW-SDF-PARAMETERS	Display assigned group and system syntax files
SHOW-SYNTAX-VERSIONS	Display the syntax files that are integrated in active syntax files

If the SDF-P subsystem is loaded, then the output of the SHOW command can be redirected to S variables. Output can be redirected, for example, by executing the desired SHOW command with EXECUTE-CMD (see the “SDF-P” manual [14]) or the CMD macro (see the “Executive Macros” manual [9]). Information on outputting to S variables can also be found in the “Commands, Volume 6” user guide [4].

The description of further SDF domain commands (with the exception of the START command) can be found in the manual “Introductory Guide to the SDF Dialog Interface” [1].

4.2 Description of the commands

MODIFY-SDF-OPTIONS
Activate user syntax files and modify SDF options

Domain: SDF

Privilege: STD-PROCESSING
TSOS
HARDWARE-MAINTENANCE
SAT-FILE-EVALUATION
SAT-FILE-MANAGEMENT
SECURITY-ADMINISTRATION

The MODIFY-SDF-OPTIONS command is used to

- activate or deactivate user syntax files
- modify the following settings for the input and processing of commands/statements for a specific user job:
  - type of dialog guidance (GUIDANCE operand)
  - extent of logging (LOGGING operand)
  - input interface for utilities (UTILITY-INTERFACE operand)
  - syntax error dialog in procedures (PROCEDURE-DIALOGUE operand)
  - position of the continuation character (CONTINUATION operand)
  - syntax checking of commands (MODE operand)
  - syntax checking of statements (DEFAULT-PROGRAM-NAME operand)
  - selection of function key assignments (FUNCTION-KEYS operand)
  - storage of previous inputs (INPUT-HISTORY operand)

When a task is first started, SDF options are taken from the global information of the activated system or from the group syntax file. If the user subsequently activates a user syntax file, the settings for these options are modified according to the global information contained in that file. Note that GUIDANCE=*EXPERT is automatically set under the OMNIS utility routine when the user syntax file is changed, regardless of what is defined in the global information of the activated user syntax file. In general, if a user syntax file is named $<userid>.SDF.USER.SYNTAX, and a task is started under the user ID <userid>, that syntax file is activated automatically following LOGON processing.

Current settings are displayed in a (temporary) guided dialog as default values in the operand form of the MODIFY-SDF-OPTIONS command. They can also be queried with the SHOW-SDF-OPTIONS command.

MODIFY-**SDF-OPT**IONS	Alias: **MDSDFO**

SYNTAX-FILE = ***UNCHA**NGED / ***ADD**(...) / ***REMOVE**(...) / ***NONE**

***ADD**(...)

ADD-NAME = ***STD** / list-poss(2000): ***STD** / <filename 1..54>

***REMOVE**(...)

REMOVE-**NAME** = ***LAST** / ***ALL** / ***BY-SEL**ECTION / list-poss(2000): <filename 1..54> / ***STD**

,**GUID**ANCE = ***UNCHA**NGED / ***EXPERT** / ***NO** / ***MAX**IMUM / ***MED**IUM / ***MIN**IMUM

,**LOG**GING = ***UNCHA**NGED / ***INPUT-FORM** / ***ACCEPT**ED-**FORM** / ***INVAR**IANT-**FORM**

,**UTILITY-INTERF**ACE = ***UNCHA**NGED / ***OLD-MODE** / ***NEW-MODE**

,**PROC**EDURE-**DIA**LOGUE = ***UNCHA**NGED / ***Y**ES / ***NO**

,**CONT**INUATION = ***UNCHA**NGED / ***OLD-MODE** / ***NEW-MODE**

,**MENU-LOG**GING = ***UNCHA**NGED / ***NO** / ***Y**ES

,**CMD-STATIS**TICS = ***UNCHA**NGED / ***NO** / ***Y**ES

,**MODE** = ***UNCHA**NGED / ***EXEC**UTION / ***TEST**(...)

***TEST**(...)

CHECK-PRIVILEGES = ***UNCHA**NGED / ***NO** / ***Y**ES

,**DEFAULT-PROG**RAM-**NAME** = ***UNCHA**NGED / ***NONE** / <structured-name 1..30>

,**FUNC**TION-**KEYS** = ***UNCH**ANGED / ***STYL**E-**GUI**DE-**MODE** / ***BY-TERM**INAL-**TYPE** / ***OLD-MODE**

,**INP**UT-**HIST**ORY = ***UNCH**ANGED / ***ON**(...) / ***OFF** / ***RESET**

***ON**(...)

NUMBER-**OF-INP**UTS = ***UNCH**ANGED / <integer 1..100>

,**PASS**WORD-**PROT**ECTION = ***UNCHA**NGED / ***NO** / ***Y**ES

SYNTAX-FILE =

Specifies whether a user syntax file is to be activated or deactivated. Two or more user syntax files may be activated simultaneously.
The user syntax file to be activated must be accessible and shareable (if the job submitter is not the owner of the syntax file). If the syntax file is protected by basic ACL, ACL or GUARDS, the job submitter must have at least execute privileges (see the “SECOS” [13] manual).
If the file is protected by an execute password, the password must be contained in the password table for the job (see the description of the ADD-PASSWORD command in the “Commands” [3] manual).

SYNTAX-FILE = *UNCHANGED
The existing value is retained.

SYNTAX-FILE = *ADD(...)
One or more user syntax files are to be additionally activated.

ADD-NAME = list-poss(2000): *STD / <filename 1..54>
Defines the user syntax file to be activated.
More than one syntax file may be specified in a list.

ADD-NAME = *STD
Activates the user syntax file with the default name $<user-id>.SDF.USER.SYNTAX. If this does not exist, no additional user syntax file is activated.

ADD-NAME = <filename 1..54>
Activates the specified user syntax file.

SYNTAX-FILE = *REMOVE(...)
Deactivates one or more user syntax files.

REMOVE-NAME = *LAST / *ALL / list-poss(2000): <filename 1..54> / *STD
Defines the user syntax file to be deactivated.
More than one syntax file may be specified in a list.

REMOVE-NAME = *LAST
Deactivates the last activated user syntax file.

REMOVE-NAME = *ALL
Deactivates all activated user syntax files.

REMOVE-NAME = *BY-SELECTION
SDF displays all activated user syntax files in a selection mask. The user syntax files selected in this mask are deactivated.

REMOVE-NAME = <filename 1..54>
Deactivates the specified user syntax file.

REMOVE-NAME = *STD
Deactivates the user syntax file with the default name $<user-id>.SDF.USER.SYNTAX.

SYNTAX-FILE = *NONE
No user syntax file has been activated for the job.
All currently active user syntax files are deactivated.

GUIDANCE =
Defines the level of user guidance.

GUIDANCE = *UNCHANGED
The existing GUIDANCE definition is retained.

GUIDANCE = *EXPERT
The EXPERT form of unguided dialog is set.
The system requests command input with / and statement input with %//. No syntax error dialog is conducted. Blocked input of commands/statements is possible in the dialog, i.e. several commands/statements, separated by logical end-of-line characters, may be issued at the same time.

GUIDANCE = *NO
The NO form of unguided dialog is set.
Depending on which language is set (E for English; D for German), the system requests command input with “%CMD:” or “%KDO” and statement input with “%STMT:” or “%ANW:”, respectively. A syntax error dialog allows input errors to be corrected without having to repeat the entire command/statement. Blocked input of commands/statements is possible in the dialog, i.e. several commands/statements, separated by logical-end-of-line characters, may be issued at the same time.

GUIDANCE = *MAXIMUM
Guided dialog with maximum user support: includes all alternative operand values with suffixes, and all help texts for the displayed domains, commands, statements and operands.

GUIDANCE = *MEDIUM
Guided dialog with medium user support: includes all alternative operands without suffixes, and all help texts for the displayed domains, commands and statements.

GUIDANCE = *MINIMUM
Guided dialog with minimum support: includes only the default values of operands with no suffixes.

LOGGING =
Defines how input is to be logged. Passwords and entries for secret operands are always blanked out on the screen.

LOGGING = *UNCHANGED
The current specification for logging is retained.

LOGGING = *INPUT-FORM
In unguided dialog, the input strings are logged in their original form. In guided dialog or error dialog, logging is the same as for *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.
Entries made in guided dialog are chained to form a single string.

LOGGING = *INVARIANT-FORM
The following are logged:
– all names with the default name defined in the syntax file
– each specified operand with its name and specified value
– all optional operands implicitly contained in the input, with their names and default values
– any updated versions resulting from a correction dialog.
Entries made in guided dialog are chained to form a single string.

UTILITY-INTERFACE = *UNCHANGED / *OLD-MODE / *NEW-MODE
Controls the input interface of utilities offering an old and a new (SDF) interface in parallel.

PROCEDURE-DIALOGUE = *UNCHANGED / *YES / *NO
Specifies whether, during a procedure run, the user is to be requested to correct any faulty procedure commands (syntax error dialog in procedures). The operand also controls the help query (“?” input) within procedures.

CONTINUATION = *UNCHANGED / *OLD-MODE / *NEW-MODE
Specifies in which column of procedures or ENTER files the continuation character for commands is to be output. If *OLD-MODE is set, the continuation character “-” may only be specified in column 72; if *NEW-MODE is set, it may be entered in any column from columns 2 through 72.

MENU-LOGGING = *UNCHANGED / *NO / *YES
Specifies whether menus of the guided dialog are to be logged in their entirety. This operand is intended for diagnostic purposes only.

CMD-STATISTICS = *UNCHANGED / *NO / *YES
Defines whether statistics on issued commands from the system syntax file are to be generated. This operand is reserved for systems support staff.
When the statistics routine is active (CMD-STATISTICS=*YES), SDF counts every command that is issued. All commands contained in the system syntax file or in subsystem syntax files are taken into account.

CMD-STATISTICS = *NO
Deactivates the statistics routine. SDF evaluates the counter and writes edited statistics to the file $SYS.SDF.CMD.STATISTICS.
The edited statistics file contains the following:
– a header line with the date and time at which the statistics routine was activated and deactivated
– for each subsystem, a header with the name of the subsystem
– for each command of a subsystem, the command name and the counter.

