1 Preface

1.1 Brief product description

This manual contains utility routines for controlling and monitoring the BS2000 operating system.

The following routines are described:

Name of utility routine	Privilege required	Description starts on page
CONDMPPD	yes	29
DPAGE	for certain statements	41
INIT	yes	57
JMU	yes	105
LMSCONV	yes	141
PAMCONV	no	277
PASSWORD	no	363
PDPOOLS	yes	377
PVSREN	yes	401
RFUPD	yes	417
SODA	yes	443
SPCCNTRL	for certain statements	481
TPCOMP2	no	511
VOLIN	yes	523
SYSUPD	no	557

The utility routines handle two areas:

- volume processing
- file processing

The utility routines can be assigned either of these two areas as follows:

Utility routines for processing volumes

Processing of magnetic tapes/floppy disks

INIT	Initialization of tapes and floppy disks
TPCOMP2	Comparison of data on tapes

Processing of disks

CONDMPPD	Protection of private disks against write access
DPAGE	Output and modification of data in PAM pages
PDPOOLS	Generation and management of private disk pools
PVSREN	Conversion of pubsets
SPCCNTRL	Checking and management of disk space allocation
VOLIN	Initialization of disk storage units

Utility routines for processing files

JMU	Generation and maintenance of the file for stream and job class definitions
LMSCONV	Generation and management of libraries
PAMCONV	Conversion of file formats
PASSWORD	Encryption of passwords
RFUPD	Editing of REP files in BS2000 and generation of REP files on magnetic tape/disk
SODA	Evaluation of dump files generated by the dump generators SLED and SNAP
SYSUPD	Reading in of files stored on tape to public or private disk

1.2 Target group

This manual is intended both for privileged and nonprivileged BS2000 users.

1.3 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.BS2CP.112.E.** The user ID under which the README file is cataloged can be obtained from your system administrator. 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 FROM-FILE=SYSRME.BS2CP.112.E,LINE-SPACING=*BY-EBCDIC-CONTROL
```

1.4 Changes since the last version

Structural change

The manuals "Computer Center Utility Routines (BS2000/OSD-BC V1.0A)" and "Utility Routines (BS2000/OSD-BC V1.0A)" have been merged to make a single manual with the name "Utility Routines (BS2000/OSD-BC V2.0)". This contains utility routines intended for both privileged and nonprivileged users. The routines are arranged in alphabetical order. The utility routines MSGMAKER, MSGEDIT and MSGLIB as well as the chapter entitled "Message editing" are described in the manual "MSGMAKER" [18] starting with BS2000/OSD-BC V2.0. MSGEDIT and MSGLIB will be supported in BS2000/OSD-BC V2.0 for the last time.

Changes in the utility routines

DPAGE utility routine

As of BS2000/OSD-BC V2.0, the DPAGE utility routine is started by means of the system command START-DPAGE.

INIT utility routine

As of BS2000/OSD-BC V2.0, the INIT routine is started by means of the system command START-INIT.

The FORMAT operand in the INIT statement for magnetic tapes can be used to format a volume before writing new labels. This is possible only if permitted by the volume type.

JMU utility routine

Vector processors are no longer supported in BS2000/OSD-BC V2.0. The VECTOR-SPACE-LIMIT operand no longer exists. This affects the JMU statements DEFINE-JOB-CLASS, MODIFY-JOB-CLASS and SHOW-JOB-CLASS, in which this operand was used to specify, modify or output the vector page allocation.
The operand is no longer available in guided dialog. However, it is still accepted in procedure and batch mode to ensure that existing procedures remain executable.
Existing SJMSFILES also continue to be supported.

PAMCONV utility routine

As of BS2000/OSD-BC V2.0, the PAMCONV routine is started by means of the system command START-PAMCONV.

The description contains specifications for the required diagnostics documents if problems occur during a PAMCONV run.

PASSWORD utility routine

As of BS2000/OSD-BC V2.0, it will be possible to define LOGON passwords with up to 32 characters. In order to process such paswords with the PASSWORD routine, they must be shortened to 8 characters on the system level. The chargeable product SDF-P V2.0 provides the predefined function HASH-STRING(), which can be used for this conversion.

PDPOOLS utility routine

As of BS2000/OSD-BC V2.0, the PDPOOLS routine is started by means of the system command START-PDPOOLS.

SODA utility routine

SODA no longer evaluates the system files REPLOG (REP backup file) and VM2LOG (log file for the VM2000 program system).

The statement BASE ALL=Y is no longer used to output the DMS tables. With regard to SYS=Y, moreover, the address translation tables of the system/task and the PU-CRIA (concise rep information area of the program unit) are no longer extracted for diagnosis.

SPCCNTRL utility routine

As of BS2000/OSD-BC V2.0, the SPCCNTRL routine is started by means of the system command START-SPCCNTRL.

VOLIN utility routine

Support for new disk storage units and the statement OPTION=NO-WR-HA were already documented in the README file supplied with BS2000/OSD-BC V1.0.

VOLIN V11.2A supports the following functions:

- Repair function
  The tracks of a fully initialized and formatted CKD disk batch are checked and repaired. One or more tracks are selected by means of the DEFECTS statement and repaired with REPAIR=YES.
  The repair function supported by VOLIN V11.2A can only be used under the BS2000 operating system.
- DUALCOPY
  VOLIN V11.2A supports DUALCOPY [19]. This enables tracks to be repaired on one disk of the DUALCOPY disk pair while the other disk continues to be operated normally without interruption.
- D3492 disks
  VOLIN V11.2A supports D3492 disks with RAID5 architecture.

SYSUPD utility routine

As of BS2000/OSD-BC V2.0, the SYSUPD routine is started by means of the system command START-SYSUPD.

1.5 Metasyntax

1.5.1 SDF format

Statement format in the utility routines JMU, PAMCONV and PVSREN.

As of BS2000 V10.0A, the SDF command language replaces the ISP command language. ISP commands are still supported and remain compatible with the functions in BS2000 V10.0 but will not be further developed as functions. Only commands and statements with SDF syntax, however, can use all of the SDF functions. ISP commands are only checked for correct input of the command name.

Structure of the SDF command language

The syntax description of SDF commands and statements is contained in the syntax files:

– Commands begin with a slash; in interactive mode this is set by the system. SDF expects command input from the logical system file SYSCMD.
– SDF statements, i.e. statements intended for routines with an SDF interface, begin with two slashes (set by the system in interactive mode). SDF expects statement input from the logical system file SYSSTMT, which has the same assignment as the system file SYSDTA.

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

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

Command/statement names, operand names are contained as keywords in the syntax description. Keywords are specified as such. Variable operand values are described by means of data types (see Table 2 on page 15ff). The data type defines from which character set and according to which rules an operand value is formed. SDF checks whether the specification for a variable operand value lies within the range of permissible values as defined by the data type.

Keywords usually consist of a number of partial names connected by hyphens. As a rule, the names are taken from the ordinary English language environment and are chosen with a view to ensuring that analogous facts are described by means of identical keywords throughout the command set. Commands always start with a verb; this is followed by the object to which the activity is to be applied (e.g. MODIFY-FILE-ATTRIBUTES is used to modify the attributes of a file).

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

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

Every operand has a name and at least one possible value. Operands may be hierarchically subordinate to an operand value, in which case this operand value initiates a structure which contains all the lower-ranking operands. The operands of a structure are enclosed in parentheses. Further structures may be initiated within a structure; this feature is known as structure nesting.
Example of a structure:
SHOW-FILE-ATTRIBUTES SELECT=*BY-ATTRIBUTES(FILE-STRUCTURE=*ISAM)
The FILE-STRUCTURE operand defines a particular file attribute and is subordinate to the *BY-ATTRIBUTES operand value.

Concurrent specification of two or more operand values in the form of a list may be permitted. Operand values permitted as list elements are enumerated following ”list-poss(n):” (see SDF syntax representation, page 13), where “n” indicates the maximum number of list elements allowed.

Example of a list:
Format:
FILE-STRUCTURE = *ANY / list-poss(5): *PAM / *SAM / *ISAM / *BTAM / *NONE
Input:
SHOW-FILE-ATTRIBUTES SELECT=*BY-ATTRIBUTES(FILE-STRUCTURE=
(*SAM,*ISAM))
The FILE-STRUCTURE operand defines the file attributes SAM and ISAM.

Most operands are preset with a default value. This operand value is used if no explicit specification is made.

Conventions for input

The following points should be borne in mind when entering commands and statements:

- SDF interprets inputs in accordance with the standard code table **EBCDIC.DF.03** (e.g. uppercase/lowercase conversion).
  If XHCS[1] is available, it is possible to use extended character sets (CCS[2]). The coding of the dollar sign ($), number sign (#), commercial at (@),exclamation mark (!) and quotation mark (") in an extended character set must comply with the coding in the standard code table.
  SDF interprets additional characters of an extended character set only within the data types <c-string> and <text>, i.e. for uppercase/lowercase conversion a code table supplied by XHCS is used for that extended character set.
  SDF rejects all additional characters used within any of the other data types and reports a syntax error.

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

  – Input at the data display terminal:

    The CCS from the user entry is used for command input, provided 8-bit mode has been activated by means of the MODIFY-TERMINAL-OPTIONS command.
    For statement input, the CCS specified in the relevant macro call (RDSTMT, TRSTMT or CORSTMT) is used. If no CCS has been specified, the same applies as for command input.

  – Input from a file or a PLAM library member:

    For command input, the CCS of the input file is used. Alternatively, if SYSDTA was assigned by means of the ASSIGN-SYSDTA command, the CCS of the PLAM library member is used.
    For statement input, the CCS specified in the relevant macro call must be identical with the CCS for SYSDTA inputs. The statement will be rejected if the two coded character sets are not identical.
    During a procedure interrupt, the CCS selected for input at the data display terminal is used.

  – Input from an S variable:

    If SYSDTA is assigned to a compound S variable, the CCS selected for input at the data display termianl is used in interactive mode, while the CCS **EDF03IRV** is used in batch mode.

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

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

[2] Coded Character Set

- The first character of a command input is the slash (/); a statement is preceded by two slashes (//).
  In the event of input at the data display terminal, the slash or slashes appear automatically as a system prompt. In the event of input from a procedure file, the slash or slashes must be included in the input records.
- A label may be placed between slash and command name to identify the command line as a branch destination within procedures. At least one blank must separate the label from the command name. Labels have different formats for S and non-S procedures:
  - In *S procedures* the label consists of a name with up to 255 characters (corresponding to <structured-name 1..255>), followed by a colon. Labels in S format do not belong to the command. They are analyzed by SDF-P only.
  - In *non-S procedures* the label consists of a leading period and a name with up to 8 characters (corresponding to <name 1..8>).
- The command/statement name must be separated from the following operand(s) by at least one blank.
- Operands must be comma-separated.
- Operand values within a list must be comma-separated. If two or more list elements are specified, the enumeration must be enclosed in parentheses.
- Operands may be entered either as keyword operands or as positional operands. In keyword operands, an equals sign links operand name and operand value. In positional operands, only the operand value is specified, the assignment being determined via the relative position of the operand within the input stream.
- Further blanks between keywords, variable operand values, commas and equals signs are possible for documentation purposes and are ignored.
- Strings enclosed in quotes are interpreted as comments and ignored. Comments can be used in the same way as further blanks, but they are not permitted in front of labels.
- End-of-line comments:
  In S procedures the character string &* introduces an end-of-line comment, i.e. all subsequent characters up to the end of the line are ignored. Of particular importance in this respect is the fact that continuation characters, semicolons and & expressions lose their special meaning.
- Continuation lines:
  A hyphen as the last character of an input record (with any number of trailing blanks) is interpreted as a continuation character. The following input line thus becomes the continuation line of the preceding command or statement.
  If input is from the display terminal, the system issues “/” or “//” as a prompt requesting continuation of input. If commands/statements are entered from procedure or ENTER files, the continuation line must begin with “/” or “//”.

The continuation character must be written within the range of columns from 2 through 72 if CONTINUATION=*NEW-MODE has been set (see SHOW-SDF-OPTIONS or MODIFY-SDF-OPTIONS). If CONTINUATION=*OLD-MODE applies, the continuation character can only be in column 72.
Any characters following column 72 are ignored.
The total length of a command (including any continuation lines) can be up to 16364 bytes, inclcuding blanks and comments. For ISP commands which are still supported, the total length can be up to 4096 bytes.

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

In interactive mode (terminal input), the input record length and the position of any continuation character depends on the size of the input buffer of the display terminal; the maximum possible length is 16364 bytes.

In S procedures, the operand INPUT-FORMAT=*FREE-RECORD-LENGTH of the SET-PROCEDURE-OPTIONS command can be used to define a freely selectable length for an input record, provided this value does not exceed the maximum length of 16364 bytes. The continuation character may be input as of column 2.

- In the case of input from non-S procedure files or ENTER files, keywords must be written in uppercase form. This also applies to values of procedure parameters, S variables and job variables if they are to replace parts of commands or statements.
- Using a semicolon between two commands enables the input of two or more commands within the same input record (including continuation lines). This input option is only available in interactive mode and within *S procedures*.

SDF syntax representation

Figure 1 gives an example of the representation of the syntax of a command in a manual. The command format consists of a field with the command name. All operands with their legal values are then listed. Operand values which introduce structures and the operands dependent on these operands are listed separately.

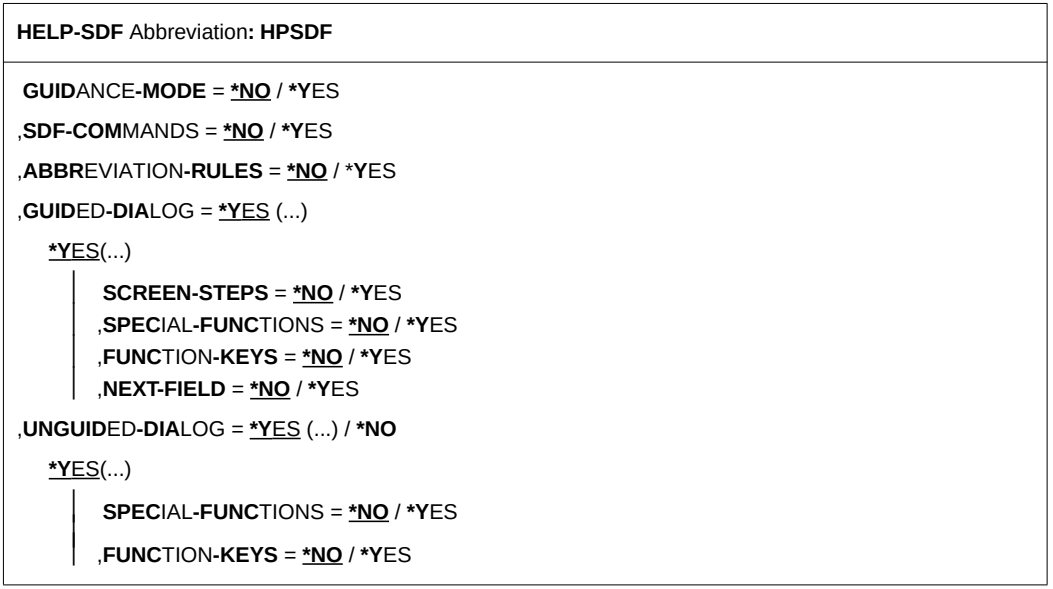


Figure 1: Representation of the syntax of the user command HELP-SDF

This syntax description is valid for SDF V4.0A.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 this table.

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 this table.

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 can be used to extend or limit the set of values. The following short forms are used in this manual for data type suffixes:

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

The description of the ‘integer’ data type in this table 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.

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, operand names, key word values) and constant operand values. The keywords 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 = <full-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 (Section 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**(...) / <full-filename 1..54>
Abbreviation:	The name that follows represents a guaranteed alias for the command or statement name.	**HELP-SDF** Abbreviation: **HPSDF**

Table 1: Metasyntax (Section 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 full-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 interactive prompting, 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 (Section 1 of 6)

Data type	Character set	Special rules
full-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 max. 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 (Section 2 of 6)

Data type	Character set	Special rules

full-filename (continued)		#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 (Section 3 of 6)

Data type	Character set	Special rules
partial-filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.][partname.] :cat: see full-filename $user. see full-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 full-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. 1024 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 (Section 4 of 6)

Data type	Character set	Special rules
product-version	A…Z 0…9 period single quote	Input format: [[C]'][V][n]n.nann['] correction status release status where n is a digit and a is a letter. The release and correction status must be specified if product-version does not include a suffix (see suffix without-corr and without-man 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 $, # or @.

Table 2: Data types (Section 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 may not be prefixed. There may be an odd number of characters.

Table 2: Data types (Section 6 of 6)

Suffixes for data types

Suffix	Meaning
x..y *unit*	a) 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* b) with the other data types: length specification 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.
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.
-under	Permits underscores “_” for the data type “name”.

Table 3: Data type suffixes (Section 1 of 6)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with (contd.)
-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, wildcards from the UNIX world (referred to below as POSIX wildcards) are now accepted in addition to the usual BS2000 wildcards. However, 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 les between s_x and s_y in the alphabetic collating sequence; numbers are sorted after letters (A...Z0...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 (Section 2 of 6)

Suffix	Meaning	
with-wild(n)		
(continued)	-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 full-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 the 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.
	$[c_x\text{-}c_y]$	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 $[c_x\text{-}c_y]$ and [s] can be combined into $[s_1c_1\text{-}c_2s_2]$
	$[!c_x\text{-}c_y]$	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 $[!c_x\text{-}c_y]$ and [!s] can be combined into $[!s_1c_1\text{-}c_2s_2]$
	[!s]	Replaces exactly one character not contained in string s. The expressions [!s] and $[!c_x\text{-}c_y]$ can be combined into $[!s_1c_1\text{-}c_2s_2]$

Table 3: Data type suffixes (Section 3 of 6)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with (contd.)
-constr</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.
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 (Section 4 of 6)

Suffix	Meaning
with-constr (continued)	The following rules must be observed when specifying a constructor: – The constructor must include at least one wildcard of the selector. – If the number of identical wildcards exceeds those in the selector, the index notation must be used. – 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 the wildcard “*” is to be duplicated. For example: “<n><n>” must be specified instead of “**”. – 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 selector ends with a period, the constructor must also end with a period (and vice versa).

Table 3: Data type suffixes (Section 5 of 6)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with-constr
(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][n]n.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][n]n.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>-user</td><td>Specification of a user ID is not permitted.</td></tr>
<tr><td>-vers</td><td>Specification of the version (see “file(no)”) is not permitted for tape files.</td></tr>
</table>

Table 3: Data type suffixes (Section 6 of 6)

1.5.2 ISP format

Statement format in the utility routines CONDMPPD, DPAGE, INIT, LMSCONV, PASSWORD, PDPOOLS, RFUPD, SODA, SPCCNTRL, TPCOMP2,VOLIN and SYSUPD.

Metacharacters are used in the representation of the statement formats and declarations made which are described briefly below.

WRPASS	Uppercase characters denote constants which must be input exactly as shown.
programname	Lowercase characters denote variables which are replaced with current values on input.
YES NO	An underscored word means that this is a default value which is automatically used if no specification is made.
{ YES NO }	Braces are used to indicate alternatives, i.e. a specification must be selected from the values shown in the brackets. The alternatives are written one below another.
{YES|NO}	A vertical line between two adjacent specifications in braces also means that these are alternatives from which the user must select one value.
[ ]	Square brackets indicate optional specifications that can be omitted, in which case the entire specification in the brackets must be omitted
( )	Parentheses are part of operands and must be input with them.
␣	Means a space (blank).
D2,ALL	Commas separating operands must also be entered.
(filename1),... (vsn1, vsn2, ...)	3 periods mean that the unit in front of the comma can be repeated (possibly only up to a specified maximum value).

Table 4: ISP metasyntax

Format representation

Operation	Operands
INCLUDE	module

means that at least one space must be entered between the constants listed under "Operation" and the specifications listed under "Operands". Where this is not the case, another representation format is used which shows clearly that there are no spaces.

2 CONDMPPD
Protecting private disks against write access

Version: **CONDMPPD V11.2A**

In BS2000 it is not possible to protect a private disk against write access by setting a hardware write-protect lock. For example, even straightforward OPEN INPUT processing involves write operations, because statistical information is written into the F1 label every time an open operation is performed.

PPD (Protected Private Disk) is a function that enables individual private disks to be protected against write access in BS2000 and make them available only for read access. This is implemented by means of a VSN list maintained in virtual memory: whenever a user wishes to write to a private disk, this list is referenced in order to ascertain whether write authorization is available for this disk. If not, attempts to write to the disk are rejected with an error message, while system F1 label operations are simply suppressed. In the latter case, this means that statistical information is not updated and is therefore only of limited use.

The list can be dynamically updated in interactive mode from either the terminal or the console; at present it can accommodate up to 576 entries. Write locks can be set or canceled for each disk individually. In contrast to previous versions, it is no longer possible to dispense entirely with a check of the VSN list.
The PPD function is activated by means of the CONDMPPD utility routine (**CON**nect **D**ata **M**anagement system **P**rotected **P**rivate **D**isk).

Range of functions

PPD is loaded into class 4 memory as a selectable unit; the connections to the BS2000 system modules affected by PPD are set up automatically.

The following files are required to load PPD:

CONDMPPD	Loadable program phase that establishes the connection to PPD.
SYSLNK.PPD.112	Module library containing the PPD modules to be loaded.
SYSREP.PPD.112	REP file containing corrections carried out after the release of PPD.
SYSSSD.PPD.112	Subsystem declarations
SYSMSV.PPD.112	Message file

Key

112=version number

The CONDMPPD utility routine is called as follows:

```
/START-PROGRAM FROM-FILE=CONDMPPD
```

The operands `CPU-LIMIT`, `TEST-OPTIONS`, `MONJV`, `RESIDENT-PAGES` and `VIRTUAL-PAGES` of the `START-PROGRAM` command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the `START-PROGRAM` command in the "Commands, Volume 3" [3] manual.

The PPD function responds to the first program call with the message:

```
%  BLS0500 PROGRAM 'CONDMPPD', VERSION 'V11.2A' OF '1994-11-17' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
%  PPD0002 PPD VERSION 11.2A00 LOADED
%  PPD0003 ENTER 'HELP' FOR INFORMATION
%  PPD0004 'DMPPD' INPUT EXPECTED. RESPONSE (<STATEMENT>=INPUT)
*
```

After each subsequent call it responds with the message:

```
PPD0002 PPD VERSION 11.2A00 ALREADY CONNECTED WITH SYSTEM
```

The statements described below are then available for use. In addition, all the BS2000 commands permitted for MCLP, with the exception of CALL-PROCEDURE, LOAD-PROGRAM, START-PROGRAM, RESTART-PROGRAM and HELP-SDF, can be executed in interactive mode.

2.1 Statements

2.1.1 Overview of all CONDMPPD statements

Statement	Meaning
ADD	Add a volume serial number to the VSN list
CONSOLE	Modify the routing code and switch the dialog to the console
DELETE	Delete a volume serial number from the VSN list
END	Terminate the program
HELP	Output all statements with brief explanatory notes
LIST	Output an overview of the whole VSN list or the status of a single VSN
OPTION	Modify the status of a VSN
SYNOPSIS	Output the number of accesses made to the VSN list

The BS2000/OSD-BC V2.0 version of PPD (=PPD V11.2A) can be loaded only under Version 11.2A of CONDMPPD or later. Conversely, CONDMPPD V11.2A can run only under BS2000/OSD-BC V2.0.

2.1.2 Description of the statements

ADD
Add volume serial number to VSN list

The ADD statement enables a VSN to be added to the VSN list. If the VSN already appears in the VSN list and is in use (by NDM) (i.e. the $AQUIR macro has been executed or the system's catalog ID is in the SVL, e.g. following the command /SET-DISK-PARAMETER ASSIGN-TIME=*OPERATOR or /ADD-FILE-LINK LINK-NAME=linkname, FILE-NAME=file on the disk, or the macro OPEN file) and the privilege is to be switched from YES to NO, the following query appears:

```
PPD0017    VOLUME WITH VSN '(&00)' ALLOCATED WITH WRITE PRIVILEGE. CHANGE
           OF PRIVILEGE? REPLY (Y=YES; N=NO)
```

A response of Y causes ADD to be executed. If there are still write accesses to the disk, or to a file with an extent on the disk, these accesses are rejected. If a file with an extent on the disk is still open, it will no longer be possibly to close it normally. In order to restore the file, the command REPAIR-DISK-FILES FILE-STATUS=*ANY(FILE-NAME=file) must be given from a system in which writing is permitted. The catalog entry can then be imported into the system without write permission.

For files where BLKCTRL=NO applies, the last-page pointer (LPP) cannot be restored, however.

In batch mode, the ADD statement is rejected under the above conditions.

Operation	Operands
{ A | ADD }	vsn[,PRV={ Y[ES] | N[O] }]

vsn — The specified volume serial number is entered in the VSN list. Already existing entries are overwritten, i.e. the identifier "S" (cf. the LIST statement) can be deleted.

PRV — Specifies the required VSN status:
YES: write access permitted (status display W)
NO: write access prohibited (status display R)
The default value is NO.

CONSOLE
Switch dialog to console

The CONSOLE statement serves to modify the routing code and to switch the dialog to the console. However, the latter function is possible only if CONDMPPD was called in batch mode.

Operation	Operands
{ C CON CONSOLE }	[rc]

rc — Specifies the routing code to be modified. If "rc" is not specified, input/output is switched back to the console.

The main console is assigned the default routing code '*'.

DELETE
Delete volume serial number from VSN list

The DELETE statement enables volume serial numbers which do not have the status specification "S" (cf. the LIST and ADD statements) to be deleted from the VSN list. The DELETE statement can only be issued for individual VSNs. The statement may be rejected if the disk is currently being used (cf. "Rejecting statements" on page 38). In this case it must be reissued later.

Operation	Operands
{ D DEL DELETE }	vsn

vsn — The specified volume serial number is deleted from the VSN list unless it includes the status specification "S".

END
Terminate program

The END statement terminates CONDMPPD, i.e. the connection to PPD is cleared down. However, PPD itself remains active.

Operation	Operands
{ E END }	

HELP
Output all statements with brief explanations

The HELP statement serves to output brief explanations to any given program statement.

Operation	Operands
{ H HLP HELP }	

LIST
Output overview of whole VSN list or status of one single VSN

The LIST statement outputs an overview of the entire VSN list or displays the status of an individual volume serial number.

Operation	Operands
{ L LST LIST }	[vsn]

vsn — The status of the volume serial number specified by "vsn" is to be output. The following outputs are possible:
W: write access permitted (WRITE)
R: write access prohibited (READ)

If the VSN status is accompanied by the specification "S", this indicates that the entry was made by the system itself and not by system administration (see "Access from several processors in BS2000" below). A VSN entry with "S" cannot be removed by means of a DELETE statement. If no volume serial number is specified, the entire VSN list is output.

OPTION
Modify VSN status

The OPTION statement enables the status of a volume serial number to be modified (unless "S" is also specified). Any attempt to change the status of a VSN from "privileged" to N while the disk is in use causes the CONDMPPD caller to be asked to confirm that the privilege is really to be changed (see the ADD statement).

Operation	Operands
{ O | OPT | OPTION }	{ vsn,PRV={ Y[ES] | N[O] | R } | PRV=R[ESET] }

vsn

The specification of a VSN is mandatory with PRV=Y/N. The status modification refers to the specified VSN. If a VSN is specified together with PRV=R, the latter entry is disabled (as if PRV=N had been specified).

PRV

Describes the desired status modification as follows:

	YES	NO	RESET
VSN specified	Write access permitted	Write access prohibited	Write access prohibited
VSN not specified	Not permitted	Not permitted	Reset error flag

SYNOPSIS
Display number of VSN list accesses

In order to facilitate performance analysis, the number of VSN list accesses is counted. The counters are 4 bytes in size and can be output by means of the SYNOPSIS statement. In the event of a counter overflow, a message to this effect is displayed; the overflow can be reset manually.

Operation	Operands
{ S SYN SYNOPSIS }	[R[ESET]]

RESET — This operand specifies that all counter overflows are to be reset. If no entry is made, the current status of all counters is displayed.

2.2 Access from several processors in BS2000

Access to a common disk pool from a number of BS2000 processors is supported by the SPD function (Shareable Private Disk). However, this type of access is also possible by means of PPD, provided PPD is loaded on all of the processors in question. Unlike SPD, however, PPD can be used solely to prevent the same disk being written to by several processors at the same time. PPD does not synchronize access to the same file. For example, if an ISAM file is updated by processor A and the same file is simultaneously opened for reading by processor B, the processing job on processor B may be terminated with errors as the index structures saved in this task no longer correspond to the existing, updated structures.

If PPD is used for accessing the same disk pool from different processors, the VSN list on each of these processors must be managed with extreme care. If write access is permitted for a given VSN on one processor, it must be prohibited for this VSN on the other processors. Otherwise "double allocations" may occur, i.e. several files may occupy the same space on the disk, or catalog entries and files on the disk may be destroyed.

This increases the system administration overhead for managing the VSN lists on the various processors. To reduce this workload PPD checks disks currently mounted on a shareable device the first time they are accessed, so as to ascertain whether the VSN in question has been entered in the VSN list. If not, the system enters this VSN in the VSN list with the identifier "RS" (= read-only access) (automatic volume transfer).

A VSN marked as "RS" is automatically removed from the VSN list after it has been released by the last user.

The volume serial numbers entered explicitly by means of the ADD statement are not affected by this process.

If '*' is specified in the DVC statement at system generation, it indicates that in addition to the connection defined for a particular device, there is also a link from the device to a second CPU (multiple connection). It is possible to define a device as an SPD device by specifying '**' in the DVC statement. This means that any private disk mounted on the device is to be "system-shareable" by default. The disk can be reserved by up to four systems at the same time (see also the manuals "System Installation" [15] and "Introductory Guide to Systems Support" [14]). PPD excludes such SPD devices from automatic volume transfer. However, as the VSN list is checked without restrictions, it is possible to declare write protection for private disks on SPD devices also.

As OPEN=INOUT is the default value for many programs, PPD permits this open mode. Whenever a given file is processed it checks every action macro for write authorization; if it encounters "write access prohibited", the macro is rejected.

2.3 Rejecting statements

OPTION, ADD and DELETE statements should not be issued for a VSN if the disk is being used, as they will be rejected if:

– the disk in question is being used by a user and execution of the statement would lead to another USAGE mode or access protection (enter the NDM command SHOW-DISK-STATUS in order to display the output fields "ACCESS" and "SVL-ALLOC")

– the disk is occupied for OPTION, ADD in a procedure in batch mode and execution of the statement would lead to the withdrawal of an explicitly granted PPD write privilege.

This means that write protection can be defined only before opening a file on the appropriate private disk or before the preceding ADD-FILE-LINK command.

2.4 Files on two or more disks

PPD also supports files with extents on two or more disks.

However, there exists a restriction with regard to commands used to generate file catalog entries or modify file attributes; namely that volume serial numbers specified in the second or a later position in these commands cannot be assigned to (possibly shareable) disk drives, with the result that a subsequent write protection may no longer be effective.

Volume serial numbers explicitly protected by means of ADD PRV=N are checked without exception. Consequently it is advisable to dispense with the dynamic volume transfer and always to specify the disks to be protected explicitly with ADD PRV=Y. For disks protected in this way it is irrelevant whether the drive in use was generated in the DVC statement by entering one (*) or two (**) asterisks.

If a file has extents on two or more private disks, then either all these disks should have PPD write protection, or none of them.

```
/MODIFY-FILE-ATTRIBUTES SUPPORT=*PRIVATE-DISK(SPACE=
*RELATIVE(PRIMARY-ALLOCATION=< >))...,
OPEN OUTPUT, PAM, GETK etc.
```

are rejected even if only a single file extent (and even if it is empty) resides on a protected disk. The sole exception is the /DELETE-FILE statement, where empty extents are not checked. If the catalog entry contains an empty extent on a write-protected volume, the file must be copied before it can be made available for write access.

2.5 Private disks with the same VSN

PPD does not support the simultaneous operation of a number of private disks with the same VSN, since system behavior would be unpredictable. If explicit write protection is requested for one VSN, it will apply to all private disks with this VSN.

2.6 Messages

The messages of the CONDMPPD utility routine belong to message class PPD. You will find the messages in "System Messages, Volume 2" [17].

PPD

PPD generates DMS error codes whenever attempts are made to write to protected disks. Before the job is aborted with the appropriate error code, the user always receives the message:

```
DMS0881 VSN vvvvvv PROTECTED BY PPD
```

The individual error codes supported by PPD are as follows:

ISAM:	0AA3
PAM:	0995
SAM:	0BB6
CMS:	033A
ALLOCATOR:	0443
OPEN:	0DBB
CATALOG:	0558
DELETE-FILE	0558
MOD-FILE-ATTR	0558
MOD-FILE-ATTR SUP=ANY-DISK(SPACE=...)	0543

Explanation

MODIFY-FILE-ATTRIBUTES usually returns ’0558’, but its return code is ’0543’ if SPACE is modified.

If a VSN cannot be automatically transferred to the VSN list because there is a catalog overflow, the following message is displayed:

```
PPD0882 DMPPD: VSN LIST OVERFLOW. PRIVATE DISK WITH VSN vvvvvv IS NOT
PROTECTED
```

This message appears only at the console and serves to inform the operator that the disk "vvvvvv" currently mounted on a shareable device is not protected at present, and that manual intervention is necessary.

2.7 Suppressing volume transfer

Automatic volume transfer to the VSN list by the system can be suppressed by means of an optional REP record. Disks on shareable devices are then handled in the same way as when the PPD function is not activated.

3 DPAGE
Output and modification of disk files

Version: **DPAGE V11.2A**

The DPAGE (display page) utility routine enables the user, as well as the system administration staff and the systems programmer, to perform the following functions:

- Output files in PAM format to the SYSOUT file (i.e. the terminal in an interactive task, and the printer in a batch task).
- Write files in PAM format to the SYSLST file (for high-volume printing).
- Modify data contained in a PAM page (2048 bytes) or in the PAM key (16 bytes).
- Only system administration is permitted to process volumes.

3.1 Support for multiple public volume sets (MPVS)

In a system with several pubsets, each pubset has its own TSOSCAT file catalog. Each of these catalogs is uniquely identified by its catalog ID (catid), which is part of the file name.

A file name has the following format:

:catid:$userid.filename

catid	Catalog identifier. This has a length of 1-4 characters and must be enclosed in colons.
userid	User identification under which the file is cataloged. It must be preceded by a $ sign, has a maximum length of 8 characters and must be concluded with a period.
filename	Name of the file as entered in TSOSCAT. The file name component (including all periods designating partial qualification) must not exceed 40 characters.

A fully qualified file name, consisting of catalog ID, user ID and file name, may have a total length of up to 54 characters:

1-4 characters	Catalog ID
2 characters	Colons as delimiters before and after the catalog ID
1 characters	$ signifying the beginning of the user ID
8 characters	User ID
1 characters	Delimiter between user ID and file name
38-40 characters	(Depending on the length of the catalog ID:) file name including all periods for partial qualification

3.2 Starting the program run

The DPAGE routine can be started in two ways:

```
/START-DPAGE
```

START-DPAGE

VERSION = **_*STD_** / <product-version 6..10> / <product-version 4..8 without-corr> /
<product-version without-man>

,**MONJV** = **_*NONE_** / <full-filename without-gen-vers>

,**CPU-LIM**IT = **_*JOB-REST_** / <integer 1..32767>

The VERSION, MONJV and CPU-LIMIT operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in "Commands, Volume 3" [3].