CMD-STATISTICS = *YES
Command statistics are recorded. Whenever a command defined in the basic system syntax file or a subsystem syntax file is called, a counter in the central SDF table is incremented by one. Note that speed suffers to some extent when statistics are activated, because access to the central SDF table is handled by a lock manager. Aliases defined in the system syntax files are counted like commands, because they are entered separately in the SDF command table.

MODE =
Specifies whether subsequent input is to be executed or subjected to a syntax analysis.

MODE = *UNCHANGED
The existing test mode definition is retained.

MODE = *EXECUTION
Deactivates test mode, i.e. entries are merely executed.

MODE = *TEST(...)
Activates test mode, i.e. subsequent commands are not executed, but simply checked for correct syntax. However, the MODIFY-SDF-OPTIONS and SHOW-SDF-OPTIONS commands are always executed.
In the case of S procedures, SDF-P control flow commands are also executed. This can lead to errors, since the commands that declare or set S variables are not executed in test mode.The chargeable subsystem SDF-P includes its own debugging aid for testing S procedures (see the “SDF-P” User Guide [14]). The handling of statements in test mode can be defined in the DEFAULT-PROGRAM-NAME operand.

CHECK-PRIVILEGES = *UNCHANGED / *NO / *YES
Defines whether, in addition to the syntax check, a check is to be carried out as to whether the user possesses the privileges required for input.

CHECK-PRIVILEGES = *NO
The privileges of the user are not checked. This setting may be required, e.g. if the procedure to be checked is created for other users.

CHECK-PRIVILEGES = *YES
The privileges of the user are checked in addition to the syntax.

DEFAULT-PROGRAM-NAME =
Used only if MODE=*TEST has been specified; defines a program name for checking the syntax of the statements. All statements defined under this program name are checked, using the syntax descriptions as a reference.
Statements in S procedures cannot be checked.

DEFAULT-PROGRAM-NAME = *UNCHANGED
The current definition of the program name is retained.

DEFAULT-PROGRAM-NAME = *NONE
All statements are ignored with MODE=*TEST(...).

DEFAULT-PROGRAM-NAME = <structured-name 1..30>
Program name (ADD-PROGRAM NAME= ..., see “SDF-A” [7]). If MODE=*TEST(...) is specified, the syntax check is executed for the specified program name. Syntax analysis is independent of the program call as LOAD-/START-EXECUTABLE-PROGRAM (or LOAD-/START-PROGRAM) are not executed in test mode.

FUNCTION-KEYS =
Defines function key assignments. A detailed description of the different modes can be found in the “Introductory Guide to the SDF Dialog Interface” [1]. Unsupported function keys do nothing; they do not have the same effect as the DUE key.

FUNCTION-KEYS = *UNCHANGED
Function key assignments are not changed.

FUNCTION-KEYS = *STYLE-GUIDE-MODE
Function keys are assigned in accordance with the Fujitsu Siemens 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 Fujitsu Siemens 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.

FUNCTION-KEYS = *OLD-MODE
Function key 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)

INPUT-HISTORY = *UNCHANGED / *ON(...) / *OFF / *RESET
Specifies whether the input buffer is to be turned on, turned off, or reset.

INPUT-HISTORY = *ON(...)
The input buffer is turned on, and SDF saves all syntactically correct inputs (commands and statements) in it. SET-LOGON-PARAMETERS, RESTORE-SDF-INPUT and SHOW-INPUT-HISTORY are not saved.
The specifications in the PASSWORD-PROTECTION operand define whether ISP commands are saved.
The saved inputs can be output by the user by means of the SHOW-INPUT-HISTORY command. 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 as plain text, dependent on 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.

NUMBER-OF-INPUTS = *UNCHANGED / <integer 1..100>
Defines the maximum number of inputs that can be saved in the input buffer. When the maximum number is reached, the oldest respective input is deleted whenever a new input is saved.

PASSWORD-PROTECTION = *UNCHANGED / *NO / *YES
Defines whether values for “secret” operands and ISP commands are saved in the input buffer.

PASSWORD-PROTECTION = *NO
Values for “secret” operands are saved in plain text. ISP commands are also saved in the input buffer.

PASSWORD-PROTECTION = *YES
Values for “secret” operands are saved in “^” (corresponds to the operand value *SECRET). ISP commands are not saved in the input buffer.

INPUT-HISTORY = *OFF
The input buffer is turned off. Subsequent inputs are not stored; but inputs saved earlier remain accessible.

INPUT-HISTORY = *RESET
The input buffer is reset, i.e. all saved entries are deleted and no longer accessible. Subsequent inputs are saved again.

Command return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed normally
1	32	CMD0500	Syntax description in current syntax file invalid Guaranteed message: CMD0500
	64	CMD0554	Command not successful Guaranteed messages: CMD0300, CMD0302, CMD0490, CMD0508, CMD0509, CMD0552, CMD0554, CMD0555, CMD0579

Example

```
/show-sdf-opt ——————————————————————————————————————————————— (1)
%SYNTAX FILES CURRENTLY ACTIVATED :
%  SYSTEM    : :2OSH:$TSOS.SYSSDF.SDF.045
%              VERSION : SESD04.5A300
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.ACS.140
%              VERSION : SESD14.0B100
 .
 .
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.SDF-A.041
%              VERSION : SESD04.1E10
%  SUBSYSTEM : :2OSH:$TSOS.SYSSDF.TASKDATE.140
%              VERSION : SESD14.0A100
%  GROUP     : *NONE
%  USER      : :2OSG:$USER1.SDF.USER.SYNTAX
%              VERSION : 13.11.2000
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
/modify-sdf-opt syntax-file=*add(syssdf.example.03) ————————————— (2)
/show-sdf-opt information=*user ————————————————————————————————— (3)
%  USER      : :2OSG:$USER1.SDF.USER.SYNTAX
%              VERSION : 13.11.2000
%  USER      : :2OSG:$USER1.SYSSDF.EXAMPLE.03
%              VERSION : 16.10.2001
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
```

```
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
/modify-sdf-opt guid=*max ──────────────────────────────────────────── (4)
  .
  .
   (in an interactive dialog:)
   NEXT = mod-sdf-opt synt-file=*remove,guid=*expert ──────────────── (5)
  .
/show-sdf-opt information=*user ─────────────────────────────────────── (6)
%  USER      : :2OSG:$USER1.SDF.USER.SYNTAX
%              VERSION : 13.11.2000
%CURRENT SDF OPTIONS :
%  GUIDANCE           : *EXPERT
%  LOGGING            : *INPUT-FORM
%  CONTINUATION       : *NEW-MODE
%  UTILITY-INTERFACE  : *NEW-MODE
%  PROCEDURE-DIALOGUE : *NO
%  MENU-LOGGING       : *NO
%  MODE               : *EXECUTION
%     CHECK-PRIVILEGES   : *YES
%  DEFAULT-PROGRAM-NAME : *NONE
%  FUNCTION-KEYS      : *STYLE-GUIDE-MODE
%  INPUT-HISTORY      : *ON
%     NUMBER-OF-INPUTS   : 20
%     PASSWORD-PROTECTION: *YES
```

(1) SHOW-SDF-OPTIONS is used to request information on the syntax files activated for this user job and on the valid settings for command processing. The user syntax file SDF.USER.SYNTAX is active in addition to the basic system syntax file and a number of subsystem syntax files. User guidance has been set to expert mode.

(2) A user syntax file SYSSDF.EXAMPLE.03 is additionally activated.

(3) The user-specific information is displayed. The old user syntax file SDF.USER.SYNTAX remains active and the user syntax file SYSSDF.EXAMPLE.03 is additionally activated. The command definitions from SYSSDF.EXAMPLE.03 are used for commands defined in both user syntax files, as these are activated last.

(4) The system switches to guided dialog mode.

(5) The most recently activated user syntax file is deactivated and the system switches back to unguided dialog mode.

(6) The repeated output of the user-specific SDF options shows that only the user syntax file SYSSDF.EXAMPLE.03 was deactivated.

MODIFY-SDF-PARAMETERS
Modify SDF parameters

Domain: SDF

Privilege: TSOS

The MODIFY-SDF-PARAMETERS command enables systems support staff to create/update the SDF parameter file and thereby exchange the system syntax files, define the assignment of group syntax files to users, and define system-wide LOGON or LOGOFF procedures (one call and one include procedure).

MODIFY-**SDF-PAR**AMETERS

SCOPE = ***TEMP**ORARY / ***PERMANENT** / ***NEXT-SESSION**(...)

***NEXT-SESSION**(...)

PARAMETER-**FILE-NAME** = ***CURR**ENT / <filename 1..54>

,**SYNTAX-F**ILE-**TYPE** = ***UNCHA**NGED / ***SYS**TEM(...) / ***GR**OUP(...) / ***SUBSYS**TEM(...)

***SYS**TEM(...)

NAME = <filename 1..54>

***GR**OUP(...)

NAME = ***NO**NE / <filename 1..54>

,**PROF**ILE-**ID** = <structured-name 1..30> / <filename 1..54>

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

***SUBSYS**TEM(...)

NAME = ***NO**NE / <filename 1..54>

,**SUBSYS**TEM-**NAME** = <structured-name 1..8>

,**SYS**TEM-**LOGON-PROC** = ***UNCHA**NGED / ***NO** / ***STD** / <filename 1..54>

,**SYS**TEM-**LOGON-INCL** = ***UNCHA**NGED / ***NO** / ***STD** / <filename 1..54>

,**SYS**TEM-**LOGOFF-PROC** = ***UNCHA**NGED / ***NO** / ***STD** / <filename 1..54>

,**SYS**TEM-**LOGOFF-INCL** = ***UNCHA**NGED / ***NO** / ***STD** / <filename 1..54>

SCOPE =
Defines the scope of activation. If SCOPE=*TEMPORARY or *PERMANENT is specified, the group syntax file belonging to the profile ID of the user ID TSOS cannot be deactivated. If the case of SCOPE=*PERMANENT or *NEXT-SESSION, an SDF parameter file is created if none already exists.

SCOPE = *TEMPORARY
For system syntax files:
The specified system syntax file is activated globally for all user jobs. However, it is not stored in the SDF parameter file, nor is it taken into account for the next session.

For subsystem syntax files:
The specified subsystem syntax file is activated or deactivated globally for all user jobs. However, it is not stored in the SDF parameter file, nor is it taken into account for the next session.

For group syntax files:
The specified group syntax file is assigned to the specified profile ID. This assignment is only effective for future user jobs (next LOGON in the current session). Existing user jobs are not affected. The specified group syntax file name is not stored in the SDF parameter file and is therefore not taken into account in the next session. In the event of deactivation (*NONE), the existing profile ID and the related group syntax file name are *not* deleted from the SDF parameter file.

*With SYSTEM-LOGON-PROC or SYSTEM-LOGON-INCL = *NO:*
The system-wide LOGON procedure is deactivated for all subsequent LOGON processing in the current session. The SDF parameter file is not changed.

*With SYSTEM-LOGON-PROC or SYSTEM-LOGON-INCL = <filename 1..54> or *STD:*
The specified system-wide LOGON procedure is activated for all subsequent LOGON processing in the current session. Since the SDF parameter file is not changed, this definition is applicable only to the current session.

*With SYSTEM-LOGOFF-PROC or SYSTEM-LOGOFF-INCL = *NO:*
The system-wide LOGOFF procedure is deactivated for all following LOGOFF processing in the current session. The SDF parameter file is not changed.

*With SYSTEM-LOGOFF-PROC or SYSTEM-LOGOFF-INCL = <filename 1..54> or *STD:*
The specified system-wide LOGOFF procedure is activated for all following LOGOFF processing in the current session. Since the SDF parameter file is not changed, this specification is only valid for the current session.

SCOPE = *PERMANENT

For system syntax files:
The specified system syntax file is activated globally for all user jobs. In addition, this syntax file name is stored in the SDF parameter file and thus taken into account in the next session.

For subsystem syntax files:
The specified subsystem syntax file is activated globally for all user jobs. In addition, this syntax file name is stored in the SDF parameter file and thus taken into account in the next session. If the subsystem syntax file is subsequently deactivated, the name of the subsystem and the corresponding subsystem syntax file are deleted from the SDF parameter file.

For group syntax files:
The specified group syntax file is assigned to the specified profile ID. This assignment is only effective for future user jobs (next LOGON in the current session). Existing user jobs are not affected. The specified group syntax file is stored in the SDF parameter file and taken into account in the next session. In the event of deactivation (*NONE), the existing profile ID and the associated group syntax file name are deleted from the SDF parameter file.

*With SYSTEM-LOGON-PROC or SYSTEM-LOGON-INCL = *NO:*
The system-wide LOGON procedure is deactivated for all subsequent LOGON processing in the current session. The name of the procedure is deleted from the SDF parameter file. As of the next session, the system-wide LOGON procedure with the default name is used if it exists.

*With SYSTEM-LOGON-PROC or SYSTEM-LOGON-INCL = <filename 1..54> or *STD:*
The specified system-wide LOGON procedure is activated for all subsequent LOGON processing in the current session. The name of the procedure is stored in the SDF parameter file and is thus also taken into account in the next session.

*With SYSTEM-LOGOFF-PROC or SYSTEM-LOGOFF-INCL = *NO:*
The system-wide LOGOFF procedure is deactivated for all following LOGOFF processing in the current session. Its name is deleted from the SDF parameter file. When present, the system-wide LOGOFF procedure with the standard name is used starting with the next session.

*With SYSTEM-LOGOFF-PROC or SYSTEM-LOGOFF-INCL = <filename 1..54> or *STD:*
The specified system-wide LOGOFF procedure is activated for all following LOGOFF processing in the current session. Its name is stored in the SDF parameter file and is therefore taken into account in the next session.

SCOPE = *NEXT-SESSION(...)

For system syntax files:
The specified system syntax file name is stored in the SDF parameter file without being checked and is taken into account in the next session. The check occurs in the next session. The current session is not affected.

For subsystem syntax files:
The specified subsystem syntax file is stored in the SDF parameter file without being checked and is taken into account in the next session. The check occurs in the next session. The current session is not affected. If the subsystem syntax file is deactivated, the names of the subsystem and the corresponding subsystem syntax file are deleted from the SDF parameter file (the current session is not affected).

For group syntax files:
The specified group syntax file is assigned to the specified profile ID in the SDF parameter file. The assignment is not taken into account until the next session. Existing user jobs and subsequent user jobs of the current session are not affected. The specified syntax file name is stored in the SDF parameter file without being checked. The check occurs in the next session. In the event of deactivation (*NONE), the existing profile ID and the associated group syntax file name are removed from the SDF parameter file.

*With SYSTEM-LOGON-PROC or SYSTEM-LOGON-INCL = *NO:*
The name of the system-wide LOGON procedure is deleted from the SDF parameter file. As of the next session, the system-wide LOGON procedure with the default name is used if it exists. This has no effect on the current session.

*With SYSTEM-LOGON-PROC or SYSTEM-LOGON-INCL = <filename 1..54> or *STD:*
The name of the specified system-wide LOGON procedure is stored in the SDF parameter file. This definition applies as of the next session and has no effect on the current session.

*With SYSTEM-LOGOFF-PROC or SYSTEM-LOGOFF-INCL = *NO:*
The name of the system-wide LOGOFF procedure is deleted from the SDF parameter file. When present, the system-wide LOGOFF procedure with the standard name is used starting with the next session. This has no effect on the current session.

*With SYSTEM-LOGOFF-PROC or SYSTEM-LOGOFF-INCL = <filename 1..54> or *STD:*
The name of the specified system-wide LOGOFF procedure is stored in the SDF parameter file. The specification is valid starting with the next session has no effect on the current session.

PARAMETER-FILE-NAME = *CURRENT / <filename 1..54>
Defines the name of the SDF parameter file to be created or updated. The default value is *CURRENT, i.e. the SDF parameter file of the currently running session.

SYNTAX-FILE-TYPE =
Specifies the type of syntax file.

SYNTAX-FILE-TYPE = *SYSTEM(...)
The syntax file is a system syntax file.

NAME = <filename 1..54>
Specifies the name of the file to be used as the basic system syntax file.

SYNTAX-FILE-TYPE = *GROUP(...)
The syntax file is a group syntax file.

NAME =
Specifies the name of the file to be used as the group syntax file.

NAME = *NONE
The assignment of the group syntax file to the specified profile ID is deleted.

NAME = <filename 1..54>
The specified group syntax file is assigned to the specified profile ID.

PROFILE-ID = <structured-name 1..30>
Defines the profile ID which was (or is to be) assigned to the group syntax file.

HIERARCHY =
Specifies whether or not the SDF file hierarchy is to be retained for the syntax analysis of the commands/statements of a user job with the defined 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 when setting up the user job. The SDF file hierarchy is retained.

HIERARCHY = *NO
The system syntax files are deactivated immediately after LOGON processing. This means that the definitions in the system syntax files are meaningless for users with the defined profile ID. Only the definitions stored in the associated group syntax file are valid. A group syntax file defined with HIERARCHY=*NO **must** contain the EXIT-JOB or LOGOFF command in addition to the global information, otherwise it is not possible to terminate a user job which is assigned the specified profile ID.

SYNTAX-FILE-TYPE = *SUBSYSTEM(...)
The syntax file is of type “system” and is used in addition to the already active system syntax files. It appears as a subsystem syntax file in the parameter file. The global information and application area in this syntax file are ignored.

NAME = *NONE / <filename 1..54>
Defines the name of the file to be used as the subsystem syntax file. If *NONE is specified, the subsystem syntax file defined by SUBSYSTEM-NAME is deactivated. *NONE may only be specified if no DSSM subsystem is involved.

SUBSYSTEM-NAME = <structured-name 1..8>
Specifies the name of the subsystem to which the subsystem syntax file belongs. The subsystem does not have to be a subsystem generated by DSSM.

SYSTEM-LOGON-PROC =
Determines whether or not a system-wide LOGON procedure should run as a call procedure. The SCOPE operand determines the effect of the changes for the system-wide LOGON call procedure.

SYSTEM-LOGON-PROC = *UNCHANGED
No change in the system-wide LOGON call procedure.

SYSTEM-LOGON-PROC = *NO
Deactivates the system-wide LOGON call procedure (see the SCOPE operand).

SYSTEM-LOGON-PROC = *STD
The system-wide LOGON call procedure with the default name $TSOS.SYS.SDF.LOGON.SYSPROC is activated (see the SCOPE operand).

SYSTEM-LOGON-PROC = <filename 1..54>
The specified system-wide LOGON call procedure is activated (see the SCOPE operand).

SYSTEM-LOGON-INCL =
Determines whether or not a system-wide LOGON procedure should run as an include procedure. The SCOPE operand determines the effect of the changes for the system-wide LOGON include procedure.

SYSTEM-LOGON-INCL = *UNCHANGED
No change in the system-wide LOGON include procedure.

SYSTEM-LOGON-INCL = *NO
Deactivates the system-wide LOGON include procedure (see the SCOPE operand).

SYSTEM-LOGON-INCL = *STD
The system-wide LOGON include procedure with the default name $TSOS.SYS.SDF.LOGON.SYSINCL is activated (see the SCOPE operand).

SYSTEM-LOGON-INCL = <filename 1..54>
The specified system-wide LOGON include procedure is activated (see the SCOPE operand).

SYSTEM-LOGOFF-PROC =
Specifies if a system-wide LOGOFF procedure is to run as a call procedure. The specifications for the system-wide LOGOFF call procedure have different effects depending on the value of the SCOPE operand.

SYSTEM-LOGOFF-PROC = *UNCHANGED
Do not change the system-wide LOGOFF call procedure.

SYSTEM-LOGOFF-PROC = *NO
The system-wide LOGOFF call procedure is deactivated (see also the SCOPE operand).

SYSTEM-LOGOFF-PROC = *STD
The system-wide LOGOFF call procedure with the standard name $TSOS.SYS.SDF.LOGOFF.SYSPROC is activated (see also the SCOPE operand).