```
/START-PROGRAM FROM-FILE=$DPAGE
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in "Commands, Volume 3" [3].

3.3 Statements

3.3.1 Overview of the DPAGE statements

Operation	Operands	Meaning
{ /[bs-cmd] | BKPT }		Interrupt DPAGE / enter a BS2000 command
{ DISPLAY | D }	{ page | page1-page2 | page-$ | * } [,{ byte | byte1-byte2 | K }]	Output a specific portion of one or more PAM pages to SYSOUT
{ EDT | @ }		Call the file editor EDT as a subroutine
{ HALT | H | END | E }		Terminate the DPAGE routine
{ MODIFY | M }	[{ byte1 | Kn1 }] [,{ X‘hex-string‘ | nnX‘hex-string‘ | ‘character-string‘ | nn‘character-string‘ | C‘character-string‘ | nnC‘character-string‘ }] [,{ byte2 | Kn2 }]	Change the contents of a page (or the key) in an internal work area

OPEN O	filename[,INOUT INPUT] 'vsn',devtype[,S[HAREABLE] E[XCLUSIVE]]	Open a file Open a volume
PRINT P	page page1-page2 page-$ * [,byte byte1-byte2 K]	Output a specific portion of one or more PAM pages to SYSLST
READ R	page	Read a page into an internal work area
WRITE W		Write the page currently in the internal work area back to the file

Formats

The following terms are used in the description of the statements:

page	Decimal number of up to 7 digits
byte,byte1,byte2	Decimal number of up to 4 digits in the range 1-2048 *Exception* After opening 4K-formatted volumes, a decimal number in the range 1-4096.
Kn	The letter K followed by an integer in the range 1-16

After opening a volume, the meaning of the page number to be specified in the statement varies depending on the type of formatting:

– For files and 2K-formatted volumes, the page number refers to the physical half-page number (PHP=2048 bytes).

– For 4K-formatted volumes, the page number refers to the logical 4K block (contains 2 physical half-pages, i.e. 4096 bytes).

3.3.2 Description of the statements

BKPT or system statement

The BKPT statement, or alternatively a slash, can be used to interrupt DPAGE (i.e. to set a breakpoint). Control is returned to DPAGE via the BS2000 command RESUME-PROGRAM.

"/" may be followed by any BS2000 command.

Operation	Operands
{ /[bs-cmd] | BKPT }	

bs-cmd BS2000 command.

DISPLAY statement

The DISPLAY statement outputs one or more pages, or parts thereof, to the SYSOUT file.

Operation	**Operands**
{ DISPLAY | D }	{ page | page1-page2 | page-$ | * } [,{ byte | byte1-byte2 | K }]

Page range:

page Specifies the page of the file (or the volume) to be displayed.

page1-page2 Specifies the range of pages which are to be displayed.

Note

page1 < page2

page-$	Specifies that all pages, starting at "page" and continuing through to the end of the file (or volume), are to be displayed.
*	Specifies that the page currently in the internal work area is to be displayed.

Byte range:

byte	Specifies the byte to be displayed.
byte1-byte2	Specifies the byte range to be displayed. *Note* byte1 < byte2 Up to 3048 bytes can be used for byte2. For 4K-formatted volumes up to 4096 bytes.
K	Only the PAM key is to be displayed.

Example

DISPLAY 1,1-224

```
*OPEN TEST
OPEN COMPLETED
*DISPLAY 1,1-224
PAGE:0000001                PAMKEY:  57739BDE  01000001  00000138  00010006


001 --> (0001)  7CD7C602  02F8F5C1  D7C1D4C5  C4C9E340   @PF  90ATEST
011 --> (0017)  000057E8  000057E8  F8F7F0F2  F2F50001      Y   Y870225
021 --> (0033)  00000000  00000006  00000001  00000000
031 --> (0049)  00010000  00000000  00000000  00000000
041 --> (0065)  00D4010D  00000000  0000000C  027CD7C6    M           @PF
051 --> (0081)  E5F9F04B  C1F0F040  40400000  00000000   V90.A00
061 --> (0097)  00000000  00000000  D7C1D4C5  C4C9E340           TEST
=071 --> (0113)  40404040  40404040  40404040  40404040
091 --> (0145)  40000000  00000000  00000000  00000000
0A1 --> (0161)  00000000  00000000  40404040  40404040
0B1 --> (0177)  40404040  F1F9F8F7  00000000  00000000       1987
0C1 --> (0193)  00000000  00000000  00000000  E5F2F100              V21
0D1 --> (0209)  00000000  00000000  6CD9D6D6  E3404040           %ROOT
```

EDT statement

The EDT statement calls the file editor EDT in full-screen mode. Control is returned to DPAGE via H[ALT] or by hitting the K1 key. If @E was used to switch to line mode, return to DPAGE is alternatively possible via @RETURN or @HALT. The data in the virtual memory of EDT is released in this process.

EDT can take into account pages that, for example, were modified by means of the MODIFY statement or written to a SYSLST file by means of the PRINT statement.

Operation	Operands
{ EDT @ }	

END/HALT statement

This statement terminates DPAGE and closes the open file.

Operation	Operands
{ HALT H END E }	

MODIFY statement

This statement modifies the contents of the page currently in the internal work area (see READ). The page in the file (volume) is not affected (see WRITE).

Operation	Operands
{ MODIFY | M }	[- { byte1 | Kn1 }] [, { X‘hex-string‘ | nnX‘hex-string‘ | ‘character-string‘ | nn‘character-string‘ | C‘character-string‘ | nnC‘character-string‘ }][, { byte2 | Kn2 }]

First operand:

byte1 — Specifies the location in the PAM page (1-2048) where the text specified in the second operand is to replace the original text. All bytes before byte 1 remain unchanged.

Kn1 — Specifies the position in the PAM key (1-16) where the text specified in the second operand is to replace the original text. All bytes before Kn1 remain unchanged.

The default value is 1. The text specified in the second operand changes the original contents of the page, starting with byte 1.

Second operand:

{ X‘hex-string‘ \| nnX‘char.-string‘ \| ‘char.-string‘ \| nn‘char.-string‘ \| C‘char.-string‘ \| nnC‘char.-string‘ }

Specifies the text that is to replace the original text.
If byte 2 is not specified, the new text replaces the old text for the length of the new text, starting with byte 1. "nn" is an integer which specifies a repetition factor for the specified string.

The default value is 0, i.e. no text is changed.

The text must not extend beyond the PAM page or PAM key.

Third operand:

byte2	Specifies the location in the original PAM page where the text which is to be added to the new text begins, regardless of whether the new text is shorter, longer or equally long.
Kn2	Specifies the location in the original PAM key where the text which is to be added to the new text begins.

If the first operand is within the page (1-2048), the second operand must also be within the page.

Modifications in the first 8 bytes of the PAM key will not be transcribed back to the file.

Example

The original page is assumed to contain:

```
'ABCCATDOGXX...'
```

and the original key is assumed to contain:

```
'1234DEFGO...'
```

Example a

```
MODIFY 5, 'OW', 7
```

changes the entire page to:

```
'ABCCOWDOGXX....'
```

Example b

```
MODIFY 5, 'OW'
```

changes the page, as in example 1, to:

```
'ABCCOWDOGXX...'
```

Example c

```
MODIFY , 3X'C8C1', 10
```

changes the page to:

```
'HAHAHAXX...'
```

The page will be padded at the end with 3 bytes of hexadecimal zeros (X’000000’).

Example d

```
M K16, X'00'
```

followed by

```
M ,,K16
```

sets the key to hexadecimal zero. This is equivalent to:

```
M K1, 16X'00'
```

The second method is faster.

OPEN statement

The OPEN statement permits an open file or volume to be closed and another one to be opened.
Only system administration is authorized to open a volume.

Operation	Operands
{ OPEN | O }	{ filename[, { INOUT | INPUT }] | 'vsn',devtype[, { S[HAREABLE] | E[XCLUSIVE] }] }

Opening a file

filename	Fully qualified file name or name of a file generation group. The file protection offered by DMS takes effect when the file is opened (e.g. password, external access, read/write access).
INOUT	The file is opened for reading and writing.
INPUT	The file is opened for reading only. WRITE statements lead to a PAM write error.

Opening a volume

‘vsn‘	The volume serial number, up to six characters long, of a volume (either public or private) is enclosed in single quotes.
devicetype	Defines the volume device type. See the appendix for possible specifications.
S[HAREABLE]	The volume is opened as a public volume.
E[XCLUSIVE]	The volume is opened exclusively for one task.

Example

OPEN ‘ABCDEF‘,D3455
volume ‘ABCDEF‘ is opened if it is available and can be mounted on a 3455 Disk Storage Unit.

PRINT statement

This statement prints portions of a page or whole pages via the SYSLST file.

Operation	Operands
{ PRINT / P }	{ page / page1-page2 / page-$ / * } [,{ byte / byte1-byte2 / K }]

This statement is identical with the DISPLAY statement except for the fact that output is to SYSLST. The meanings of the operands are given under the DISPLAY statement.

READ statement

This statement reads a page into the internal work area. The data read in thus exists twice: once in the original PAM page and once in the internal work area. The MODIFY statement can be used to modify the contents of the PAM page stored in the internal work area. With the WRITE statement, the data from the internal work area may be written back to the original area. The internal work area is cleared by means of the HALT statement.

Operation	Operands
{ READ R }	page

page Specifies the logical page number to be read. If no page is specified, the next page is read.

Example

```
READ 7
READ 8
READ 3
```

This sequence of statements will cause the DPAGE routine to read pages 7, 8 and 3 consecutively. It should be noted that only one internal work area is provided.

WRITE statement

This statement writes the page currently in the internal work area back to the file (or volume). The page may have been modified by a MODIFY statement.

Operation	Operands
{ WRITE W }	

Example

```
READ 1
MODIFY 10,X‘FF‘
WRITE
```

Page 1 is read to memory.
X‘FF‘ is assigned to the tenth byte.
After modification, page 1 is written back to the file.

Note on programming

DPAGE requires each volume opened to have PAM format, otherwise the results are unpredictable. It should be noted that public volumes with BBS/BPBS format contain IPL (Initial Program Load) and SVL (Standard Volume Label) records in physical pages 1, 2 and 3.

3.4 DPAGE messages

```
CLOSE ERROR, ERROR-CODE: xxxx
```

Meaning
Error on closing a file or disk. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
ERROR IN EDT-CALL
```

Meaning
Program error in the EDT call. EDT could not be started properly.

```
INVALID BS2000-COMMAND
```

Meaning
The specified BS2000 command is invalid, or an error occurred in the CMD macro call.

```
INVALID COMMAND
```

Meaning
Invalid command, no action.

```
INVALID OPERAND, COMMAND REJECTED
```

Meaning
Invalid operand specification, command ignored.

```
NO PAMKEY AVAILABLE
```

Meaning
The file/disk has no PAM key.

```
OPEN COMMAND MUST BE GIVEN FIRST, COMMAND REJECTED
```

Meaning
Wrong statement sequence.

Response
Enter OPEN first.

```
OPEN ERROR, ERROR-CODE: xxxx
```

Meaning
Error on opening a file or disk. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx, consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
OPEN VOLUME RESTRICTED TO SYSTEM ADMINISTRATOR
```

Meaning
Only the system administration is allowed to open a disk.

```
PAM-READ ERROR, ERROR-CODE: xxxx
```

Meaning
Error on reading a PAM page. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx, consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
PAM-WRITE ERROR, ERROR-CODE: xxxx
```

Meaning
Error on writing a PAM page. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx, consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT DISPLAYED.
```

Meaning
DISPLAY was entered, but no page had been read in.

Response
Enter READ first.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT MODIFIED.
```

Meaning
MODIFY was entered, but no page had been read in.

Response
Enter READ first.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT PRINTED.
```

Meaning
PRINT was entered, but no page had been read in.

Response
Enter READ first, or enter PRINT with page specification.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT WRITTEN
```

Meaning
WRITE was entered, but no page had been read in.

Response
Enter READ first.

REQUEST MEMORY ERROR

Meaning
Error on requesting memory via the REQM macro.

SEARCHED STRING NOT FOUND

Meaning
The desired string could not be found.

VOLUME NOT FOUND

Meaning
DPAGE could not find the specified volume.

Response
Enter correct volume serial number or contact system administration.

4 INIT
Initialization of magnetic tapes and floppy disks

Version: **INIT V11.2A**

The INIT utility routine initializes magnetic tapes and 8" floppy disks.

As used in this section, the term ’magnetic tape’ means all types of magnetic tapes and magnetic tape cartridges supported by BS2000.

For magnetic tapes, initialization means writing a volume label (VOL1) and, in some cases, two file labels (HDR1 and HDR2 of a dummy file) at the start of the tape. For magnetic tape cartridges, the labels are never compressed when written, even if write access with compression was specified (volume type TAPE-C2 or TAPE-C4).

Initializing a floppy disk means formatting the volume and then writing a volume label (VOL1) and a file label (HDR1 of a dummy file) on a particular track.

As of BS2000 V10.0, initializing floppy disks is a function of the SPOOL subsystem and therefore requires SPOOL to be loaded. The INIT utility routine calls SPOOL when a floppy disk is to be initialized.

Up to 16 volumes can be initialized with one statement.

A specific tape device or disk drive may be selected for any INIT or LIST function.

In order to prevent data being overwritten by mistake, each volume is checked for existing labels before new labels are created. The contents of existing labels are output to SYSOUT for scrutiny by the user. This means that the user has an opportunity to abort initialization.

Magnetic tapes are always checked for existing labels, unless the BS2000 tape monitor has already recognized the volume as an empty tape and the INIT operand NEW has been specified.

If a magnetic tape is checked and a VOL1, HDR1 or HDR3 label is found to contain an access restriction (access pointer, release date, read or write password, 'read-only' flag), a message to this effect is output.

The functions 'read labels' (LIST statement) and 'write new labels or tape marks' (INIT statement) are not executed unless the following conditions are satisfied:

– if the volume is a magnetic tape, the user must possess the 'TAPE-ADMINISTRATION' privilege

– if the volume is a floppy disk, the INIT utility must be running under the TSOS user ID.

The functions supported by the INIT utility routine are as follows:

1. For magnetic tapes:
   – Write the volume header label VOL1 and the file header labels HDR1 and HDR2 (followed by two tape marks).

     Contents of tape: VOL1-HDR1-HDR2-TM-TM

   – Write only the volume header label VOL1 (followed by two tape marks).

     Contents of tape: VOL1-TM-TM

   – Write two tape marks at the start of the tape. No labels are written.

     Contents of tape: TM-TM

   – Dump the contents of VOL1, HDR1, HDR2 and HDR3 labels, if any, to SYSOUT or console. New labels are not written.

   – Format the magnetic tape, if allowed by the volume.
2. For floppy disks:
   – Reformat and write the volume label VOL1 plus the file header label HDR1.

   – Dump the contents of the VOL1 and HDR1 labels, if any, to SYSOUT or console. New labels are not written.
3. General:
   – Activate or deactivate special functions. The setting remains valid for all INIT and LIST statements for the duration of the program session or until reset.

   – Output an outline description of statements and operands.

   – Switch input/output to console after the program is started with an ENTER task.

4. Security functions:
   - Specify a check volume serial number. A magnetic tape is not processed unless the VSN in its VOL1 label tallies with the check VSN.
   - Erase the entire contents of the tape before writing new labels (DSE = data security erase).

     Note

     In this context, erasure means that the magnetic tape is overwritten with a device-dependent deletion pattern.
     It may nevertheless still be possible to recover the information of earlier records using special devices and technically complex procedures.
     The only way to be absolutely certain than no unauthorized user reads any residual information left on the magnetic tape is to physically destroy the volume.
   - Abort initialization if a read-before-write check finds an access restriction.
   - Exclude special characters (characters not defined in DIN 66003) from use in labels.

4.1 Operating modes

The functions of the INIT utility routine can be executed in two operating modes.

4.1.1 Normal mode

This is the default operating mode when the INIT utility is started with the system command /START-INIT.

START-INIT
VERSION = ***STD** / <product-version 6..10> / <product-version 4..8 without-corr> / <product-version without-man> ,**MONJV** = ***NONE** / <full-filename without-gen-vers> ,**CPU-LIM**IT = ***JOB-REST** / <integer 1..32767>

The VERSION, MONJV and CPU-LIMIT operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in "Commands, Volume 3" [3].

To guarantee compatibility with earlier versions, the command

```
START-PROGRAM FROM-FILE=$INIT
```

is still supported.

Once the program is started, messages are output via SYSOUT and the system awaits inputs from SYSDTA. The program runs as described in section 4.2, “Program run”, on page 63ff. Statements to the INIT routine can be

– entered at the terminal in interactive mode

– contained within a command procedure (reassignment from SYSDTA to SYSCMD)

– contained in an ENTER procedure (SYSDTA reassigned by default to SYSCMD).

Examples

Command procedure:

```
/BEGIN-PROCEDURE
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/START-INIT
       .
       .    (INIT statements)
       .
END
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/END-PROCEDURE
```

ENTER procedure:

```
/LOGON
/START-PROGRAM FROM-FILE=$INIT
      .
      .    (INIT statements)
      .
END
/LOGOFF
```

4.1.2 Console mode

In this operating mode, the INIT utility is controlled by means of a dialog which takes place on the console. Before INIT statements can be entered via the console in interactive mode, an ENTER job must be sent from the console to start the INIT program and OPTION CONS used to direct outputs to the console.