SYSTEM-LOGOFF-PROC = <filename 1..54>
The specified system-wide LOGOFF call procedure is activated (see also the SCOPE operand).

SYSTEM-LOGOFF-INCL =
Specifies if a system-wide LOGOFF procedure is to run as an include procedure. The specifications for the system-wide LOGOFF include procedure have different effects depending on the value of the SCOPE operand.

SYSTEM-LOGOFF-INCL = *UNCHANGED
Do not change the system-wide LOGOFF include procedure.

SYSTEM-LOGOFF-INCL = *NO
The system-wide LOGOFF include procedure is deactivated (see also the SCOPE operand).

SYSTEM-LOGOFF-INCL = *STD
The system-wide LOGOFF include procedure with the standard name $TSOS.SYS.SDF.LOGOFF.SYSINCL is activated (see also the SCOPE operand).

SYSTEM-LOGOFF-INCL = <filename 1..54>
The specified system-wide LOGOFF include procedure is activated (see also the SCOPE operand).

Command return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed normally
2	0	CMD0677	Access rights for system syntax file changed (warning) Guaranteed message: CMD0677
2	0	CMD0689	Specification for HIERARCHY operand ignored (warning) Guaranteed message: CMD0689
1	32	CMD0500	Syntax description in current syntax file invalid Guaranteed message: CMD0500
	64	CMD0556	Command not successful Guaranteed messages: CMD0300, CMD0302, CMD0490, CMD0508, CMD0509, CMD0556, CMD0557, CMD0671, CMD0672, CMD0674, CMD0678, CMD0679, CMD0681, CMD0682, CMD0687, CMD0688, CMD0690, CMD0814, CMD0815
1	64	CMD0601	Command reserved for systems support staff Guaranteed message: CMD0601

Notes

- Access to the SDF parameter file (MODIFY-SDF-PARAMETERS command) is possible only for a task under the TSOS ID. Concurrent accesses of further tasks are rejected with an error message.
- The group syntax file allocated to the profile ID of systems support (TSOS) need not be shareable unless it is used by other user IDs. If the file is protected by BASIC-ACL or GUARDS, the user IDs must have at least execute permission.
- If SCOPE=*TEMPORARY or *PERMANENT is specified, the TSOS group syntax file cannot be deactivated.
- If SCOPE=*NEXT-SESSION is specified, specified syntax file names and the names of the system LOGON procedures (call and include procedures) are stored in the parameter file without being checked. Checking does not take place until the start of the next session.
- File names specified without a user ID are given the current catalog ID or user ID of the task.
- When replacing alias names, only the real file names are entered.
- Global information from subsystem syntax files is ignored.

- A subsystem syntax file may also be activated automatically by DSSM when the subsystem is started (defined in the subsystem declarations). If a subsystem syntax file has already been declared and hence activated for the subsystem in the SDF parameter file, only the last syntax file activated for the subsystem is evaluated.
  A syntax file automatically activated by DSSM cannot be removed for the subsystem by means of MODIFY-SDF-PARAMETERS; it can only be exchanged.
  Syntax files automatically activated by DSSM are deactivated again by DSSM when the subsystem is shut down.
  Syntax files activated by means of the SDF parameter file must in all cases be deactivated with a suitable MODIFY-SDF-PARAMETERS command.

Examples

1. *Defining a basic system syntax file for the next BS2000 session*

```
/MODIFY-SDF-PARAMETERS SYNTAX-FILE-TYPE=*SUBSYSTEM(-
/                      NAME=SYSSDF.RZ-TOOLS.010,SUBSYSTEM=RZTOOLS),-
/                        SCOPE=*NEXT-SESSION
```

   The name of the subsystem syntax file SYSSDF.RZ-TOOLS.010 for the RZTOOLS subsystem created by the computer center is stored in the SDF parameter file. The name of the SDF parameter file is either defined in the current DSSM catalog or corresponds to the default name $.SYSPAR.SDF.
   The specified basic system syntax file is activated when SDF is loaded at the next BS2000 startup (NEXT-SESSION).

2. *Defining a new group syntax file for the privileged user ID TSOS*

   The TSOS ID has the predefined profile ID SYS-TSOS. A new group syntax file can be defined by means of the MODIFY-SDF-PARAMETERS command.

```
/MODIFY-SDF-PARAMETERS SYNTAX-FILE-TYPE=*GROUP(NAME=SYS.SDF.TSOS.0002,-
/                                              PROFILE-ID=SYS-TSOS),-
/                      SCOPE=*PERMANENT
```

   The specified group syntax file is assigned to profile ID SYS-TSOS.
   The specified group syntax file is activated at the next LOGON issued by the TSOS ID.
   Group syntax file assignment is permanent, i.e. retained beyond the ongoing session.
   The name of the group syntax file is entered in the current SDF parameter file.

3. *Assigning a group syntax file to a user group*

   Before a group of users can be assigned a group syntax file, the same profile ID has to be defined for each of the user IDs.

```
/MODIFY-USER-ATTRIBUTES USER-ID=A,PROFILE-ID=SDF-A-ADM
/MODIFY-USER-ATTRIBUTES USER-ID=B,PROFILE-ID=SDF-A-ADM
/MODIFY-USER-ATTRIBUTES USER-ID=C,PROFILE-ID=SDF-A-ADM
/MODIFY-SDF-PARAMETERS SYNTAX-FILE-TYPE=*GROUP(NAME=SYS.SDF-A-GROUP,-
/                                              PROFILE-ID=SDF-A-ADM),-
/                      SCOPE=*PERMANENT
```

   The group syntax file SYS.SDF-A-GROUP is assigned to user IDs A, B and C via the common profile ID SDF-A-ADM. The name of the group syntax file and the associated profile ID are entered in the current SDF parameter file.

SHOW-SDF-OPTIONS
Display active syntax files and SDF settings

Domain: SDF

Privilege: STD-PROCESSING
TSOS
HARDWARE-MAINTENANCE
SAT-FILE-EVALUATION
SAT-FILE-MANAGEMENT
SECURITY-ADMINISTRATION

The SHOW-SDF-OPTIONS command outputs the names and versions of the currently active syntax files as well as the current settings of the SDF options. The real file name is always shown for syntax files, even if the user has designated a user syntax file in the MODIFY-SDF-OPTIONS command via its alias defined in the alias catalog (see the ADD-ALIAS-CATALOG-ENTRY command).

The INFORMATION operand controls the scope of the information to be shown. INFORMATION=*USER restricts the information to user-specific settings that can only be modified by the user (using MODIFY-SDF-OPTIONS) within the local task.

The command supports structured output in S variables (see page 64 or the “Commands” manual, Volume 6 [4]).

SHOW-SDF-OPTIONS	Alias: **SHSDFO**
INFORMATION = ***ALL** / ***USER**	

INFORMATION =
Defines whether all or only user-specific information is to be output.

INFORMATION = *ALL
All information on active syntax files and SDF settings is output. This also includes the user-specific information.

INFORMATION = *USER
Only user-specific information is output. This includes the name and version of the activated user syntax file and the options set with MODIFY-SDF-OPTIONS.

Command return codes:

(SC2) SC1	Maincode	Meaning
0	CMD0001	Command executed normally

Output in S variables

The output of the SHOW-SDF-OPTIONS command can be redirected to an S variable, provided SDF-P is loaded on your system. The S variable must be declared as a list variable of type “structure” (see the DECLARE-VARIABLE command in the “SDF-P” manual [14]).

Output information	Name of the S variable	T	Contents
Checking of user privileges	var(*LIST).CHECK-PRIV	S	*YES *NO
Statistics for commands from the system syntax file	var(*LIST).CMD-STATIS	S	(empty string) *NO *YES
Position of the continuation character on command input	var(*LIST).CONTI	S	*NEW-MODE *OLD-MODE
Function key assignments	var(*LIST).FUNC-KEY	S	*OLD-MODE *STYLE-GUIDE-MODE
Type of user guidance	var(*LIST).GUIDE	S	*EXPERT *MAX *MED *MIN *NO
History of inputs	var(*LIST).INPUT-HIST	S	*OFF *ON
Method of logging inputs	var(*LIST).LOG	S	*ACCEPT-FORM *INPUT-FORM *INVARIANT-FORM
Menu logging	var(*LIST).MENU-LOG	S	*NO *YES
Syntax check for command	var(*LIST).MODE	S	*EXEC *TEST
Size of the input buffer	var(*LIST).NUM-OF-INPUT	I	<integer 0..100>
Protection of secret operand values in the input buffer	var(*LIST).PASSWORD-PROT	S	*YES *NO
Syntax error dialog	var(*LIST).PROC-DIALOG	S	*NO *YES
Name of the syntax file	var(*LIST).SF(*LIST).F-NAME	S	<filename 1..54>

continued ➠

Output information	Name of the S variable	T	Contents
Type of syntax file	var(*LIST).SF(*LIST).TYPE	S	*GROUP *SUBSYS *SYS *USER
Version of syntax file	var(*LIST).SF(*LIST).VERSION	S	<text 1..12>
Name of test program to check syntax of SDF statements	var(*LIST).TEST-PROG-NAME	S	*NONE <structured-name 1..30>
Input interface for utilities	var(*LIST).UTILITY-INTERF	S	*NEW-MODE *OLD-MODE

Notes on the contents of variables

CMD-STATIS: For user jobs without TSOS privileges, the variable contains an empty string.

F-NAME: Name of a syntax file with the complete path

TYPE: Only the information on active syntax file types is output. If no subsystem syntax files are active, for example, no list element of type *SUBSYS is output. Exception: If there is no user syntax file activated, then the list element of type *USER is created once anyway (F-NAME and VERSION contain only an empty string in this case).
If INFORMATION=*USER was specified, the variable will not contain any information on system, subsystem and group syntax files.

VERSION: The syntax file version defined in the global information of the syntax file. If no version was specified there, then the variable contains the value UNDEFINED.

All other fields correspond to settings that can be defined with the MODIFY-SDF-OPTIONS command.

SHOW-SDF-PARAMETERS
Display SDF parameters

Domain: SDF

Privilege: TSOS

The SHOW-SDF-PARAMETERS command provides information on the entries in an SDF parameter file. The names of the syntax files and all assignments of profile IDs to a group syntax file are output. The names of the system-wide LOGON or LOGOFF procedures (call and include procedures) can also be output if required.

The command supports structured output in S variables (see page 69 or the “Commands” manual, Volume 6 [4].

```
SHOW-SDF-PARAMETERS

SCOPE = *TEMPORARY / *NEXT-SESSION(...)
    *NEXT-SESSION(...)
        PARAMETER-FILE-NAME = *CURRENT / <filename 1..54>
,SYNTAX-FILE-TYPE = *ALL / *SYSTEM / *GROUP(...) / *SUBSYSTEM(...) / *NONE
    *GROUP(...)
        PROFILE-ID = *ALL / <structured-name 1..30> / <filename 1..54>
    *SUBSYSTEM(...)
        SUBSYSTEM-NAME = *ALL / <structured-name 1..8>
,SYSTEM-LOGON-PROC = *YES / *NO
,SYSTEM-LOGON-INCL = *YES / *NO
,SYSTEM-LOGOFF-PROC = *YES / *NO
,SYSTEM-LOGOFF-INCL = *YES / *NO
```

SCOPE =
Defines the scope of the output based on the activated syntax files.

SCOPE = *TEMPORARY
The syntax files activated during the current session and/or the system-wide LOGON or LOGOFF procedure are output.

SCOPE = *NEXT-SESSION(...)
The syntax files and/or the system-wide LOGON or LOGOFF procedure contained in an SDF parameter file are output.

PARAMETER-FILE-NAME = *CURRENT / <filename 1..54>
Defines the name of the SDF parameter file containing the desired information.

The displayed name of the SDF parameter file can be interpreted as the result of a file name replacement if aliases are used via ACS. The fully qualified, real file name is only saved and output by SDF if no alias was defined for the SDF parameter file.

PARAMETER-FILE-NAME = *CURRENT
The desired information is taken from the SDF parameter file that was used at the start of the current session.

SYNTAX-FILE-TYPE =
Defines the syntax file for which information is to be displayed.

SYNTAX-FILE-TYPE = *ALL
The names of the (basic) system syntax file, of the subsystem syntax files, and of the group syntax files (with the associated profile IDs) are displayed.

SYNTAX-FILE-TYPE = *SYSTEM
The name of the basic system syntax file name is displayed.

SYNTAX-FILE-TYPE = *GROUP(...)
Specifies that group syntax file names are to be displayed.

PROFILE-ID =
Specifies which group syntax file names are to be displayed.

PROFILE-ID = *ALL
All group syntax files (with associated profile IDs) are displayed.

PROFILE-ID = <structured-name 1..30> / <filename 1..54>
Defines the profile ID for which the associated group syntax file name is displayed. The value <filename> is supported only for reasons of compatibility and is not permitted during guided dialog.

SYNTAX-FILE-TYPE = *SUBSYSTEM(...)
The names of the subsystem syntax files are displayed.

SUBSYSTEM-NAME =
Defines the subsystems for which the names of the subsystem syntax files will be displayed.

SUBSYSTEM-NAME = *ALL
The names of all the subsystem syntax files are displayed.

SUBSYSTEM-NAME = <structured-name 1..8>
Only the name of the subsystem syntax file belonging to the specified subsystem is displayed.

SYNTAX-FILE-TYPE = *NONE
No syntax file names are displayed.

SYSTEM-LOGON-PROC = *YES / *NO
Defines whether or not the name of the system-wide LOGON call procedure (called with the CALL-PROCEDURE command) is to be displayed.

SYSTEM-LOGON-INCL = *YES / *NO
Defines whether or not the name of the system-wide LOGON include procedure (called with the INCLUDE-PROCEDURE command) is to be displayed.

SYSTEM-LOGOFF-PROC = *YES / *NO
Specifies if the name of the system-wide LOGOFF call procedure (called with the CALL-PROCEDURE command) is to be displayed or not.

SYSTEM-LOGOFF-INCL = *YES / *NO
Specifies if the name of the system-wide LOGOFF include procedure (called with the CALL-PROCEDURE command) is to be displayed or not.

Command return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed normally
1	32	CMD0500	Syntax description in current syntax file invalid Guaranteed message: CMD0500
1	64	CMD0601	Command reserved for systems support staff Guaranteed message: CMD0601
1	64	CMD0680	SDF parameter file invalid Guaranteed messages: CMD0300, CMD0680, CMD0687

Note

Subsystem syntax files that are located on a pubset that is not available during system initialization are not displayed. These syntax files can only be activated when importing the pubset.

Output in S variables

The output of the SHOW-SDF-PARAMETERS command can be redirected to an S variable, provided SDF-P is loaded on your system. The S variable must be declared as a list variable of type “structure” (see the DECLARE-VARIABLE command in the “SDF-P” [14] User Guide).

Output information	Name of the S variable	T	Contents
Scope of the active syntax files	var(*LIST).SCOPE	S	*TEMP <filename 1..54>
Name of the syntax file	var(*LIST).SF(*LIST).F-NAME	S	<filename 1..54>
Syntax analysis including system syntax file	var(*LIST).SF(*LIST).HIERARCHY	S	(empty string) *NO *YES
Name of the subsystem	var(*LIST).SF(*LIST).NAME	S	(empty string) <structured-name 1..8> BS2000
Profile ID for the group syntax file	var(*LIST).SF(*LIST).PROF-ID	S	(empty string) <structured-name 1..30>
Status of the subsystem	var(*LIST).SF(*LIST).STATE	S	(empty string) ACTIVE COEXISTENT DEACTIVATED HOLD IN-EXCHA IN-HOLD TO-IMPORT
Type of syntax file	var(*LIST).SF(*LIST).TYPE	S	*GROUP *SUBSYS *SYS
Version of the syntax file	var(*LIST).SF(*LIST).VERSION	S	<text 1..12>
Name of the system-wide LOGOFF call procedure	var(*LIST).SYS-LOGOFF-PROC	S	*STD <filename 1..54>
Name of the system-wide LOGOFF include procedure	var(*LIST).SYS-LOGOFF-INCL	S	*STD <filename 1..54>
Name of the system-wide LOGON call procedure	var(*LIST).SYS-LOGON-PROC	S	*STD <filename 1..54>
Name of the system-wide LOGON include procedure	var(*LIST).SYS-LOGON-INCL	S	*STD <filename 1..54>

Notes on the contents of variables

SCOPE: If SCOPE=*NEXT-SESSION(...) is specified, the name of the parameter file is output as <filename 1..54>.

F-NAME: Name of a syntax file with the complete path.

HIERARCHY: Information corresponding to the settings in the MODIFY-SDF-PARAMETERS command. For TYPE=*SYS and TYPE=*SUBSYS, the field contains the empty string.

NAME: Name of the subsystem for TYPE=*SUBSYS. If TYPE=*SYS is specified, the field contains the value 'BS2000'; if TYPE=*GROUP is specified, the field contains the empty string.

PROF-ID: For TYPE=*SYS and TYPE=*SUBSYS, the field contains the empty string.

STATE: Describes the status of a subsystem if it was activated by DSSM. The field will contain the empty string in the following cases:
– if TYPE=*SYS was specified
– if TYPE=*GROUP was specified
– if the path name of the parameter file was specified in the command with SCOPE=*NEXT-SESSION(...).

The status of subsystems not activated by DSSM is always displayed as ACTIVE except in the following situations:
– If the subsystem syntax file could not be activated during system initialization, then TO-IMPORT is displayed.
– If the subsystem syntax file was deactivated by IMON, then DEACTIVATED is displayed.

TYPE: The output depends on which types of syntax files were requested with the SYNTAX-FILE-TYPE operand:

Operand SYNTAX-FILE-TYPE=	**Output**
*ALL	List elements of all syntax file types
*SYSTEM	Only TYPE=*SYS
*GROUP(*ALL)	Only TYPE=*GROUP; for all profile IDs
*GROUP(<name>)	Only TYPE=*GROUP; for a specific profile ID
*SUBSYSTEM(*ALL)	Only TYPE=*SUBSYS, for all subsystems
*SUBSYSTEM(<name>)	Only TYPE=*SUBSYS; for a specific subsystem

VERSION: Version of the syntax file.

SYS-LOGOFF-PROC:
The list element only appears if the command SYSTEM-LOGOFF-PROC=*YES was specified (default value).

SYS-LOGOFF-INCL:

The list element only appears if the command SYSTEM-LOGOFF-INCL=*YES was specified (default value).

SYS-LOGON-PROC:

The list element only appears if the command SYSTEM-LOGON-PROC=*YES was specified (default value).

SYS-LOGON-INCL:

The list element only appears if the command SYSTEM-LOGON-INCL=*YES was specified (default value).

SHOW-SYNTAX-VERSIONS
Output syntax file versions

Domain: SDF

Privilege: STD-PROCESSING
HARDWARE-MAINTENANCE
SAT-FILE-EVALUATION
SAT-FILE-MANAGEMENT
SECURITY-ADMINISTRATION

The SHOW-SYNTAX-VERSIONS command returns information on the system, subsystem and group syntax files currently activated for the task. The output contains the names and versions of all software products and components that are currently being used. The versions of syntax files integrated in the basic syntax file are output at the start of the list. SHOW-SYNTAX-VERSIONS can also be used to request information on a specific software product or a list of software products and components.

The command supports structured output in S variables (see page 73 or the “Commands” manual, Volume 6 [4].)

SHOW-SYNTAX**-VERS**IONS
SOFTWARE-UNIT-NAME = ***ALL** / list-poss(2000): <structured-name 1..16>

SOFTWARE-UNIT-NAME =
Defines the scope of the output.

SOFTWARE-UNIT-NAME = *ALL / list-poss(2000): <structured-name 1..16>
Outputs the versions and names of all product-specific syntax files which are integrated in the system, subsystem and group syntax files currently activated for the task.

SOFTWARE-UNIT-NAME = list-poss(2000): <structured-name 1..16>
Shows the versions of the specified software products. If a specified product is integrated in a number of active syntax files, it will appear more than once in the output list.