```
/LOGON
/START-INIT
*OPTION CONS
/LOGOFF
```

When the job is started, all messages and inputs are handled via the console, with the task sequence number (TSN) being specified for each operation. The program runs as described in section 4.2, “Program run”, on page 63.

The OPTION CONS statement is valid only within a ENTER job; it is always rejected in interactive mode.

Examples

```
<tsn>.INIT TAPE-C2,VSN=MBK001
<tsn>.LIST T6250
```

<tsn> is the task sequence number.

4.2 Program run

Typical logs are used below as examples to demonstrate how a magnetic tape and a floppy disk are initialized.
Messages marked (OUT) are output to SYSOUT in normal mode, while those preceded by (MSG) are always output to the console.

4.2.1 Program start

Example 1

```
(IN)  /start-init ————————————————————————————————————————————————  (1)
(OUT) %  BLS0523 ELEMENT 'INIT-PRG', VERSION 'V11.2R10' FROM LIBRARY
                 ':1SBZ:$TSOS.SYSLNK.INIT.112' IN PROCESSING
(OUT) %  BLS0524 LLM 'INIT-PRG', VERSION 'V11.2' OF '1995-04-10:10:19:12'
                 LOADED
(OUT) %  BLS0551 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1994.
                 ALL RIGHTS RESERVED
(OUT) %  NVI0000 INIT VERSION V11.2 READY
(OUT) %  NVI0001 ENTER COMMAND. (END = TERMINATE INIT) ————————————  (2)
```

Example 2

```
(IN)  /start-program $init ———————————————————————————————————————  (1)
(OUT) %  BLS0500 PROGRAM 'INIT', VERSION 'V11.2' OF'1995-04-10' LOADED
(OUT) %  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993.
                 ALL RIGHTS RESERVED
(OUT) %  NVI0000 INIT VERSION V11.2 READY
(OUT) %  NVI0001 ENTER COMMAND. (END = TERMINATE INIT) ————————————  (2)
```

Explanation:

(1) The INIT routine is started.

(2) INIT reports ready and awaits statements.

4.2.2 Initializing a magnetic tape (example)

```
(OUT) % NVI0001 ENTER COMMAND. (END = TERMINATE INIT)
(IN) INIT TAPE-C1,VSN=BAND1,ERASE ————————————————————————————————  (3)
```

```
(MSG) %  NKVT013 MOUNT TAPE '*SCRAT' ON DEVICE'T9'MONTIEREN; ————————  (4)
                (USE='SPECIAL',WR='UNDEF',TYPE='TAPE-C1',
                INIT T-C1,VSN=BAND1).
                (ETX = YES; MN; N = NO )'?
(IN)  OABC.T7
```

```
(OUT)  % NVI0003 LABELS ON TAPE(ISO7): ———————————————————————————  (5)
(OUT)  % NVI0004 VOL1 LABEL:
               'VOL1TAPE01                                '
               '                                         1'
(OUT)  % NVI0004 HDR1 LABEL:
                'HDR1BEISPIEL.DATEI   TAPE010001000100010'
                '0 92104 92104 000000BS2000               '
(OUT)  % NVI0004 HDR2 LABEL:
                'HDR2V8000102044 0                 P     '
                '      .C..04                             '
(OUT)  % NVI0004 HDR3 LABEL:
                'HDR3TSOS    BEISPIEL.DATEI               '
                '                 ............00          '
(OUT)  %  NVI0007 OVERWRITE TAPE? ————————————————————————————————  (6)
                 REPLY (YES=YES N=NO)
(IN)   YES
(OUT)  %  NVI0208 DATA SECURITY ERASE STARTET ————————————————————  (7)
(OUT)  %  NVI0209 DATENSICHERHEITSLOESCHEN COMPLETED
(OUT)  %  NVI0210 INITIALIZATION OF TAPE'BAND1' ——————————————————  (8)
                 ON DEVICE 'T7' COMPLETED
(OUT)  %  NVI0001 ENTER COMMAND. (END = TERMINATE INIT) ——————————  (9)
```

Explanation :

(3) To initialize a magnetic tape, enter INIT with a magnetic tape volume type (in this case, with data security erase).

(4) Message output via console requests the operator to mount the volume. Once the volume is mounted (on device T9 in this case), the operator confirms by specifying the task sequence number (TSN). If the volume is mounted on the device specified in the MOUNT request, the mnemonic device name can be omitted.

(5) The volume is read and the labels, if any, are output.
If the labels are written in ISO7 code, the output includes a message to this effect. In this example, the contents of the labels are spread over a number of lines and are delimited by quotation marks. In normal use, the contents of the labels are

output as a single string; blanks at the end of the string are not included.
If the data volume has no volume label (VOL1) or the label is invalid, explanatory messages are issued instead of the labels.

Empty tape :

```
(OUT) % NVI0102 NO LABELS ON TAPE. TAPE EMPTY
```

Tape marks at beginning of tape :

```
(OUT) % NVI0103 NO LABELS ON TAPE. TAPE MARKS READ
```

Error when reading VOL1 label :

```
(OUT) % NVI0104 I/O-ERROR WHILE READING LABELS
```

(6) This is followed by the question whether or not to overwrite the volume. This is the last opportunity to prevent the volume being overwritten. If the answer is affirmative, the word YES must be typed in full. For safety's sake, abbreviations are rejected. All other entries are interpreted as negatives.

(7) If data security erase was requested, the data on the tape is erased. This procedure can take several minutes, depending on the device and the volume; messages are output to indicate the start and end of the process.

(8) After deletion, the new labels are written to tape and a message is output indicating that initialization is completed.
If initialization is unsuccessful, a message is output indicating the source of the problem and initialization is aborted.

```
(OUT) % NVI0309 TAPE WITH VSN ' ' CANNOT BE USED.
                INITIALIZATION ABORTED
```

(9) The system prompts for the next statement.

4.2.3 Initializing a floppy disk

```
OUT)  %  NVI0001 ENTER STATEMENT. (END = TERMINATE INIT)
(IN)   INIT FD,VSN=DISK01 ──────────────────────────────────────────── (10)
(OUT)  UNIT FOR FLOPPY DISK INIT: MN=D1 ─────────────────────────────── (11)
(OUT)  IF READY: CONTINUE WITH (YES/NO)
(IN)   YES
(OUT)  LABELS ON FLOPPY DISK: ──────────────────────────────────────── (12)
(OUT)  VOL1VSN001                          OWN
(OUT)  ERNAM                          1    03 2
(OUT)  HDR1 DATA               080 01001 74026
(OUT)                                    01001
(OUT)  OVERWRITE? (YES/NO) ─────────────────────────────────────────── (13)
(IN)   YES
```

```
(MSG)  %  SPS0433 INITIALIZATION OF VOLUME 'DISK01'
       STARTET ──────────────────────────────────────────────────────── (14)
(MSG)  %  SPS0431 INITIALIZATION OF VOLUME 'DISK01'
      COMPLETED
```

```
(OUT)  LABEL-UPDATE STARTED ─────────────────────────────────────────── (15)
(OUT)  VOL1: DISK01 WRITTEN ON MN=D1 ────────────────────────────────── (16)
(OUT)  %  NVI0001 ENTER STATEMENT. (END = TERMINATE INIT)
```

Explanation :

(10) The INIT statement is specified with ’FDISK’ as the volume type.

(11) If the user does not explicitly define a device (no entry for UNIT=), the operating system suggests a free floppy disk drive. Once the user has inserted the floppy disk to be initialized, he or she must confirm by entering ’YES’.

(12) The index track is read and the labels read are output. The labels are output in the form shown above.

(13) Before initialization commences, the INIT routine asks whether or not to overwrite the volume.
This is the last opportunity to prevent the data on the volume being overwritten.
If the answer is affirmative, the word YES must be typed in full. For safety’s sake, abbreviations are rejected. All other entries are interpreted as negatives.

(14) A message informing the operator that the new labels are being written appears on the console.

(15) Message informing the user that the labels are being updated.

(16) Message indicating that initialization has been successfully completed.
If initialization was not successful, a message indicating the source of the problem is output (for example, if the device does not support double density and DEN=2 was specified), followed by a second message indicating that initialization is aborted:

```
(OUT)  WARNING (PRESENT DEN/SID)
(OUT)  PROBLEMS (UNSUITABLE FD)
```

A message indicating unsuccessful initialization is output on the console:

```
(OUT)   % SPS0432 VOLUME 'DISK01' CANNOT BE USED DUE TO ERROR(S).
                  INITIALIZATION TERMINATED ABNORMALLY
```

4.2.4 Program termination

```
(OUT)  %  NVI0001 ENTER STATEMENT. (END = TERMINATE INIT)
(IN)   END ──────────────────────────────────────────────────────────── (17)
(OUT)  %  NVI0011 PROGRAM INIT TERMINATED NORMALLY ──────────────────── (18)
```

Explanation:

(17) The END statement is entered.

(18) The INIT routine is terminated and a message indicating correct termination is output.

4.2.5 Problems in volume initialization

If an error occurs in the read-before-write check that precedes label updating, when data is erased or when labels are updated, the device error handling system may output messages to the console. These messages contain useful information indicating the source of the error.
If a message requires an answer, the possible answers are shown in the message text. The response must be entered via the console and must include the task sequence number. For example, if the answer required is 'NO', the following entry is required:

```
<tsn>.NO
```

<tsn> is the current task sequence number.

4.3 Entering statements

INIT reads statements from the system file SYSDTA (by default, from the terminal in interactive mode) or from the console (in CONSOLE mode).

An INIT statement consists of the operation name (INIT, LIST, etc.) plus operands with operand values, as applicable. If a statement requires more than one operand, the operands are separated by commas. Blanks are not permissible between the operand and the operand value or values.

An input line cannot consist of more than 72 characters. Excess characters are truncated without warning. Leading blanks are ignored.

If necessary, an INIT statement can be continued on a continuation line or lines. A continuation character (hyphen: '-') must be set at some point in the statement to indicate the presence of continuation lines.
INIT then prompts for continuation of statement input:

```
(OUT) % NVI0002 ENTER ADDITIONAL OPERANDS
```

All characters coming after a blank following the continuation character are ignored (line comment). If characters other than blanks come immediately after a hyphen, the hyphen is not recognized as the continuation character.

Leading blanks in a continuation line are ignored. The operands specified in a continuation line are appended to the preceding line at the point at which the continuation character occurs.

Even if it extends over a number of continuation lines, an INIT statement cannot consist of more than 240 characters (including permissible blanks, excluding leading blanks and character after a continuation character).

Example

```
Statement parts                  Comment
(IN)   INIT 3 T6250,-            INITIALIZE 3 TAPES
(IN)          VSN= (VSN01,-      1. VSN
(IN)                VSN02,-      2. VSN
(IN)                VSN03),-     3. VSN
(IN)          UNIT=A1            ON DEVICE A1
```

This is equivalent to the following input line:

```
INIT 3 T6250,VSN=(VSN01,VSN02,VSN03),UNIT=A1
```

If an asterisk (*) appears in the first column of an input line, the contents of the line in question are not evaluated. By this means, comments can be added to procedures in which the INIT utility routine is called or to a tracer listing.

4.4 Outline description of INIT statements

INIT	Magnetic tapes : Write the volume header label VOL1 and the file header labels HDR1 and HDR2 or write two tape marks. Optional formatting of the volume before writing new labels is possible if the volume type permits it.
	Floppy disks : Reformat and write the volume label VOL1 and the file header label HDR1.
LIST	Output the contents of existing VOL1 and HDR1 labels, plus the contents of HDR2 and HDR3 labels if the case of magnetic tapes.
OPTION	Activate and deactivate special functions; the setting remains valid for all INIT and LIST statements throughout the program session or until reset.
HELP	Output an outline description of statements and operands.
END	Terminate the routine.

4.5 Overview of all INIT functions

Magnetic tape functions

Operation	Operands	Meaning	Page
INIT	[nr] voltyp,VSN=vsnvalue [,UNIT=mn] [,ISO7] [,{REW | RUN}][,OWN={name | *}] [,{CHECK=vsn | NEW}][,ERASE] [,ZERO][,FORMAT]	Write new labels on a magnetic tape	72
INIT	[nr] voltyp,WTM [,UNIT=mn] [,{REW | RUN}][,{CHECK=vsn | NEW}] [,ERASE][,FORMAT]	Write two tape marks at the beginning of tape	72
LIST	[nr] voltyp [,VSN=vsnvalue] [,UNIT=mn] [,{REW | RUN}]	Read and output labels on the magnetic tape	80

Floppy disk functions

Operation	Operands	Meaning	Page
INIT	[nr] FD[ISK],VSN=vsnvalue [,UNIT=mn] [,FEED] [,OWN={name / *}] [,ISO7] [,SEQ=sequence] [,LEN={seclength / (reclength)}] [,DEN=density][,SID=format]	Reformat and write new labels on floppy disks	83
LIST	[nr] FD[ISK] [,UNIT=mn] [,FEED]	Read and output labels on the floppy disk	89

Program control

Operation	Operands	Meaning	Page
OPTION	{CONS / PROT / DIN / NOHDR / NONE}	Activate and deactivate optional functions	90
E[ND]		Terminate the routine	94

Special functions

Operation	Operands	Meaning	Page
HE[LP]		Output an outline description of permitted statements	94

4.6 Description of the individual functions

4.6.1 Statements for magnetic tapes

All the functions of the INIT utility routine that relate to magnetic tapes are described in this section.

INIT - Initialize magnetic tape

The INIT statement in the form described below is used to define at least

– either the volume type (recording density and type) and the VSN (volume sequence number)
– or the initialization mode 'write only tape marks'.

The user can also choose to define

– the number of volumes to be initialized
– a specific volume
– a specific device
– the code for the labels (ISO7 or EBCDIC)
– the way in which the volume is handled after initialization.

Before writing new labels, the INIT routine checks for existing labels. If labels can be read, they are output so that they can be checked.
If OPTIONS CONS is selected, the labels are output to the console; if this option is not selected, they are output to SYSOUT.

Operation	Operands
INIT	[no] voltyp,{VSN={vsn / (vsn1,vsn2,...,vsn16) / (initval) / *} [,OWN={name / *}][,ZERO][,ISO7] / WTM} [,UNIT=mn] [,{CHECK=vsn / NEW}][,{REW / RUN}] [,ERASE] [,FORMAT]

Description of the operands

no — Number of volumes to be initialized with this statement.

Permissible values: 1, 2, ..., 16.

Note

The operands ’no’ and ’voltyp’ must be separated by at least one blank.

voltyp — Defines the volume type, and thus also the recording mode.

Permissible values:

Volume type	**Volume type code**	**Meaning**
T1600 T9P	B2	9-track tape device, PE, 1600 bpi
T6250 T9G	B4	9-track tape device GCR, 6250 bpi
TAPE-C1 T-C1	B5	18-track magnetic tape cartridge device
TAPE-C2 T-C2	B6	18-track magnetic tape cartridge device, large block, compressed
TAPE-C3 T-C3	BB	36-track magnetic-tape cartridge device, large block

Volume type	Volume type code	Meaning
TAPE-C4 T-C4	BC	36-track magnetic tape cartridge device, large block, compressed
TAPE-C5		Device reserved
TAPE-CS1 T-CS1	BA	Magnetic tape cartridge, 155 Mbytes streamer, only for mainframe with bus peripherals
TAPE-V1 T-V1	B9	Video tape, only for mainframe with bus peripherals

There is no default value for this operand; the value must always be specified by the user.

VSN Volume serial number which is to be entered in the VOL1 label (no default). If two or more volumes are to be initialized with a single statement, the user must specify either a corresponding number of VSNs or an initial value.

=vsn Single volume serial number.

A single volume is to be initialized. Consequently, the “no” operand must be either 1 or not specified.

Permissible values:
Max. 6 characters (A..Z, 0..9, #,$,@).

The special characters #,$,@ are permitted by INIT but must not be used in labels conforming to DIN 66029, DIN 66003. If OPTION DIN is active, these special characters are rejected.

= (vsn1,vsn2,...,vsn16)
Two or more volumes are to be initialized with the specified VSNs.
The number of VSNs must tally with the value specified for the “no” operand.

Permissible values: each value must be a valid VSN (see above).

= (initval) Two or more volumes, beginning with the VSN initval, are to be initialized. After every successful initialization, the VSN is automatically incremented by 1.
The number of volumes must be specified in the “no” operand.

Permissible values:
initval must be a valid VSN and must contain at least one decimal digit. The number specified as the initial VSN is read from left to right. The initial value of the VSN to be generated is the first sequence of decimal digits found. If initval contains more than one sequence of digits separated by other characters (e.g. letters), only the string of digits furthest left is used as the

initial value. The number of digits in the initial value identified in this way must also be able to accept the highest VSN; if it cannot, the statement is rejected.

Examples

The statement INIT 5 T9G,VSN=(A12B12) initializes volumes with the VSNs A12B12, A13B12, A14B12, A15B12 and A16B12.
The statement INIT 16 T9G,VSN=(A1B) is rejected, because only single-digit decimal numbers can be formed.

= * The existing VSN is to be used.
If the volume does not have a readable VOL1 label with a valid VSN, a message to this effect is output and no new labels are written.

WTM Write two tape marks at the beginning of tape.
No labels are written.

UNIT = mn Mnemonic device name for seizure of a particular device. A volume is to be mounted on the specified device.
Unsuitable devices, i.e. devices that cannot process volumes of the specified type, are rejected.

CHECK=vsn Specifies a check volume serial number.

A volume is not processed unless the VSN in the VOL1 label tallies with the specified VSN. This mechanism prevents the wrong tape from being overwritten by mistake.
If the CHECK operand is specified and the requested volume is already mounted on the correct device, no MOUNT request is sent to the operator and the volume is initialized immediately.

Permissible values:
Max. 6 characters (A..Z, 0..9, #,$,@).

The special characters #,$,@ are permitted by INIT but must not be used in labels conforming to DIN 66029, DIN 66003. If OPTION DIN is active, these special characters are rejected.

Restrictions

CHECK can be specified only if not more than one volume is to be initialized with this statement (operand no=1 or not specified).

NEW The effect of this operand depends on whether the tape already contains data or if it has been recognized as empty by the tape monitor of the operating system. If the tape is empty , both the read-before-write check of the labels and the OVERWRITE query are suppressed.
If a tape already contains data, the read-before-write check of the labels is implemented without error handling.

If the NEW operand is not specified, INIT always attempts to read existing labels. This incorporates full error handling if the tape monitor of BS2000 has not already recognized the tape as empty during mounting. For safety's sake, INIT attempts to read without error handling in this event.

REW Specifies that the magnetic tape is to be rewound after initialization but not unloaded.
If a single volume is initialized, REW is the default.

RUN Specifies that the magnetic tape is to be rewound and unloaded after initialization.
If two or more volumes are to be initialized with a single statement (operand no=2..16), RUN is the default value.

ERASE All the data on the magnetic tape is to be erased before the labels are updated.

Note

Depending on the device and the volume, this procedure may take several minutes.

If this operand is not specified, only the new labels are written at the beginning of the tape and the logical end of tape is marked by means of two tape marks. The original contents of the tape behind these marks, however, is retained, and under certain circumstances it may still be readable (see note under "Security functions" on page 59).
If INIT can read a valid VOL1 label, this label is retained after erasure. If not, everything from the beginning of tape onward is erased.

FORMAT The volume is formatted, if the volume type and the device permit this. If not, an explanatory message is displayed, the operand is ignored and the initialization procedure continues.

Note

It may take several minutes to format a volume.

The following operands are valid only if labels are to be written, i.e. if the WTM operand is not specified.

OWN Valid only if WTM is not specified. Name of the owner to be entered in the VOL1 label.

= name Name to be entered in the VOL1 label.

Permissible values:
Max. 8 characters (A...Z, 0...9, #, $, @, ., -)

Default value:
Blanks are entered if nothing is specified.

= * If an owner's name already exists in the VOL1 label, it is to be transferred to the new VOL1 label. If the volume does not have a legible VOL1 label with a valid owner's name (blanks are also a valid name), a message to this effect is output and no new labels are written.

ZERO In the VOL1 label (byte 11), the printable character 0 is entered as the access flag. If this operand is not specified, a blank is entered as the access flag.

ISO7 The labels are to be written in ISO 7-bit code.

Examples for writing labels

Example 1

Initializing a magnetic tape (specifying minimum parameters):

```
INIT T9G,VSN=VSN001
```

Example 2

Initializing

- three magnetic tapes
- with one statement
- with explicitly specified VSNs
- with special characters in the VSNs:

```
INIT 3 T6250,VSN=(VSN#00,VSN$00,VSN@00)
```

Example 3

Initializing

- 8 magnetic tape cartridges
- with one statement
- with consecutive VSNs:

```
INIT 8 T-C3,VSN=(VSN001)
```

The VSNs assigned to the volumes are VSN001, VSN002, ...,VSN008.

Example 4

Initializing

- a virgin magnetic tape cartridge
- on a specified device (with mnemonic name M1):

```
INIT TAPE-C2,VSN=MBK01,UNIT=M1,NEW
```

Example 5

Initializing

- a particular magnetic tape (with volume serial number VSN001)
- on a particular device (with mnemonic name T1)
- in ISO 7-bit code
- unloading it after initialization
- with entry of an owner identifier
- with data security erase
- with entry of ’0’ as access code:

```
INIT T1600,VSN=TAPE01,CHECK=VSN001,UNIT=T1,ISO7,RUN,-
OWN=RZ#A0001,ERASE,ZERO
```

Example 6

Initializing

- A magnetic tape cartridge of type TAPE-C5
- A preceding format operation

```
INIT TAPE-C5,VSN=MBK007,FORMAT
```

Examples for writing two tape marks

Example 1

Writing two tape marks at the beginning of tape, specifying minimum parameters:

```
INIT T9G,WTM
```

Example 2

Writing two tape marks at beginning of tape

- on three virgin magnetic tapes
- with one statement:

```
INIT 3 T6250,WTM,NEW
```

Example 3

Writing two tape marks at the beginning of tape

- on a magnetic tape cartridge
- on a particular device (with mnemonic name M1)
- and with data security erase:

```
INIT T-C2,WTM,UNIT=M1,ERASE
```

Example 4

Writing two tape marks at the start of tape

- of a particular magnetic tape cartridge (with volume serial number VSN001)
- on a particular device (with mnemonic name M1)
- with data security erase
- and unloading it after initialization:

```
INIT T-C2,WTM,UNIT=M1,CHECK=VSN001,RUN
```

LIST - Read and output magnetic tape labels

The LIST statement is used in the form described below to read and output volume labels. The user can choose

– the number of volumes to be listed

– a particular volume

– and a particular device.

If OPTION CONS is set, the list is output to the console. If this option is not set, the list is output to SYSOUT.

Operation	Operands
LIST	[no] voltyp [,VSN={vsn / (vsn1,vsn2,...,vsn16) / (initval)}] [,UNIT=mn] [,{REW / RUN}]

Description of the operands

no — For a description of this operand, see the INIT statement on page 72.

voltyp — For a description of this operand, see the INIT statement on page 72.

VSN — The labels of the magnetic tape with the specified volume serial number are to be output. To list two or more volumes with a single statement, the appropriate number of volume serial numbers or an initial value must be specified.

Default value:
If the VSN is not specified, a magnetic tape having any volume serial number is requested.
In this way, it is possible to read the labels of an unknown magnetic tape.

Permissible values:
Max. 6 characters (A..Z, 0..9, #,$,@).
The special characters #,$,@ are permitted by INIT but must not be used in labels conforming to DIN 66029, DIN 66003. If OPTION DIN is active, these special characters are rejected.

= vsn Single volume serial number.
The labels of a single volume are to be output. If a single volume serial number is specified, the value of the “no” operand must be 1 or “no” must not be specified.

= (vsn1,vsn2,...,vsn16)
The labels of the magnetic tapes with the specified volume serial numbers are to be output. The corresponding volumes are requested in consecutive order. The number of volumes must tally with the number specified for the no operand.

= (initval) The labels of multiple magnetic tapes beginning with the tape with the volume serial number initval are to be output. Every time the labels of a volume are read successfully, the volume serial number is automatically incremented by 1. The number of volumes must be specified in the no operand.

UNIT = mn Mnemonic device name for seizure of a particular device. A volume is to be mounted on the specified device.
Unsuitable devices, i.e. devices that cannot process volumes of the specified type, are rejected.

REW Specifies that the magnetic tape is to be rewound after listing but not unloaded.
If a single volume is listed, REW is the default.

RUN Specifies that the magnetic tape is to be rewound and unloaded after listing.
If two or more volumes are to be listed with a single statement (operand no=2..16), RUN is the default value.

Examples

Example 1

Outputting the labels of a single magnetic tape with any volume serial number (specifying minimum parameters):

```
LIST T6250
```

Example 2

Outputting labels of three particular magnetic tapes with a single statement:

```
LIST 3 T1600,VSN=(TAPE01,TAPE02,TAPE03)
```

Example 3

Outputting the records of 8 particular magnetic tape cartridges with a single statement:

```
LIST 8 T-C3,VSN=(VSN001)
```

The volumes with the volume serial numbers VSN001, VSN002,...,VSN008 are requested one after the other.

Example 4

Outputting the labels

- of a particular magnetic tape cartridge (with the volume serial number VSN001),
- on a particular device (with the mnemonic name M1),
- and remove after initialization:

```
LIST T-C2,VSN=VSN001,UNIT=M1,RUN
```

4.6.2 Statements for floppy disks

All the statements referring to floppy disks are described below.

INIT FD - Initialize floppy disk

The INIT statement is used in the form described below to format a floppy disk and write new labels.

The user can choose to specify

– the number of volumes to be initialized
– a particular device
– the way in which the volume is to be handled after initialization
– the code for the labels (ISO7 or EBCDIC)
– the density, number of sides and sector length for formatting.

Before writing new labels, the INIT routine checks for existing labels. If labels can be read, they are output so that they can be checked.
If OPTION CONS is selected, the labels are output to the console; if this option is not selected, they are output to SYSOUT.

Operation	Operands
INIT	[no] FD[ISK], VSN={vsn / (vsn1,vsn2,...,vsn16) / (initval) / *} [,UNIT=mn] [,FEED] [,OWN={name / *}] [,ISO7] [,SEQ=sequence] [,LEN={seclength / (reclength)}] [,DEN=density] [,SID=format]

Description of the operands

no — Number of volumes to be initialized with this statement.

Permissible values: 1, 2, ..., 16.

FD[ISK] — Specifies that a floppy disk is to be initialized.

VSN — Volume serial number which is to be entered in the VOL1 label (no default). If two or more volumes are to be initialized with a single statement, the user must specify either a corresponding number of VSNs or an initial value.

= vsn — Single volume serial number.

A single volume is to be initialized. Consequently, the “no” operand must be either 1 or not specified.

Permissible values:
Max. 6 characters (A..Z, 0..9, #,$,@).

The special characters #,$,@ are permitted by INIT but must not be used in labels conforming to DIN 66029, DIN 66003. If OPTION DIN is active, these special characters are rejected.

= (vsn1,vsn2,...,vsn16)

Multiple volumes are to be initialized with the specified VSNs.

The number of VSNs must tally with the value specified for the no operand.

Permissible values: Each value must be a valid VSN.

= (initval) — Multiple volumes, beginning with the VSN initval, are to be initialized. After every successful initialization, the VSN is automatically incremented by 1. The number of volumes must be specified in the “no” operand.

Permissible values:
initval must be a valid VSN and must contain at least one decimal digit. The number specified as the initial VSN is read from left to right. The initial value of the VSN to be generated is the first sequence of decimal digits found. If initval contains more than one sequence of digits separated by other characters (e.g. letters), only the string of digits furthest left is used as the initial value. The number of digits in the initial value identified in this way must also be able to accept the highest VSN; if it cannot, the statement is rejected.

= * — The existing VSN is to be used.
If the volume does not have a legible VOL1 label with a valid VSN, a message to this effect is output and no new labels are written.

UNIT = mn
: Two-character mnemonic device name for seizure of a particular device. A volume is to be mounted on the specified device.
Unsuitable devices, i.e. devices that cannot process volumes of the specified type, are rejected.

FEED
: Once the floppy disk has been initialized, it is to be removed and the next floppy disk loaded from the stack.

Restriction

Applicable only to devices connected via channels of type 2.

OWN
: Name of the owner to be entered in the VOL1 label.

= name
: Name to be entered in the VOL1 label.

Permissible values:
Max. 8 characters (A...Z, 0...9, #, $, @, ., -)

Default value:
Blanks are entered by default.

= *
: If an owner's name already exists in the VOL1 label, it is to be transferred to the new VOL1 label. If the volume does not have a legible VOL1 label with a valid owner's name (blanks are also a valid name), a message to this effect is output and no new labels are written.

ISO7
: The labels are to be written in ISO 7-bit code.

Restriction

Possible only with FD3171.

SEQ=sequence
: Specifies the sector sequence.

Permissible values:
Decimals 00 to 13

Default value:
If the SEQ operand is not specified, the default values are as follows:

Drive	Default value
FD3171 andere	03 07

Restrictions

– The SEQ operand can be specified only if the default is used for the operand LEN=seclength.
– 01 must be specified for volume interchange with non-SNI DP systems.

LEN=seclength

Physical sector length to be selected when a floppy disk is formatted.

Permissible values:

Drive	seclength (in bytes)	Default value
Channel type 1: FD3171	 128 256 512 1024 256 512 1024 2048 with DEN=2	 128 256
Channel type 2: FD75407-2	 128 256 512 256 512 1024 with DEN=2	
Others	128 256 512 1024	128

Note

The sector length must be 128 bytes for data interchange with non-SNI DP systems.

= (reclength)

Logical record length to be selected. This value specifies the maximum usable number of characters per sector. The logical record length has no influence on floppy disk formatting. The specified value is merely entered right-justified in the HDR1 label (positions 23-27). Blanks are used as fillers where characters are missing.

Example

LEN=(012) Entry : '_ _012'

Permissible values:
Minimum values: 1/01/001/0001/00001
Maximum values: 128/0128/00128

Default value:
If no sector length or LEN=128 is specified for single-density floppy disks, the disks are formatted with a physical sector length of 128 bytes.
The default value '_ _080' is entered as the logical record length in the HDR1 label.

DEN=density Desired density when a floppy disk is formatted.

Permissible values:

– 1 or S: single density (SINGLE)
– 2 or D: double density (DOUBLE)

Default value
If the DEN operand is not specified, single density is the default (DEN=1).

Restriction

Possible only for FD3171 and 75407-2 drives.

SID=format Specifies whether floppy disks are to be formatted as single-sided or double-sided disks.

Permissible values:

– 1 or S: single-sided
– 2 or D: double-sided

Default value:
If the SID operand is not specified, the default value is

– single-sided formatting (SID=1) for DEN=1
– double-sided formatting (SID=2) for DEN=2.

Restrictions

Possible only for FD3171 and 75407-2 drives. The combination of single-sided formatting (SID=1) and double density (DEN=2) is not permitted.

Examples

Example 1

Initializing a floppy disk (specifying minimum parameters):

```
INIT FD,VSN=VSN001
```

Example 2

Initializing

– three floppy disks
– with one statement
– with explicitly specified volume serial numbers
– with special characters in the volume serial numbers:

```
INIT 3 FDISK,VSN=(VSN#00,VSN$00,VSN@00)
```

Example 3

Initializing

- 8 floppy disks
- with one statement
- with consecutive archive numbers:

```
INIT 8 FD,VSN=(VSN001)
```

The volume serial numbers VSN001,VSN002,...,VSN008 are assigned to the volumes.

Example 4

Initializing

- one floppy disk
- on a particular device (with mnemonic name F1)
- in ISO 7-bit code
- with a physical sector length of 512 bytes
- with double density
- double-sided:

```
INIT FD,VSN=DISK01,UNIT=F1,ISO7,LEN=512,DEN=2,SID=2
```

Example 5

Initializing

- one floppy disk
- with a logical record length of 64 bytes
- with a sector sequence of 5
- with entry of an owner's name
- with automatic feed of another floppy disk:

```
INIT FDISK,VSN=DISK02,LEN=(0064),SEQ=05,OWN=RZ#A0001,FEED
```

LIST FD - Read and output floppy disk labels

The LIST statement in the form described below is used to read and output the labels of a floppy disk.
The operator can choose to specify

– the number of volumes to be listed

– a particular device and

– the way in which the volume is to be handled after processing.

If OPTION CONS is set, the information is output to the console. If this option is not set, the information is output to SYSOUT.

Operation	Operands
LIST	[no] FD[ISK] [,UNIT=mn] [,FEED]

Description of the operands

no — Number of volumes to be processed with this statement.

Permissible values: 1, 2, ..., 16.

FD[ISK] — Specifies that the labels of a floppy disk are to be read and output.

UNIT = mn — Mnemonic device name for seizure of a particular device. A volume is to be mounted on the specified device.
Unsuitable devices, i.e. devices that cannot process volumes of the specified type, are rejected.

FEED — Once the floppy disk has been processed, it is to be removed and the next floppy disk loaded from the stack.

Restriction

Applicable only to devices connected via channels of type 2.

Examples

Example 1

Outputting the labels of a single floppy disk (specifying minimum parameters):

```
LIST FD
```

Example 2

Outputting the labels

- of 8 floppy disks
- with one statement
- on a particular device (with mnemonic name F1)
- with automatic feed of a new floppy disk:

```
LIST 8 FDISK,UNIT=F1,FEED
```

4.6.3 Control statements

OPTION - Activate and deactivate optional functions

The OPTION statement is used to activate and deactivate optional functions; the setting remains valid for all subsequent INIT or LIST operations. The options are enforced until the program is ended or until reset.

Operation	Operands
OPTION	{ CONS | PROT | DIN | NOHDR | NONE }

Description of the operands

CONS — Diverts inputs/outputs to the console once the program has been started from the console by means of the ENTER job.

Restrictions

CONS is valid only within an ENTER job and is rejected in interactive mode.

PROT — Aborts initialization of a magnetic tape if the read-before-write check discovers an access restriction.

Each of the following criteria is a valid access restriction:

– The access flag in the VOL1 volume label permits access to the volume only by the owner (byte 11 in the VOL1 label contains neither ’0’ nor ’␣’ (space)).

– The access flag in the file label of the first file permits access to the file only by the user (byte 54 in the HDR1 label contains neither ’0’ nor ’␣’ (space)).

– The expiration date of the first file is not yet reached (bytes 48-53 in the HDR1 label).

– The first file is protected by a read and/or write password (bytes 57-64 in the HDR3 label).

– The first file is write-protected (byte 69 in the HDR3 label contains a ’1’).

In all the cases listed above, the labels are read and output, followed by a message defining the situation and a prompt calling for another statement. If the user wants to initialize the tape despite the access restriction, the OPTION NONE statement must be used to revoke the restriction.

DIN — Excludes special characters that do not conform to DIN 66003 from being used in labels. If the user wants to employ special characters (#, $, @) in labels nevertheless, OPTION NONE must be used to revoke the restriction.

NOHDR — Initializes a magnetic tape with the VOL1 volume label only.
Once this option is activated, all subsequent initializations are implemented without HDR labels until the option is reset. In order to resume initialization with the HDR1 and HDR2 labels, OPTION NONE must be used to revoke the NOHDR setting.

NONE — Resets all options.

Note

Option CONS cannot be reset.

Examples

Example 1

OPTION PROT

```
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   OPTION PROT ──────────────────────────────────────────────────  (1)
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   INIT T-C4,VSN=TEST01 ─────────────────────────────────────────  (2)
(OUT)  %  NVI0003 LABELS ON TAPE       : ────────────────────────────  (3)
(OUT)  %  NVI0004 VOL1 LABEL:
                  'VOL1DAR15K                                '
                  '                                        1'
(OUT)  %  NVI0004 HDR1 LABEL:
                 'HDR1V11.TEST-R.TXT.TEDAR15K0001000100010'
                 '0 92104 92107 000000BS2000                '
(OUT)  %  NVI0101 EXPIRATION DATE OF FIRST FILE NOT YET REACHED ──────  (4)
(OUT)  %  NVI0004 HDR2 LABEL:
                 'HDR2V8000102044 0                 P     '
                 '      .C..04                            '
(OUT)  %  NVI0004 HDR3 LABEL:
                 'HDR3TSOS    V11.TEST-R.TXT.TESTDATEI.T-C'
                 '4               ............00          '
(OUT)  %  NVI0201 TAPE PROTECTED. ──────────────────────────────────  (5)
                  INITIALIZATION REQUEST REJECTED
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
```

Explanation:

(1) OPTION PROT is set.

(2) A magnetic tape cartridge is to be initialized.

(3) The labels of the volume are output. The expiration date of the first file on the volume has not yet been reached (bytes 48-53 of the HDR1 label).

(4) A message to this effect is output.

(5) No new labels are written. Initialization is aborted.

Example 2

OPTION DIN

```
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   OPTION DIN ──────────────────────────────────────────────────────  (1)
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   LIST T9G,VSN=T#$@ ──────────────────────────────────────────────  (2)
(OUT)  %  NVI0405 INVALID OPERAND VALUE ',VSN=T#'
(OUT)  %  NVI0001 ENTER COMMAND (WND = TERMINATE INIT)
(IN)   INIT T9G,VSN=DAR10A,OWN=TEST#$@ ────────────────────────────────  (3)
(OUT)  %  NVI0405 INVALID VALUE ',OWN=TEST#'
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
```

Explanation:

(1) OPTION DIN is set.

(2) The labels of a tape with a volume serial number containing special characters are to be output. This statement is rejected.

(3) A tape is to be initialized and the owner's name entered in the VOL1 label. The name in question includes special characters. This statement is rejected.

Example 3:

OPTION NOHDR

```
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   OPTION NOHDR ────────────────────────────────────────────────────  (1)
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)  INIT T9G,VSN=TAPE01,CHECK=TAPE01 ─────────────────────────────────  (2)
(OUT)  %  NVI0003 LABELS ON TAPE       ─────────────────────────────────  (3)
(OUT)  %  NVI0004 VOL1 LABEL: 'VOL1TAPE01...
(OUT)  %  NVI0004 HDR1 LABEL: 'HDR1      ...
(OUT)  %  NVI0004 HDR2 LABEL: 'HDR2U00001...
(OUT)  %  NVI0007 OVERWRITE TAPE ? ─────────────────────────────────────  (4)
          REPLY (YES=YES  N=NO)
(IN)   YES
(OUT)  %  NVI0010 INITIALIZATION OF TAPE 'TAPE01'
          ON DEVICE 'TA' COMPLETED
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   LIST T9G,VSN=TAPE01 ─────────────────────────────────────────────  (5)
(OUT)  %  NVI0003 LABELS ON TAPE       : ────────────────────────────────  (6)
(OUT)  %  NVI0004 VOL1 LABEL: 'VOL1TAPE01...
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
```

Explanation:

(1) OPTION NOHDR is set.

(2) A tape is to be initialized.

(3) The VOL1, HDR1 and HDR2 labels on the tape are output.

(4) The tape is reinitialized.

(5) The labels are to be output.

(6) The tape is initialized with a VOL1 label and no others.

END - Terminate the INIT session

The END statement terminates the INIT utility routine. The INIT subsystem is released for unloading.
In command procedures, the INIT routine is ended even without an END statement as soon as a BS2000 command is read (beginning with /) or the end of the file is reached.

Operation	Operands
E[ND]	

4.6.4 Special functions

HELP - Outline description of INIT statements

The HELP statement calls up outline information on all operations and operands in the INIT utility routine.
If OPTION CONS is set, the information is output to the console. If this option is not set, the information is output to SYSOUT.

Operation	Operands
HE[LP]	

4.7 Structure of the labels

The general format of the labels complies with the provisions of the standards DIN 66029 (magnetic tapes) and DIN 66239 (floppy disks).

Magnetic tape labels are 80 characters long. Floppy disk labels are 128 characters long.

Only the structure and contents of those magnetic tape and floppy disk labels read or written by INIT are described below.
The column headed 'Field contents' contains the information that INIT writes into the field in question (or the possible contents in the case of the HDR3 label).

Key to symbols

x: Current value

_: Blank

The other fields contain fixed defaults.

4.7.1 Volume label VOL1 for magnetic tapes

Byte no.	Field name	Length	Field contents
1 - 3	Label name	3	VOL
4	Label number	1	1
5 - 10	Volume serial number <vsn>	6	xxxxxx
11	Access flag <flg>	1	_ or 0
12 - 37	- reserved -	26	_..._
38 - 51	Owner label <owner>	14	_ _ _ _xxxxxxxx_ _
52 - 79	- reserved -	28	_..._
80	Standard flag	1	1

V	O	L	1		vsn					zer	_	_	_	_	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
_	_	_	_	_	_	_	_	_	owner						
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	1
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

4.7.2 File label HDR1 for magnetic tapes

Byte no.	Field name	Length	Contents
1 - 3	Label name	3	HDR
4	Label number	1	1
5 - 21	File name	17	_..._
22 - 27	Volume serial number <vsn>	6	xxxxxx
28 - 31	File section number	4	0001
32 - 35	File sequence number	4	0001
36 - 39	Generation number	4	0001
40 - 41	Version number	2	00
42 - 47	Creation date <credat>	6	xxxxxx
48 - 53	Expiration date <reldat>	6	xxxxxx
54	Access flag	1	_
55 - 60	Block counter	6	000000
61 - 73	System code	13	_..._
74 - 80	- reserved -	7	_..._

Format of the creation date and expiration date : cyyjjj

c = Code for century: blank=20th century, 0 = 21st century.

yy = Year

jjj = Julian date (number of the day in the year).

Examples

July 3, 1992: _92185

January 10, 2002: 002010

H	D	R	1	_	_	_	_	_	_	_	_	_	_	_	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

_	_	_	_	_	vsn						0	0	0	1	0
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

0	0	1	0	0	0	1	0	0	credat						
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48

reldat					_	0	0	0	0	0	0	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

4.7.3 File label HDR2 for magnetic tapes

Byte no.	Field name	Length	Contents
1 - 3	Label name	3	HDR
4	Label number	1	2
5	Record format	1	U
6 - 10	Block length	5	00001
11 - 15	Record length	5	00001
16 - 50	- reserved -	35	_..._
51 - 52	Buffer displacement	2	00
53 - 80	- reserved-	28	_..._

H	D	R	2	U	0	0	0	0	1	0	0	0	0	1	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
_	_	0	0	_	_	_	_	_	_	_	_	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

4.7.4 File label HDR3 for magnetic tapes

The HDR3 label is never written as part of initialization. If it exists, however, it is output when the labels are read.

Byte no.	Field name	Length	Contents
1 - 3	Label name	3	HDR
4	Label number	1	3
5 - 12	Owner ID <uid>	8	xxxxcccc
13 - 56	File name <fnam>	44	ccc...ccc
57 - 60	Read password <rpass>	4	cccc or X' 00000000'
61 - 64	Write password <wpass>	4	cccc or X' 00000000'
65 - 68	Expiration password <epass>	4	cccc or X' 00000000'
69	Access type <acc>	1	0 = read + write 1 = read only
70 - 80	-reserved -	11	_..._

H	D	R	3	uid											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
fnam															
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
								rpass				wpass			
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
epass				acc	_	_	_	_	_	_	_	_	_	_	_
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

4.7.5 Volume label VOL1 for floppy disks

Byte no.	Field name		Length	Field contents
1 - 3	Label name		3	VOL
4	Label number		1	1
5 - 10	Volume serial number	<vsn>	6	xxxxxx
11	Access flag		1	_
12 - 37	- reserved -		26	_..._
38 - 51	Owner label	<owner>	14	xxxxxxxx_ _ _ _
52 - 71	- reserved -		20	_..._
72	Side indicator	<sid>	1	x
73 - 75	- reserved -		3	_ _ _
76	Length indicator for sectors	<slid>	1	x
77 - 78	Sequence indicator for sectors	<seq>	2	xx
79	- reserved -		1	_
80	Vendor label indicator	<vid>	1	x
81-128	- reserved -		48	_..._

Notes

(sid) = 1 : Default

3171 and 75407-2 Floppy Disk Units:

2: Double-sided, single density

M: Double-sided, double density

(slid) = _ or decimal number (depending on the LEN operand)

(vid) = W : Default

2 : 3171 Floppy Disk Unit: both sides of the floppy disk are usable

V	O	L	1	vsn						_	_	_	_	_	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
_	_	_	_	_	owner								_	_	_
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
_	_	_	_	_	_	_	sid	_	_	_	slid	seq		_	lid
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
_	_	_	_	_	_				_	_	_	_	_	_	_
81															128

4.7.6 File label HDR1 for floppy disks

Byte no.	Field name	Length	Contents
1 - 3	Label name	3	HDR
4	Label number	1	1
5	- reserved -	1	-
6 - 13	File name	8	DATA_ _ _ _
14 - 22	- reserved -	9	_ _ _ _ _ _ _ _ _
23 - 27	Block length <len>	5	xxxxx
28	- reserved -	1	_
29 - 33	Start of file	5	01001
34	- reserved -	1	_
35 - 39	End of area <end>	5	xxxxx
40	Record format	1	_
41	Swap indicator	1	_
42	Access flag	1	_
43	Write protect	1	_
44	Swap stage	1	_
45	File sequence marker	1	_
46 - 47	File section number	2	_ _
48 - 53	Creation date	6	_ _ _ _ _ _
54 - 57	Record length	4	_ _ _ _
58 - 62	Pointer to first free record	5	_ _ _ _ _
63	Blocking indicator	1	_
64	File organization	1	_
65 - 66	- reserved -	2	_ _
67 - 72	Release date	6	_ _ _ _ _ _
73	File status indicator	1	_
74	- reserved -	1	_
75 - 79	End of file	5	01001
80 - 128	-reserved -	49	_ ... _

Notes

len = Range between 00001 and 00128 or 1 and 128, entered right-justified with blanks as fillers. Default value: _ _ 080

end = Normally : 73026

for 7.540 mainframe: 74026

Blanks are entered for the creation and release dates. This means that these specifications are of no significance. If values are entered here by someone else, the format is as follows: yymmdd yy=year (00-99), mm=month (00-12), dd=day (00-31).

H	D	R	1	_	D	A	T	A	_	_	_	_	_	_	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

_	_	_	_	_	_	len					_	0	1	0	0
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

1	_	end			_	_	_	_	_	_	_	_	_	_
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48

_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

_	_	_	_	_	_	_	_	_	_	0	1	0	0	1	_
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

_	_	_	_	_	_		_	_	_	_	_	_	_
81													128

5 JMU
Creating and maintaining the SJMSFILE system file

Version: **JMU V11.2A**

The JMU (job management utility) utility routine allows you to create and manage the SJMSFILE system file. The SJMSFILE contains the stream and job class definitions, which are stored in this file in an internal table format. The utility can run in batch or interactive mode.

At system initialization (BS2000 startup), the utility reads the file SJMSFILE and copies the job class and stream definitions to the system.

In addition, JMU can be used to modify specific JMS data while the system is running. It is possible:

– to modify access rights with immediate effect
– to assign suitable job classes to new users
– to modify, delete and create job classes.

5.1 Job management

The function of job management is to manage jobs until they are started.

The job scheduling system, based on job classes, allows an administrative strategy to be defined for the computer center in order to classify users and the system load.

Jobs that share certain characteristics are assigned to the same job class. This applies to jobs in both batch and interactive mode. The relevant characteristics are specified by system administration on defining the classes and determining which user IDs are to be served by a given class.

It is also possible to define default classes, intended for users who have not explicitly specified a class.

By setting a limit for each class and defining class priorities, the computer center can improve control over access to the system and can achieve an optimum mix of jobs, e.g. short-running vs long-running jobs, at any time of day.

By means of job classes it is possible to classify jobs, for example on the basis of CPU time required, so as to favor short-running jobs over long-running ones. It is also possible for system administration to assign privileges to certain users, such as the right to start scheduled or repeat jobs.

A job management utility (JMU) is available for creating and maintaining the file for stream and job class definitions. For a description of job streams, see the manuals "Commands, Volumes 1 - 3" [1], [2], [3].

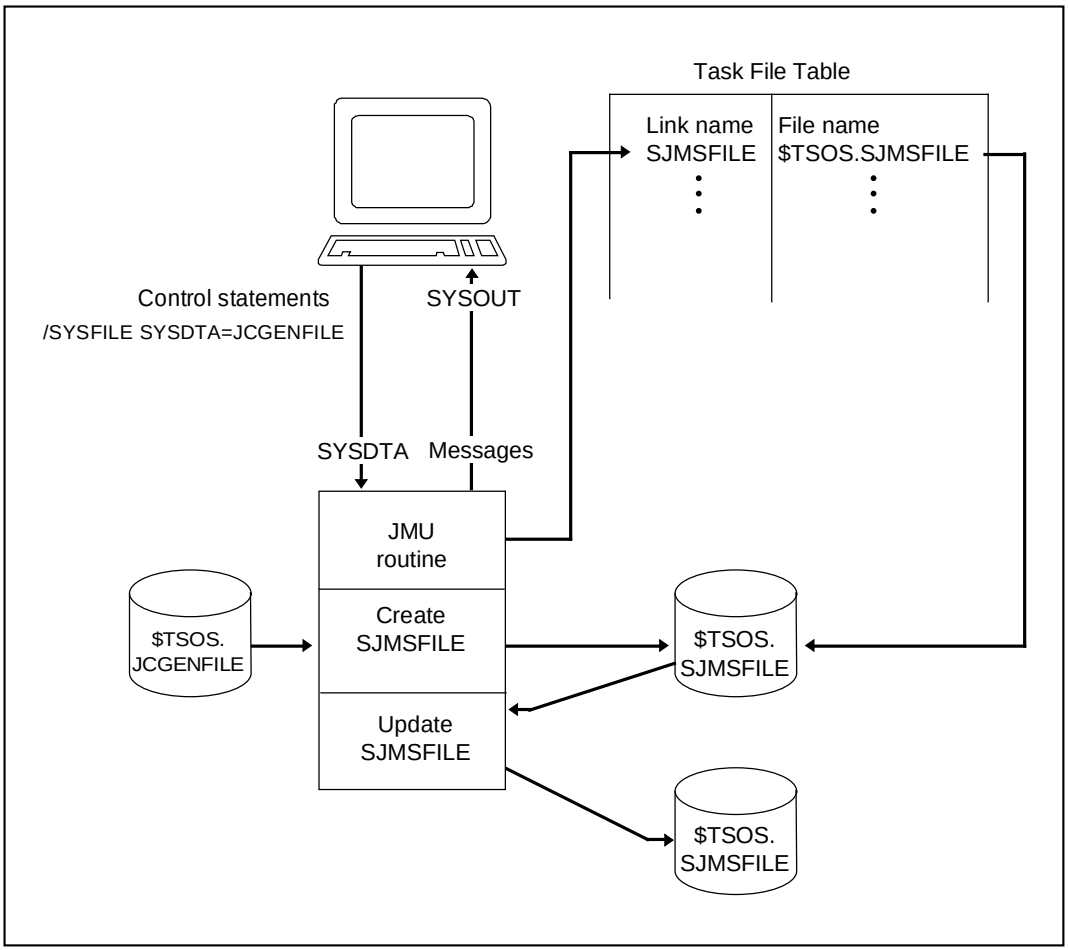


Figure 2: Creating and updating the SJMSFILE

5.2 Execution of JMU

JMU creates the ISAM file SJMSFILE. The file name is ascertained from the task file table (link name is SJMSFILE) and can be defined with the command /ADD-FILE-LINK LINK-NAME=SJMSFILE,FILE-NAME=filename. If the file already exists, JMU updates it.

The file need not necessarily exist before the ADD-FILE-LINK command is executed, or it can exist and be empty. In these cases it is created by JMU.

The ADD-FILE-LINK command is not mandatory. If the link name SJMSFILE has not been assigned, JMU processes the file with the file name SJMSFILE as before and automatically assigns the link name SJMSFILE to it.

The user must not assign the link name SJMSFILE to a file which is not to be processed by JMU.

The routine is called using the following command:

```
/START-PROGRAM FROM-FILE=$TSOS.JMU
```

The `CPU-LIMIT`, `TEST-OPTIONS`, `MONJV`, `RESIDENT-PAGES` and `VIRTUAL-PAGES` operands of the `START-PROGRAM` routine are also available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the `START-PROGRAM` command in the "Commands, Volume 3" manual [3].

JMU is controlled by means of control statements read from SYSDTA.

The file $TSOS.JCGENFILE can be used as the input file.

Since JMU reads its statements from SYSDTA, SYSDTA must first be assigned to JCGENFILE. This is done with the command

```
/ASSIGN-SYSDTA TO-FILE=$TSOS.JCGENFILE
```

In batch mode SYSDTA is the spoolin file, in interactive mode the data terminal.

The SJMSFILE file is updated in the same session in which the control statements were given. However, the updates do not have any effect until the next session.
The desired changes should be entered in a copy of the SJMSFILE, and the updated file used in a session only when the changes have been verified as correct.

It is advisable to maintain a copy of the SJMSFILE, or a procedure for reconstructing it.

Note

There is a risk that it will not be possible to process the SJMSFILE file using the job management systems of operating system versions later than the one used to create the file. It is therefore recommended that a copy of the JMU control statements be preserved for reconstructing the file in a new version of the operating system.

Conversion of the dialog interface to SDF

JMU uses the dialog interface SDF (see the manuals "Commands, Volumes 1 - 3" [1], [2], [3] and "Introductory Guide to the SDF Dialog Interface" [13]).

Syntax errors in interactive mode cause a correction dialog to be initiated with the user. This is one of the major advantages of SDF. A correction dialog is not possible in batch and procedure mode, but SDF does not allow the statement to be simply ignored. All the statements following an incorrect statement are skipped until the system encounters a "//STEP" or "//END" statement. Processing then continues with the statement (or command) following the "//STEP" or "//END" statement. Message CMD0230 is output to inform the user that statements have been omitted.

Constraints imposed by the SDF syntax file

When the SDF dialog interface is used, the JMU syntax definitions must be specified in a syntax file (see "Introductory Guide to the SDF Dialog Interface" [13]). The statements allowed for JMU are identified in this syntax file by the program name "JMU" assigned to them. If a user has defined statements under the name "JMU" for a particular program, effective JMU syntax analysis is no longer possible.

Use of link names by JMU

JMU evaluates the link names SJMSFILE and SJMUPROC. If the SJMSFILE link name is already defined in the task file table, the corresponding file is processed by JMU as the SJMSFILE. Otherwise, SJMSFILE is used as the file name.

When the CREATE-PROCEDURE-FILE statement is processed, the link name SJMUPROC is evaluated and assigned to a file where necessary. This link name should therefore only be used in the cases described under the CREATE-PROCEDURE-FILE statement. Please refer to the section dealing with this statement.

Compatibility

The JMU version supplied with BS2000/OSD-BC V2.0 can process SJMSFILEs generated by JMU versions from BS2000 V8.0 onward (i.e. from the first JMU version). Inconsistencies may occur when a SJMSFILE is processed by a JMU from a version of BS2000 earlier than that under which the SJSMFILE was created. The SJMSFILE link name is not evaluated in JMU versions prior to 250 (old version grid).

5.3 Statements

A statement can extend over more than one line. A continuation character (hyphen) must be entered to indicate that a continuation line follows. Only blanks may occur between the hyphen and the end of the line.

5.3.1 Overview of all JMU statements

Statement	**Meaning**
CREATE-PROCEDURE-FILE	Create a SAM file containing a BS2000 procedure.
DEFINE-JOB-STREAM	Write a new stream definition into the SJMSFILE.
MODIFY-JOB-STREAM	Modify an existing stream definition.
DELETE-JOB-STREAM	Delete an existing stream definition.
DEFINE-JOB-CLASS	Write a new job class definition.
MODIFY-JOB-CLASS	Modify an existing job class definition.
DELETE-JOB-CLASS	Delete an existing job class definition.
GRANT-JOB-CLASS-ACCESS	Grant or prohibit access to a job class for one or more users.
SET-JOB-CLASS-DEFAULT	Define default classes for users.
SET-MODIFICATION-MODE	Change the modification mode.
SHOW-JOB-STREAM	List the contents of stream definitions.
SHOW-JOB-CLASS	List the contents of job class definitions.
REMOVE-USER	Prohibit access to private job classes.
END	Terminate the routine.

5.3.2 Description of the statements

CREATE-PROCEDURE-FILE
Create SAM file containing BS2000 procedure

This statement can be used to create a SAM file containing a BS2000 procedure.
The procedure contains a /START-PROGRAM $.JMU. When JMU is called by START-PROGRAM during execution of the procedure, a new SJMSFILE system file is written. This replaces and corresponds to the SJMSFILE being processed before the procedure started.

This statement can be used to save the status of an open SJMSFILE during processing by means of BS2000 procedures.

The CREATE-PROCEDURE-FILE statement can be used to update the format of an SJMSFILE. If an existing SJMSFILE is processed with a different version of JMU, its format does not change. An SJMSFILE is only formatted according to the JMU version used when it is created as a new file.

However, as a result of the functional enhancements to JMU, a conversion of the SJMSFILE format is only supported if the same JMU version is used to execute both the CREATE-PROCEDURE-FILE statement and the BS2000 procedure.

For this reason the JMU version used to create the procedure is recorded for documentation purposes in a /REMARK command. If a different kind of conversion is to be made, the JMU statements in the procedure created may have to be altered in line with the operating instructions for the JMU version to be called.

CREATE-PROCEDURE-FILE
FILE-NAME = *STD-FILE-LINK / <full-filename 1..54 without-gen-vers> ,OVERWRITE = *NO / *YES

FILE-NAME =
Name of the procedure file to be created. Write access to the file must be permitted.

FILE-NAME = *STD-FILE-LINK
The file name is to be read from the task file table (TFT). The link name, which must not be changed by the user, is SJMUPROC. The user can thus define the file name before calling JMU using the BS2000 command

```
/ADD-FILE-LINK LINK-NAME=SJMUPROC,FILE-NAME=filename
```

If SJMUPROC is not defined as the link name, JMU uses the name SJMUPROC as the file name.

FILE-NAME = <full-filename 1..54 without-gen-vers>
A fully qualified file name. A file generation or file generation group must not be specified, and the file name must not be given in the form "file(no)" (where no = version number).

OVERWRITE =
Allows or prevents an existing file from being overwritten. The file name is specified via the FILE-NAME= operand.

OVERWRITE = *NO
Prevents an existing file from being overwritten. The original file remains unchanged. The user receives the message

```
JMU0114 FILE ALREADY EXISTING. OVERWRITE PROHIBITED BY USER.
```

The procedure file is not created.

OVERWRITE = *YES
If a file of the same name already exists, it is to be overwritten and a procedure file created.

Structure of the created BS2000 procedure

The file name of the SJMSFILE to be created and the JMU load module to be called can be specified as operands for the procedure. The top line (header) of the procedure is structured as follows:

```
/PROC N, (&SJMSFILE=SJMSFILE, &JMU=$TSOS.JMU)
```

Meaning of the operands:

&SJMSFILE stands for the file name of the SJMSFILE to be created. The file named &SJMSFILE is deleted within the procedure so that JMU can create a new one.
The &SJMSFILE operand should not be modified in JMU versions prior to 250 (old version grid).
The default value is SJMSFILE.

&JMU stands for the JMU load module called in the procedure.

The default value is $TSOS.JMU

BS2000 commands within the procedure

The BS2000 commands used in the procedure can be divided into two groups:

1. Immediately after the procedure header there are a number of /REMARK commands which contain information about the SJMSFILE to be created: the SJMSFILE file name defined by the CREATE-PROCEDURE-FILE statement, the date and time this statement was executed, the JMU version used and various characteristics of the SJMSFILE.
2. The commands needed to execute the procedure are:

```
/DELETE-FILE &SJMSFILE
/SET-JOB-STEP "SJMSFILE MAY NOT EXIST"
/ADD-FILE-LINK LINK-NAME=SJMSFILE,FILE-NAME=&SJMSFILE
/ASSIGN-SYSDTA SYSCMD
/START-PROGRAM &JMU
.
.               (JMU statements)
.
/END-PROCEDURE
```

JMU statements within the procedure

The JMU statements DEFINE-JOB-STREAM, DEFINE-JOB-CLASS, GRANT-JOB-CLASS-ACCESS, SET-JOB-CLASS-DEFAULT and END are used to create the SJMSFILE. The statements must be specified in a particular sequence within the procedure. First of all, the stream definitions should be specified in alphabetical order. A DEFINE-JOB-STREAM statement is required for each of these. Next, all job classes are defined, also in alphabetical order. Here, the statements GRANT-JOB-CLASS-ACCESS and SET-JOB-CLASS-DEFAULT may be required in addition to the DEFINE-JOB-CLASS statement to define the user's access rights for these classes. The JMU run is terminated by the END statement.

DEFINE-JOB-STREAM
Write stream definitions to SJMSFILE

This statement causes a new stream definition to be written to the SJMSFILE.

```
DEFINE-JOB-STREAM

 NAME = <name 1..8>
,FILE = <full-filename 1..54> / *LIBRARY-ELEMENT(...)
    *LIBRARY-ELEMENT(...)
        | LIBRARY = <full-filename 1..41>
        | ,ELEMENT = <name 1..8>
,RUN-PRIORITY = 65 / <integer 30..255>
,DEFAULT = *NO / *YES
,START = *AT-LOAD / *BY-OPERATOR / *AT(...)
    *AT(...)
        | TIME = <time 1..8>
,STOP = *AT-SHUTDOWN / *BY-OPERATOR / *AT(...) / *AFTER(...)
    *AT(...)
        | TIME = 00:00 / <time 1..8>
    *AFTER(...)
        | HOURS = 00 / <integer 0..23>
        | ,MINUTES = 00 / <integer 0..59>
,STREAM-PARAMETER = *NO / <c-string 1..127>
```

NAME = <name 1..8>
Name of the stream definition to be written to SJMSFILE.
A string of between 1 and 8 alphanumeric characters may be specified, starting with the character A-Z, @ or #.

FILE = <full-filename 1..54>
Name of the ENTER file containing the job that is initiated during the stream start and that activates the job scheduler.
The job scheduler can only process batch jobs.

FILE = *LIBRARY-ELEMENT(...)

LIBRARY = <full-filename 1..41>
File name of the library.

ELEMENT = <name 1..8>
The library element containing the ENTER file named above.

RUN-PRIORITY = 65 / <integer 30..255>
Specifies the starting priority to be assigned to the stream task under which the job scheduler runs. A priority between 30 and 255 may be specified. The default value is 65.

DEFAULT = *NO / *YES
Specifies whether the stream involved is to be the default stream for the system. The default value is NO. YES means the stream is to be the default stream for those job classes that have specified that they wish to use the default stream.

START =
Specifies when the stream is to be started.

START = *AT-LOAD
The stream is to be started automatically when the system is loaded. AT-LOAD is the default value.

START = *BY-OPERATOR
The stream must be started by the operator or system administration using the START-JOB-STREAM command.

START = *AT(...)
The stream is to be started automatically during each session at the specified time. If a session is started after the specified time, the stream can be started only within the next 30 minutes after the time specified. A start at a later point within the current session is not possible.

TIME = 00:00 / <time 1..8>
Time of day in the format hh:mm; i.e. hours and minutes only, seconds are ignored.

STOP =
Specifies when the stream is to be stopped.

STOP = *AT-SHUTDOWN
The stream is to be stopped when the system is shut down.

STOP = *BY-OPERATOR
The stream must be stopped by the operator or system administration by means of the STOP-JOB-STREAM command.

STOP = *AT(...)
The stream is to be stopped automatically at the specified time (hh:mm see START=*AT...).

TIME = 00:00 / <time 1..8>
Time of day in the format hh:mm; i.e. hours and minutes only, seconds are ignored.

STOP = *AFTER(...)
The stream is to be stopped after the specified time has elapsed.

HOURS = 00 / <integer 0..23>
HOURS= between 0 and 23 hours may be specified.

MINUTES = 00 / <integer 0..59>
MINUTES= between 0 and 59 minutes may be specified.

Note

If for any reason the system is shut down during the time between system start and the time specified for stopping the stream, and the system is then restarted before the time the stream was to be stopped, the stream is also restarted automatically.

STREAM-PARAMETER =
This operand can be used to define special scheduling parameters for the job scheduler in free syntax.
The contents of this operand are not evaluated by the system.
However, the job scheduler involved must understand both the syntax and the meaning of STREAM-PARAMETER = in order to be able to accept the scheduling parameters defined there.

The job scheduler gets this information via the job scheduler interface. The job scheduler interface provides, via a P1 interface, functions required by the job scheduler to carry out its tasks (for further details, see "Introductory Guide to Systems Support" [14])).

STREAM-PARAMETER = <c-string 1..127>
Sequence of special parameters for the job scheduler. When the system is started up the information contained in the parameters is transferred to internal tables, where it can be accessed by the job scheduler.

The following parameters are defined for the default job scheduler:

S-PAR='JOB-PRIORITY={YES / NO}, CPU-TIME={NO / YES}, WAIT-TIME={NO / YES},

JOB-QUOTA={1 / no<256}, LOGGING={YES / NO}, CATID-LIST=(catid1,...),

CAT-TIME=min'

STREAM-PARAMETER = *NO
Means that no special parameters are defined for the job scheduler. *NO is the default value.

Notes

- This statement is rejected if a stream with the specified name is already contained in the SJMSFILE.
- Only one default stream may exist in the system. An attempt to define more than one default stream will be rejected.
- No more than 16 streams may be defined.

MODIFY-JOB-STREAM
Modify stream definitions

This statement enables an existing stream definition to be modified.