Command return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Command executed normally
1	32	CMD0500	Syntax description in current syntax file invalid
	64	CMD0811	Command not executed successfully
			Guaranteed messages: CMD0300, CMD0500, CMD0811

Output in S variables

The output of the SHOW-SYNTAX-VERSIONS command can be redirected to an S variable, provided SDF-P is loaded on your system. This S variable must be declared as a list variable of type “structure” (see the DECLARE-VARIABLE command in the “SDF-P” [14] User Guide.

Output information	Name of the S variable	T	Contents
Path name of syntax file	var(*LIST).F-NAME	S	<filename 1..54>
Name of software component	var(*LIST).SW-UNIT(*LIST). COMPONENT(*LIST).NAME	S	<structured-name 1..13>
Version of software component	var(*LIST).SW-UNIT(*LIST). COMPONENT(*LIST).VERSION	S	<text 3..3> / <text 8..8>
Name of software unit	var(*LIST).SW-UNIT(*LIST).NAME	S	<structured-name 1..16>
Version of software unit	var(*LIST).SW-UNIT(*LIST).VERSION	S	<text 7..8>
Type of syntax file	var(*LIST).TYPE	S	*GROUP *SYS

Notes on the contents of variables

VERSION: Version of a software unit or component syntax file.

TYPE: No distinction is made between system and subsystem syntax files. Both are output with TYPE=*SYS.

5 SDF-I

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 Siemens Computers 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) under BS2000/OSD V2.0 and higher any more, 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.

5.1 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 Siemens Computers (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 [7]).

A new syntax file format was introduced with SDF V2.0. Earlier syntax files of any type (user, group or system) can be converted to the new format with the CONVERT-SYNTAX-FILE statement. CONVERT-SYNTAX-FILE can be executed independently of any OPEN statement.

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 79.

5.2 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> **,MON**JV = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIM**IT = ***JOB-RE**ST / <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]m.naso['] (see also “product-version” on page 14).

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.

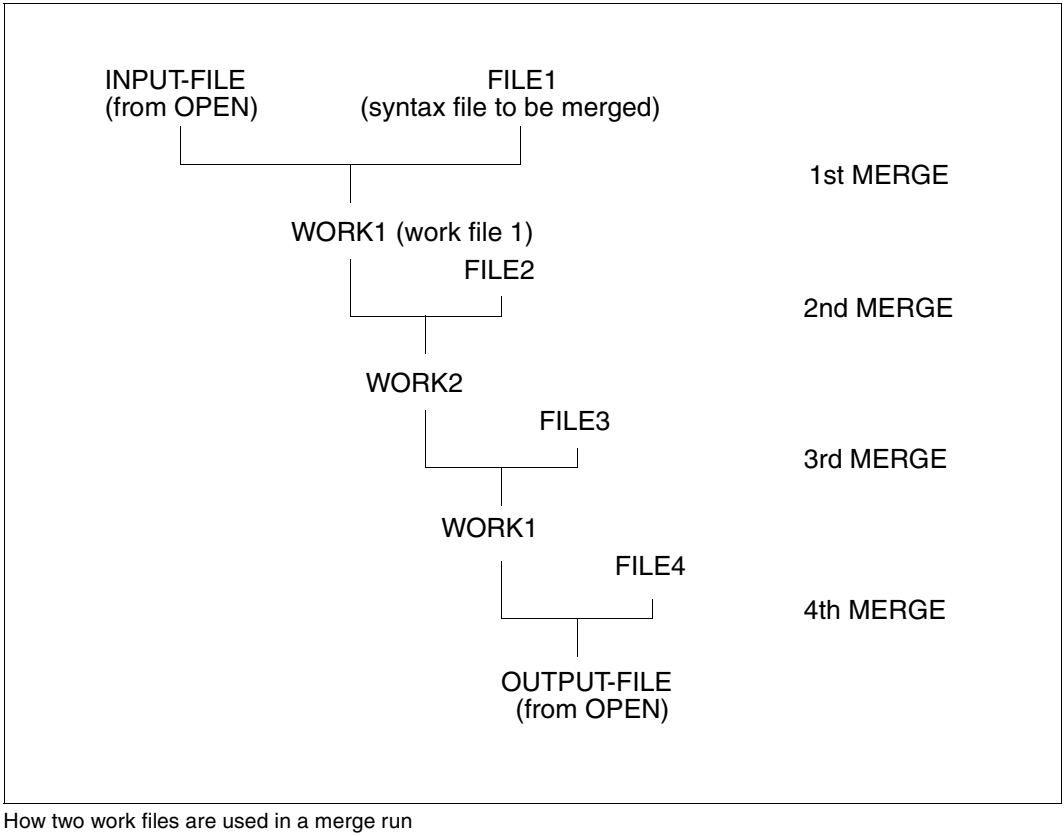
The OPEN statement must be used prior to each merge operation to assign the input syntax file and the output syntax file. MERGE statements are then used to specify the syntax files to be added. A separate MERGE statement is required for each syntax file to be incorporated.

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.

5.3 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 n=1 or 2, version: SDF-I version, e.g. 041 for SDF-I V4.1). Note that the name SDF-I.MERGE.WORK.tsn.n was used in earlier versions up to SDF-I V3.0. In general, users must make sure that they do not select names for their own files starting with the same name components as the work files. 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.

5.4 Handling SDF standard statements and domains

The SDF standard statements are contained in the SDF syntax file as of SDF V4.1. 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.

Starting with SDF-I V4.0A, 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.

5.5 Notes

Notes on 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

Notes on compatibility of syntax file formats:

- Starting with SDF V2.0, syntax files have a new format. SDF-I processes this new format as of Version 2.0. Older syntax files must be converted to the new format if they are to be used in SDF ≥ V2.0. It is, however, also possible to create output syntax files with the old structure (for SDF < V2.0A) using SDF-I.
- Syntax files that are generated with SDF-I V4.1A and the V4.1 format can no longer be processed with SDF-I versions < 4.1A, but can be used without problems in SDF as of Version 3.0.

- 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. This syntax file can then also be processed with SDF-I V3.0.
- As of SDF-I V3.0, the name of the product SDF-A is entered in the output syntax file as “SDF-A”. In older syntax file versions (for SDF ≤ V2.0), the name SDF-A was defined as “SDFA”. If such a syntax file is processed with SDF-I ≥ V3.0, 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 < V4.0 is an exception.

5.6 SDF-I statements

5.6.1 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

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.

5.6.2 Description of the statements

CONVERT-SYNTAX-FILE
Convert syntax file to new format

The CONVERT-SYNTAX-FILE statement can be used to convert any syntax file (group, system, user) with an old format (SDF < V2.0A) into the new format that is compatible with SDF V4.1 and higher.

CONVERT-**SYN**TAX-**FILE**
INPUT-**FILE** = <filename 1..54> ,**OUT**PUT-**FILE** = <filename 1..54>

INPUT-FILE = <filename 1..54>
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 earlier SDF-I versions, but may be used without difficulty in SDF as of Version 3.0.

END
Terminate SDF-I

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

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-PROD**UCT = ***NO** / <structured-name 1..15>
,**CHECK-VERS**ION = ***NO** / <c-string 1..7>
,**REPL**ACE-**PROD**UCT = **YES** / **NO**
,**UPD**ATE-**SDF-GLOB**ALS = **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 78).

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 Siemens Computers correspond to the SDF data type <product-version> (see also “product-version” on page 14).

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> ,**REPL**ACE-**PROD**UCT = **YES** / **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 78).

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 Siemens Computers, 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

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.
Furthermore, you can specify if the output file is to use the old syntax file structure (for SDF < V2.0A) or the new syntax file structure (for SDF ≥ V2.0A).

OPEN
INPUT-**FILE** = <filename 1..54>
,**OUT**PUT-**FILE** = <filename 1..54>
,**VERS**ION = **'UNDEFINED'** / <c-string 1..12> / <alphanum-name 1..12>
,**FORM**AT = ***STD** / ***OLD** / ***NEW**
,**BLKCTRL** = ***STD** / ***PAMKEY** / ***DATA**
,**WRITE-MODE** = ***NEW** / ***REPLACE**

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 Siemens Computers 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 93) shows the format entry for a syntax file as V1, V2, V3, V4 or V4.1. FORMAT=*OLD must be specified for syntax files with the V1 format; V2 to V4.1 correspond to FORMAT=*NEW.

FORMAT = *STD
Creates an old-format output syntax file (SDF < V2.0A) if all syntax files that were processed for it have the old structure. If syntax files with the new format are also present, *STD functions in the same way as FORMAT=*NEW.

FORMAT = *OLD
Creates an old-format output syntax file (for SDF < V2.0A). All syntax files processed for this output syntax file must then have this old format. Syntax files containing elements in new format will be rejected.

FORMAT = *NEW
Creates a new-format output syntax file (for SDF ≥ V2.0A). Any combination of old-format and new-format syntax files can be used to produce a new-format output syntax file. 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, ...).

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

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 Siemens Computers 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 78).

(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

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

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. As of BS2000/OSD-BC V1.0, the list also includes the privileges for commands (see example on page 95).

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 Siemens Computers 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 Siemens Computers 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 'V04.1A70' OF '1996-04-05' LOADED
%  BLS0552 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 1995. ALL RIGHTS
RESERVED
*show-syntax-file $.syssdf.bs2cp.140,information=cmd-interface ———————— (2)
% SYNTAX FILE : :T051:$TSOS.SYSSDF.BS2CP.140
% TYPE : SYSTEM    VERSION : SESD14.0A900    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  00000000000000000000000000000000001000000000000000000000000000000000
% ADD-DEVICE-DEPOT                NKCADDE  NKCADDE  ISL
% ALL  00000000000000000000000000000000001000000000000000000000000000000000
% ADD-FILE-LINK                   $DCOAFL  DCOADFL  ISL
% ALL  00000010001100000000000000000000001000000000000000000000000000000000
.
.
% SHOW-FILE                       SHFIL    DSHOWCM  ISL
% ALL  01000010001100000000000000000000001000000000000000000000000000000000 ⎫
% BA   00000000000000000000000000000000000000000000000000000000000000000000 ⎪
% BPA  00000000000000000000000000000000000000000000000000000000000000000000 ⎪
% DA   01000010001100000000000000000000001000000000000000000000000000000000 ⎬ (4)
% DPA  01000010001100000000000000000000001000000000000000000000000000000000 ⎪
% GA   01000010001100000000000000000000001000000000000000000000000000000000 ⎭
% CA   01000010001100000000000000000000001000000000000000000000000000000000
% SHOW-FILE-ATTRIBUTES            SHFAT    DCOFSDF  ISL
% ALL  01000010001100000000000000000000001000000000000000000000000000000000
% SHOW-FILE-LINK                  SHFILI   DCORTF2  ISL
% ALL  00000010001100000000000000000000001000000000000000000000000000000000
.
.
% WRITE-ACCOUNTING-RECORD         WRACRE   NACPWAC  ISL
% ALL  00000000001000000000000000000000001000000000000000000000000000000000
* ————————————————————————————————————————————————————————————————————— (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 V4.1 format, which means that it was already processed with SDF-I V4.1. Besides the formats V1 to V4.1, which correspond to the SDF-I versions V1.0 to V4.1, OLD or NEW can also be output. OLD corresponds to V1; NEW refers to all formats from V2 to V4.1.

(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” [7] 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.

6 SDF-U

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 [7]).

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-RE**ST / <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.naso['] (see also “product-version” on page 14).

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 94) are no longer defined in the SDF-U syntax file as of SDF-V4.1 and SDF-U V4.1, 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.

6.1 SDF-U statements

6.1.1 Overview of functions

SDF standard statements

A detailed description of the SDF standard statements can be found in the “Introductory Guide to the SDF Dialog Interface” [1].

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

6.1.2 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

OBJECT = ***DOM**AIN(...) / ***COM**MAND(...) / ***PROG**RAM(...)

***DOM**AIN(...)

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

***ALL**(...)

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

***COM**MAND(...)

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

***ALL**(...)

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

***PROG**RAM(...)

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

***ALL**(...)

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

,**FROM-F**ILE = <filename 1..54 without-gen-vers>

,**ATT**ACHED-**INF**O = **<u>*YES</u>** / ***NO** / ***ONLY**

,**OVERWR**ITE-**POSSIBLE** = **<u>*NO</u>** / ***EXTERN**AL-**ATTR**IBUTES / ***YES**

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

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

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-1 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-**VAL**UE

VALUE = ***UNCH**ANGED / list-poss(2000): <c-string 1..1800 with-low>(...)

<c-string 1..1800 with-low>(...)

OUTPUT = ***UNCH**ANGED / <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

OBJECT = ***DOM**AIN(...) / ***COM**MAND(...) / ***PROG**RAM(...)

***DOM**AIN(...)

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

***ALL**(...)

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

***COM**MAND(...)

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

***ALL**(...)

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

***PROG**RAM(...)

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

***ALL**(...)

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

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 = <u>*NONE</u> / <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 = ***UNCH**ANGED / <alphanum-name 1..12> / <c-string 1..12>
,**GUID**ANCE = ***UNCH**ANGED / ***STD** / ***MAX**IMUM / ***MED**IUM / ***MIN**IMUM / ***NO** / ***EXP**ERT
,**LOG**GING = ***UNCH**ANGED / ***STD** / ***INP**UT**-FORM** / ***ACCEP**TED**-FORM** / ***INVAR**IANT**-FORM**
,**PROC**EDURE**-DIALOG**UE = ***UNCH**ANGED / ***STD** / ***YES** / ***NO**
,**UTIL**ITY**-INTERF**ACE = ***UNCH**ANGED / ***STD** / ***OLD-MODE** / ***NEW-MODE**
,**CONT**INUATION = ***UNCH**ANGED / ***STD** / ***OLD-MODE** / ***NEW-MODE**
,**FUNC**TION**-KEYS** = ***UNCH**ANGED / ***STD** / ***OLD-MODE** / ***STYL**E**-GUI**DE**-MODE** / ***BY-TERM**INAL**-TYPE**
,**INP**UT**-HIST**ORY = ***UNCH**ANGED / ***STD** / ***ON** / ***OFF**
,**NUMB**ER**-OF-INP**UTS = ***UNCH**ANGED / ***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 [7]) 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 [7]) 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 "Introductory Guide to the SDF Dialog Interface" [1]. 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 Fujitsu Siemens 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 Fujitsu Siemens 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

OBJECT = ***GLOB**AL-**INF**ORMATION / ***DOM**AIN(...) / ***COM**MAND(...) / ***PROG**RAM(...)

***DOM**AIN(...)

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>

***COM**MAND(...)

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>

***PROG**RAM(...)

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>

,**ATT**ACHED-**INF**ORMATION = ***YES** / ***NO** / ***IMM**EDIATE

,**SIZ**E = ***MIN**IMUM / ***MAX**IMUM / ***MED**IUM

,**IMPL**EMENTATION-**INFO** = ***NO** (...)

***NO**(...)

FORM = ***UNGUID**ED / ***GUID**ED

,**LANGUAGE** = **E** / <name 1..1>

continued ➠

(part 2 of 2)

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

***UNLIM**ITED(...)

OUTPUT-FORM = ***STD** / ***FOR-INPUT**

,**OUTP**UT = ***SYSOUT** / ***SYSLST**(...)

***SYSLST**(...)

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

,**LINE-LEN**GTH = ***STD** / <integer 72..132>

,**PRIV**ILEGE = ***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 = *SYSOUT
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

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>
```

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.

7 SDF-PAR

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 chapter “Commands for SDF management” on page 41. 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 and BS2000 V9.5. 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-BC ≥ V2.0:

```
/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.

Notes on the installation of SDF-PAR can be found in section “Installation of SDF-PAR” on page 139 and in the release notice of the software product SDF-PAR.

7.1 SDF-PAR statements

7.1.1 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 “Introductory Guide to the SDF Dialog Interface” [1].

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

7.1.2 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 = ***UNCHA**NGED / ***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 = ***UNCHA**NGED / ***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. There are two possible formats:

- Format *V1:
  Parameter file format for SDF versions preceding V1.4A
- Format *V2:
  Parameter file format for SDF versions as of V1.4A.

MODIFY-VERSION
VERSION = ***UNCHA**NGED / ***V1** / ***V2**

VERSION = *UNCHANGED / *V1 / *V2
Defines the format of the parameter file.

VERSION = *V1
The parameter file is converted to *V1 format and can subsequently be used in SDF versions preceding V1.4A.
The record length of the parameter file is truncated to 54 bytes on conversion from *V2 format to *V1 format. The truncated information is of no consequence for SDF versions preceding V1.4A. A message to this effect is output for information purposes.
Every object of the parameter file is reconstructed and reinserted into the open parameter file, and a message is output for each such object.

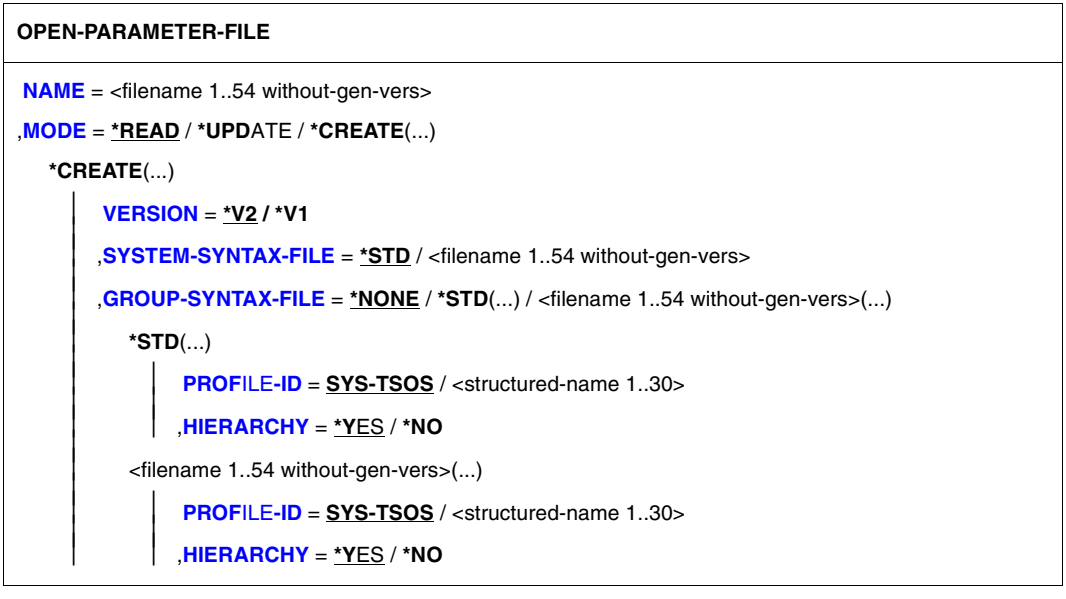
VERSION = *V2
The parameter file is converted to *V2 format and can then be used as of SDF V1.4A.
When a *V1 format is converted to *V2 format, the record length of the parameter file is extended to 67 bytes. The additional bytes are padded with X'00'. Records for group syntax file entries are extended 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.

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. Two formats are possible:

- Format *V1:
  Parameter file format for SDF versions preceding V1.4A
- Format *V2:
  Parameter file format for SDF versions as of V1.4A.

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.



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 *V2 format and can be used in SDF as of V1.4A.

VERSION = *V1
The parameter file is created in *V1 format and can be used in SDF versions preceding V1.4A.

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.

7.2 Installation of SDF-PAR

The main files supplied with SDF-PAR V1.1 are given below:

- SYSPRG.SDF-PAR.011
  The SDF-PAR program
- SYSMES.SDF-PAR.011
  Name of the message file for SDF-PAR as of BS2000/OSD-BC V2.0.
- SYSPRC.SDF-PAR.011.112
  Startup procedure that prepares the runtime environment for SDF-PAR and starts the SDF-PAR program (as of BS2000/OSD-BC V2.0)
- SYSSDF.SDF-PAR.011.USER
  User syntax file containing the SDF-PAR statements.
- SYSSII.SDF-PAR.011
  Installation information file for installing with IMON.

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 SDF-PAR can be used as of BS2000 V9.5 and SDF V1.4, no START-SDF-PAR command is included in the supplied software package. SDF-PAR is started with the START-PROGRAM command or with START-EXECUTABLE-PROGRAM (BLSSERV V2.3 and higher).

More detailed information on the installation of SDF-PAR and on specific hardware and software requirements can be found in the release notice for the SDF-PAR software product.

8 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.045 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 [12]) 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.

Related publications

Please apply to your local office for ordering the manuals.

[1] **SDF V4.5A** (BS2000/OSD)
Introductory Guide to the SDF Dialog Interface
User Guide

Target group
BS2000/OSD users
Contents
This manual describes the interactive input of commands and statements in SDF format. A Getting Started chapter with easy-to-understand examples and further comprehensive examples facilitates use of SDF. SDF syntax files are discussed.
Order number
U2339-J-Z125-8-76

[2] **BS2000/OSD-BC V5.0**
Introductory Guide to Systems Support
User Guide

Target group
This manual is addressed to BS2000/OSD systems support staff and operators.
Contents
The manual covers the following topics relating to the management and monitoring of the BS2000/OSD basic configuration: system initialization, parameter service, job and task control, memory/device/system time/user/file/pubset management, assignment of privileges, accounting and operator functions.
Order number
U2417-J-Z125-14-76

[3] **BS2000/OSD-BC V5.0**
Commands, Volumes 1 - 5
User Guide

Target group
This manual is addressed to nonprivileged users and systems support staff.
Contents
Volumes 1 through 5 contain the BS2000/OSD commands ADD-... to WRITE-... (basic configuration and selected products) with the functionality for all privileges. The command and operand functions are described in detail, supported by examples to aid understanding. An introductory overview provides information on all the commands described in Volumes 1 through 5.
The Appendix of Volume 1 includes information on command input, conditional job variable expressions, system files, job switches, and device and volume types.
The Appendix of Volumes 4 and 5 contains an overview of the output columns of the SHOW commands of the component NDM. The Appendix of Volume 5 contains additionally an overview of all START commands.
There is a comprehensive index covering all entries for Volumes 1 through 5.
Order numbers
U2338-J-Z125-15-76 Commands, Volume 1, A – C
U41074-J-Z125-2-76 Commands, Volume 2, D – MOD-JO
U21070-J-Z125-5-76 Commands, Volume 3, MOD-JV – R
U41075-J-Z125-2-76 Commands, Volume 4, S – SH-PRI
U23164-J-Z125-4-76 Commands, Volume 5, SH-PUB – Z

[4] **BS2000/OSD-BC V5.0**
Commands, Volume 6, Output in S Variables and SDF-P-BASYS
User Guide

Target group
This manual is addressed to programmers and users who write procedures.
Contents
Volume 6 contains tables of all S variables that are supplied with values by the SHOW commands in conjunction with structured output. Further chapters deal with:
– introduction to working with S variables
– SDF-P-BASYS V2.2A
Order number
U23165-J-Z125-4-76

[5] **BS2000/OSD-BC V5.0**
System Installation
User Guide

Target group
This manual is intended for BS2000/OSD system administration.
Contents
The manual describes the generation of the hardware configuration with UGEN and the following installation services: disk organization with MPVS, the installation of volumes using the SIR utility routine, and the IOCFCOPY subsystem.
Order number
U2505-J-Z125-15-76

[6] **IMON V2.5** (BS2000/OSD)
Installation Monitor
User Guide

Target group
This manual is intended for systems support staff of the BS2000/OSD operating system.
Contents
The manual describes the installation and administration of BS2000 software using the IMON installation monitor and its three components IMON-BAS, IMON-GPN and IMON-SIC. Installation (standard and customer-specific) using the component IMON-BAS for systems with BS2000-OSD V2.0 and as of BS2000-OSD V3.0/V4.0 is described in detail with the aid of examples in two separate chapters.
Order number
U21926-J-Z125-3-76

[7] **SDF-A V4.1E** (BS2000/OSD)
User Guide

Target group
This manual is intended for experienced BS2000 users and system administration staff.
Contents
It describes how to process syntax files and explains the SDF-A functions on the basis of examples. The SDF-A statements are listed in alphabetical order.
The manual also includes a description of the SDF-SIM utility routine.
Bestellnummer
U2284-J-Z125-9-76

[8] **SDF-A** (BS2000/OSD)
Ready Reference

Target group
This publication is intended for experienced BS2000 users with a good knowledge of SDF-A.
Contents
It contains the SDF-A statements in alphabetical order as well as the macros and function calls of the SDF program interface, the formats of the transfer area and the SDF-SIM statements.

[9] **BS2000/OSD-BC V5.0**
Executive Macros
User Guide

Target group
This manual is addressed to all BS2000/OSD assembly language programmers.
Contents
The manual contains a summary of all Executive macros:
– linking and loading
– virtual storage, memory pool, ESA
– task and program control
– ITC, serialization, eventing, DLM, contingencies, STXIT
– messages, accounting, JMS, TIAM, VTSU, ...
Detailed description of all macros in alphabetical order and with examples; general training section dealing with ITC, serialization, eventing, DLM, contingencies, STXIT, virtual storage, memory pool, ESA, ...
Order number
U3291-J-Z125-10-76

[10] **B2000/OSD-BC V5.0**
Utility Routines
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
The manual describes the utilities: DPAGE V14.0A, INIT V14.0A, JMP V2.0A, JMU V14.0A, LMSCONV V3.3B, PAMCONV V13.0A, PASSWORD V14.0A, PVSREN V1.4A, RMS V7.1A, SCDM V5.0A, SMPGEN V14.0A, SPCCNTRL V14.0A, TPCOMP2 V14.0A, VOLIN V14.0A.
Order number
U4303-J-Z125-7-76

[11] **MSGMAKER V1.2A** (BS2000/OSD)
Creating and Editing BS2000 Message Files
User Guide

Target group
This manual is addressed to systems support staff and nonprivileged users.
Contents
The manual describes the MSGMAKER utility routine, which is used to create and edit BS2000 message files. The description covers the operation (via masks and statements) of MSGMAKER and important operating sequences of the BS2000 operating system.
Order number
U23715-J-Z125-2-7600

[12] **FDDRL V14.0A** (BS2000/OSD V5.0)
User Guide

Target group
BS2000/OSD system administrators and operators
Contents
FDDRL physically saves and restores the contents of entire disks und pubsets. The manual describes the functions and statements of the program FDDRL for physical data saving in the computer center.
Order number
U3251-J-Z125-7-76

[13] **SECOS V4.0A** (BS2000/OSD
Security Control System
User Guide

Target group
– BS2000 system administrators
– BS2000 users working with extended access protection for files
Contents
Capabilities and application of the functional units:
– SRPM (System Resources and Privileges Management)
– SRPMSSO (Single Sign On)
– GUARDS (Generally Usable Access Control Administration System)
– GUARDDEF (Default Protection)
– GUARDCOO (Co-owner Protection)
– SAT (Security Audit Trail).
Order number
U5605-J-Z125-6-76

[14] **SDF-P V2.2A** (BS2000/OSD)
Programming in the Command Language
User Guide

Target group
This manual is addressed to BS2000 users and systems support staff.
Contents
SDF-P is a structured procedure language in BS2000. The manual begins with introductory chapters dealing with the basic principles of procedures and variables, and goes on to provide detailed descriptions of SDF-P commands, functions and macros.
Overview of contents:
– brief introduction to SDF-P
– procedure concept in SDF-P
– creating, testing, calling and controlling S procedures
– S variables, S variable streams, functions, expressions
– converting non-S procedures
– macros, predefined (built-in) functions, SDF-P commands
SDF-P V2.2A can only be used in conjunction with SDF-P-BASYS ≥ V2.1A, VAS ≥ V2.0A and SDF ≥ V4.1A.
Order number
U6442-J-Z125-5-76

[15] **XHCS V1.3** (BS2000/OSD)
8-Bit Code Processing in BS2000/OSD
User Guide

Target group
Users of the DCAM, TIAM and UTM access methods, system administrators, and users migrating from EHCS to XHCS.
Contents
XHCS (Extended Host Code Support) is a software package of BS2000/OSD that lets you use extended character sets in conjunction with 8-bit terminals. XHCS is also the central source of information on the coded character sets in BS2000/OSD. XHCS replaces EHCS.
Order number
U9232-J-Z135-4-76
Additional functions see README file for XHCS V1.4

[16] **DSSM V4.0/SSCM V2.3**
Subsystem Management in BS2000/OSD
User Guide

Target group
This manual addresses systems support staff and software consultants of BS2000/OSD.
Contents
The following are described: BS2000/OSD subsystem concept, dynamic subsystem management (DSSM) V4.0, subsystem catalog management (SSCM) V2.3 and the associated commands and statements.
DSSM supports the option of creating and managing user-specific subsystem configurations on a task-local basis.
Order number
U23166-J-Z125-3-76

[17] **BS2000/OSD**
Softbooks English
CD-ROM

Target group
BS2000/OSD users
Contents
The CD-ROM "BS2000/OSD SoftBooks English" contains almost all of the English manuals and README files for the BS2000 system software of the latest BS2000/OSD version and also of the previous versions, including the manuals listed here.
These Softbooks can also be found in the Internet on our manual server. You can browse in any of these manuals or download the entire manual.
Order number
U26175-J8-Z125-1-76
Internet address
http://manuals.fujitsu-siemens.com

Index

T

Contents

SDF V4.5A (BS2000/OSD)

SDF Management

Target group
This manual is intended for system administrators and experienced BS2000 users.
Contents
It describes how SDF is installed and administered using SDF commands and the SDF-I, SDF-U and the SDF-PAR utility routines. It includes a description of the SDF-I, SDF-U and the SDF-PAR statements.

Edition: March 2002

File: sdfv.pdf

This manual was produced by
cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Fujitsu Siemens computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00000

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on SDF V4.5A
SDF Management

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.