```
MODIFY-JOB-STREAM

 NAME = <name 1..8>
,FILE = *UNCHANGED / <full-filename 1..54> / *LIBRARY-ELEMENT(...)
    *LIBRARY-ELEMENT(...)
        LIBRARY = <full-filename 1..41>
        ,ELEMENT = <name 1..8>
,RUN-PRIORITY = *UNCHANGED / <integer 30..255>
,DEFAULT = *UNCHANGED / *NO / *YES
,START = *UNCHANGED / *AT-LOAD / *BY-OPERATOR / *AT(...)
    *AT(...)
        TIME = 00:00 / <time 1..8>
,STOP = *UNCHANGED / *AT-SHUTDOWN / *BY-OPERATOR / *AT(...) / *AFTER(...)
    *AT(...)
        TIME = 00:00 / <time 1..8>
    *AFTER(...)
        HOURS = 00 / <integer 0..23>
        ,MINUTES = 00 / <integer 0..59>
,STREAM-PARAMETER = *UNCHANGED / *NO / <c-string 1..127>
```

When changing the DEFAULT= operand you should bear in mind that there must never be more than one default stream in the system. Any attempt to define more than one will be rejected.

For the meanings of the operands, see the description of the DEFINE-JOB-STREAM statement.

If the operand STOP=*AFTER(...) is modified such that just HOURS or just MINUTES is specified, then a default value of 0 is used for the omitted MINUTES or HOURS operand.

DELETE-JOB-STREAM
Delete stream definitions

This statement enables a stream definition to be deleted from the SJMSFILE.

DELETE-JOB-STREAM
NAME = <name 1..8>

NAME = <name 1..8>
Name of the job stream to be deleted.

Note

If job classes are assigned to the stream, the delete request is rejected.

DEFINE-JOB-CLASS
Write job class definitions to SJMSFILE

This statement is used to write a new job class definition to the SJMSFILE or JMS database and define its characteristics.

```
DEFINE-JOB-CLASS

 NAME = <name 1..8>
,STREAM = *DEFAULT-STREAM / <name 1..8>
,CLASS-LIMIT = <integer 0..4095>
,CLASS-WEIGHT = <integer 1..9>
,CLASS-OPTIMUM = 0 / <integer 0..4095>
,JOB-PRIORITY = *NO / *PARAMETERS(...)
    *PARAMETERS(...)
        DEFAULT = <integer 1..9>
        ,MAXIMUM = *NO / <integer 1..9>
,JOB-TYPE = *BATCH / *DIALOG
,TP-ALLOWED = *NO / *YES(...)
    *YES(...)
        CATEGORY = *TP / <name 1..7>
,DIALOG-ALLOWED = *NO / *YES(...)
    *YES(...)
        CATEGORY = *DIALOG / <name 1..7>
,BATCH-ALLOWED = *NO / *YES(...)
    *YES(...)
        CATEGORY = *BATCH / <name 1..7>
,START-ATTRIBUTE = *BATCH / *DIALOG / *TP
,RUN-PRIORITY = *PARAMETERS(...)
    *PARAMETERS(...)
        DEFAULT = <integer 30..255>
        ,MAXIMUM = *NO / <integer 30..255>
,NO-CPU-LIMIT = *NO / *YES
```

continued ➠

```
,CPU-LIMIT = *PARAMETERS(...)
    *PARAMETERS(...)
        | DEFAULT = <integer 1..32767>
        | ,MAXIMUM = *NO / <integer 1..32767>
,SYSLST-LIMIT = *PARAMETERS(...)
    *PARAMETERS(...)
        | DEFAULT = *NO-LIMIT / <integer 0..999999>
        | ,MAXIMUM = *NO / *NO-LIMIT / <integer 0..999999>
,SYSOPT-LIMIT = *PARAMETERS(...)
    *PARAMETERS(...)
        | DEFAULT = *NO-LIMIT / <integer 0..999999>
        | ,MAXIMUM = *NO / *NO-LIMIT / <integer 0..999999>
,START = *NO / *PARAMETERS(...)
    *PARAMETERS(...)
        | DEFAULT = *SOON / *WITHIN(...)
        |     *WITHIN(...)
        |         | HOURS = 0 / <integer 0..23>
        |         | ,MINUTES = 00 / <integer 0..59>
        | ,ALLOWED = list-poss(7): *AT-STREAM-STARTUP / *AT / *EARLIEST / *SOON /
                        *LATEST / *WITHIN / *IMMEDIATELY
,REPEAT-JOB = *NO / *PARAMETERS(...)
    *PARAMETERS(...)
        | DEFAULT = *NO / *AT-STREAM-STARTUP / *WEEKLY / *DAILY / *PERIOD(...)
        |     *PERIOD(...)
        |         | HOURS = 0 / <integer 0..23>
        |         | ,MINUTES = 00 / <integer 0..59>
        | ,ALLOWED = list-poss(5): *NO / *AT-STREAM-STARTUP / *DAILY / *WEEKLY /
                        *PERIOD
,JOB-PARAMETER = *NO / <c-string 0..127>
```

NAME = <name 1..8>
Name of the new job class definition to be written to the SJMSFILE.
The name may consist of between 1 and 8 alphanumeric characters. The first character must be a letter from the set A through Z or the character @ or #.

Note

The statement is rejected if a job class with the same name already exists.

STREAM = *DEFAULT-STREAM
The default stream defined in the DEFINE-JOB-STREAM statement.

Note

The default stream in the SHMSFILE must not be identical with that of the system.

STREAM = <name 1..8>
Name of the stream under which the job scheduler to which the job class is assigned runs. The name must not be $SYSJS. The name of the stream must already have been defined using the DEFINE-JOB-STREAM statement.

CLASS-LIMIT = <integer 0..4095>
Maximum number of jobs that may be started in the class.
"n" is a value between 0 and 4095.

Note

It is not advisable to specify a value of 0 except in order to prevent jobs from being started following system startup. Otherwise, the job scheduler's performance could be impaired. The specified value is an absolute limit, but it may be exceeded by express jobs.

CLASS-WEIGHT = <integer 1..9>
Determines the start priority of the class relative to other classes whose jobs are waiting to be started.
"n" is a value between 1 and 9, where 1 is the lowest and 9 the highest weight.

CLASS-OPTIMUM =
Specifies the optimum number of jobs that should run in the job class in order to achieve a balanced job distribution within the system.
CLASS-OPTIMUM influences the sequence in which the class scheduler selects the job classes in order to start the jobs.

CLASS-OPTIMUM = 0 / <integer 0..4095>
The number of jobs. Values from 0 up to the value specified in the CLASS-LIMIT operand can be specified: $0 \leq n \leq$ CLASS-LIMIT.
The maximum number permitted is 4095 jobs.

JOB-PRIORITY =
Specifies the job scheduling priority for batch jobs; this determines the priority of a job relative to other jobs of the same class.

JOB-PRIORITY = *NO
This specification is required by the statement format; it has no significance, but must be given if JOB-TYPE=DIALOG was specified.

JOB-PRIORITY = *PARAMETERS(...)

DEFAULT = <integer 1..9>
Default priority for the job class. 1 is the highest and 9 the lowest priority.
If the user has not specified a priority for a job, DEFAULT = <integer...> is used. If the user has specified a priority no higher than MAXIMUM = <integer...>, the priority specified by the user in the ENTER-JOB command applies.
The DEFAULT operand in the DEFINE-JOB-CLASS statement must not specify a priority higher than that given in MAXIMUM, otherwise the statement is rejected with a syntax error.

MAXIMUM = NO / <integer 1..9>
Maximum permitted priority for the job class.
If MAXIMUM=NO is specified, the job is given the priority specified for DEFAULT, irrespective of the priority given by the user in the ENTER-JOB command.

JOB-TYPE =
Defines the type of job class.

JOB-TYPE = *BATCH
Specifies that the job class is to be a batch job class. This means that a job belonging to this class must not be initiated by a LOGON command in interactive mode.

JOB-TYPE = *DIALOG
Specifies that the job class is to be an interactive job class. A job belonging to this class must not be initiated by an ENTER-JOB command.

TP-ALLOWED = *NO / *YES(...)
Specifies whether the task attribute TP is permitted in a job class.

TP-ALLOWED = *NO
Means that the task attribute TP is not allowed in a job class. Jobs in this class must not be started under this task attribute; switching to this task attribute by means of the TINF macro or the MODIFY-TASK-CATEGORIES command is forbidden, unless it is permitted by the JOIN entry.

TP-ALLOWED = *YES(...)

CATEGORY = *TP / <name 1..7>
A category name may be assigned to the task attribute TP. It may be the default category name (TP) or a name freely defined by the user. In defining a name, the user must observe the BS2000 naming conventions. The name must not consist of more than 7 characters.
The standard category name SYS is not allowed.
The number of names defined by the user must not exceed 12.
If the database is updated, the corresponding default category name is used if the specified category is unknown to the system.

DIALOG-ALLOWED = *NO / *YES(...)
Specifies whether the task attribute DIALOG is permitted in a job class.

DIALOG-ALLOWED = *NO
Means that the task attribute DIALOG is not permitted in a job class. Jobs in this class must not be started under this task attribute; switching to this task attribute by means of the TINF macro or the MODIFY-TASK-CATEGORIES command is forbidden, unless it is permitted by the JOIN entry.

DIALOG-ALLOWED = *YES(...)

CATEGORY = *DIALOG / <name 1..7>
A category name may be assigned to the task attribute DIALOG. It may be the default category name (DIALOG) or a name freely defined by the user. In defining a name, the user must observe the BS2000 naming conventions. The name must not consist of more than 7 characters. The standard category name SYS is not allowed.
The number of names defined by the user must not exceed 12.
If the database is updated, the corresponding default category name is used if the specified category is unknown to the system.

BATCH-ALLOWED = *NO / *YES(...)
Specifies whether the task attribute BATCH is permitted in a job class.

BATCH-ALLOWED = *NO
Means that the task attribute BATCH is not permitted in a job class. Jobs in this class must not be started under this task attribute; switching to this task attribute by means of the TINF macro or the MODIFY-TASK-CATEGORIES command is forbidden, unless it is permitted by the JOIN entry.

BATCH-ALLOWED = *YES(...)

CATEGORY = *BATCH / <name 1..7>
A category name may be assigned to the task attribute BATCH. It may be the default category name (BATCH) or a name freely defined by the user. When defining a name, the user must observe the BS2000 naming conventions. The name must not consist of more than 7 characters. The standard category name SYS is not allowed.

The number of names defined by the user must not exceed 12.
If the database is updated, the corresponding default category name is used if the specified category is unknown to the system.

Note

At least one of the three operands must be specified; which one depends on the value specified for the START-ATTR operand, and at least one of these must be entered with YES.
If the START-ATTR operand is not specified, BATCH is assumed as the default value. In that case, BATCH-ALLOWED=*YES must also be specified.
A category name must not have two different task attributes assigned. For example BATCH-ALLOWED=*YES(CATEGORY=HUGO) and DIALOG-ALLOWED=*YES(CATEGORY=HUGO) is ambiguous and therefore not permitted.

START-ATTRIBUTE = *BATCH / *DIALOG / *TP
Defines the task attribute for the job. At the same time the value '=*YES' must be specified for the corresponding task attribute in the TP-ALLOWED, DIALOG-ALLOWED or BATCH-ALLOWED operand, e.g. START-ATTRIBUTE = TP and TP-ALLOWED = *YES(...). If the operand is omitted, BATCH is taken as the default.

RUN-PRIORITY =
Specifies the run priority with which a job is started.

RUN-PRIORITY = *PARAMETERS(...)

DEFAULT = <integer 30..255>
Default value for the job class.
Means that a job is given the priority specified by the user, providing it does not exceed the maximum permitted priority.
If, however, the user's JOIN entry indicates that a higher priority is permitted than that specified in the MAXIMUM operand, the job may exceed the value of MAXIMUM.
The value may be between 30 and 255. DEFAULT must not give a priority higher than MAXIMUM, otherwise the statement is rejected with a syntax error.

MAXIMUM = *NO / <integer 30..255>
Specifies the maximum permitted priority for the job class.
If MAXIMUM = *NO, this means that no maximum task priority is defined. A job is given the priority assigned by the user providing it does not exceed the value in the user's JOIN entry. The default value is MAX=*NO.

NO-CPU-LIMIT = *NO / *YES
Specifies whether jobs in this class may run without a time limit (NTL).
NO means that jobs in this class must not run without a time limit (see the LOGON command). NO is the default. If NTL=YES is specified in the JOIN entry for a user, that user can run jobs without a time limit even when NO-CPU-LIMIT=*NO applies to the job class.

CPU-LIMIT =
CPU time that a job in this class may utilize.

CPU-LIMIT = *PARAMETERS(...)

DEFAULT = <integer 1..32767>
Default value for the job class.

MAXIMUM = <integer 1..32767>
Maximum permissible CPU time for the job class.
If a number is specified, this means that a job may utilize the CPU time requested by the user providing it does not exceed the value specified for MAXIMUM.
The maximum CPU time that may be utilized for an account number depends on the CPU entry in the JOIN file.
The value specified for DEFAULT must not exceed that specified for MAXIMUM, otherwise the statement is rejected with a syntax error.

MAXIMUM = *NO
The job may utilize the CPU time specified in DEFAULT, irrespective of the CPU time the user requested.

SYSLST-LIMIT =
Defines the number of lines for a job when output takes place via SYSLST.

SYSLST-LIMIT = *PARAMETERS(...)

DEFAULT = *NO-LIMIT / <integer 0..999999>
Default number of lines for the job class. *NO-LIMIT means that the number of lines is not limited. The value specified for DEFAULT must not exceed that specified for MAXIMUM, otherwise the DEFINE-JOB-CLASS statement is rejected with a syntax error.

MAXIMUM =
Maximum number of lines permitted for the job class.

MAXIMUM = *NO
The job is assigned the permitted number of lines specified in DEFAULT, irrespective of the number actually requested by the user.

MAXIMUM = *NO-LIMIT
There is no limit on the number of lines for a job in this class. The number specified by the user always applies.

MAXIMUM = <integer 0..999999>
Means that the job is assigned the number of lines specified by the user, providing it does not exceed the value specified for MAXIMUM.

SYSOPT-LIMIT =
Defines the number of lines (cards) for a job when output takes place via SYSOPT.

SYSOPT-LIMIT = *PARAMETERS(...)

DEFAULT = *NO-LIMIT / <integer 0..999999>
Default number of lines for the job class.

MAXIMUM =
Maximum number of lines (cards) a job is permitted to output to SYSOPT.

MAXIMUM = *NO
The job is assigned the permitted number of lines (cards) specified in DEFAULT, irrespective of the number actually requested by the user.

MAXIMUM = *NO-LIMIT
There is no limit on the number of lines (cards) for a job in this class. The number specified by the user always applies.

MAXIMUM = <integer 0..999999>
Means that the job is assigned the number of cards specified by the user, providing it does not exceed the value specified for MAXIMUM.

START =
Assigns appropriate start options to job start requests.

START = *NO
This is a formal entry with no significance, except that it is required if the operand JOB-TYPE = DIALOG is specified.

START = *PARAMETERS(...)

DEFAULT =
This is the default value assumed if a user has not requested a specific start type in the ENTER-JOB command. The value defined for DEFAULT need not be listed under ALLOWED (see below).

DEFAULT = *SOON
The job should be started as soon as possible. If several jobs have requested SOON, the job priority determines which starts first.

DEFAULT = *WITHIN(...)
The job must be started within the time specified in hours and minutes.

HOURS = 0 / <integer 0..23>
Is a value between 0 and 23 hours.
The default value is 0 hours.

MINUTES = 00 / <integer 0..59>
Is a value between 0 and 59 minutes.
The default value is 0 minutes.

ALLOWED =
Defines those values which the user may specify for the job class concerned in the START operand of the ENTER-JOB command.

ALLOWED = *IMMEDIATELY
A job in this job class may be started immediately, even if it delays other jobs having higher priority that were supposed to be started at this time.

ALLOWED = *SOON
Same meaning as DEFAULT = *SOON, see above.

ALLOWED = *AT
The job may be started on the specified date and precisely at the specified time (hour, minutes) if possible.

ALLOWED = *LATEST
A job may be started at the latest by the specified date and time.

ALLOWED = *EARLIEST
A job may be started at the earliest at the specified date and time.

ALLOWED = *WITHIN
A job may be started within the specified time.

ALLOWED = *AT-STREAM-STARTUP / list poss(7)
A job may be started at the time the job scheduler is started.

Note

If an option is not specified under ALLOWED, it is not allowed unless the entry concerned is *SOON or *WITHIN, defined under DEFAULT as the default value.
A further exception is ALLOWED=*IMMEDIATELY:
If the user has specified PRIORITY=([p],EXPRESS) or START=*IMMEDIATELY in the LOGON, ENTER-JOB or MODIFY-JOB command, and EXPRESS is specified in his JOIN entry, the job will be started immediately, even if ALLOWED= *IMMEDIATELY is not specified in the user's job class.

REPEAT-JOB =
Controls the frequency of repeat jobs at specific time intervals. This is ignored if JOB-TYPE = *DIALOG is specified.

REPEAT-JOB = *NO
This is a formal entry with no significance, except that it is required if the operand JOB-TYPE = *DIALOG is specified.

REPEAT-JOB = *PARAMETERS(...)

DEFAULT =
This is the default value assumed if the user has made no entry for the frequency of job repetition in the ENTER-JOB or LOGON command, i.e. has omitted the REPEAT-JOB or REPEAT operand, or has specified REPEAT-JOB = STD or REPEAT = STD.

DEFAULT = *NO
Means that the job is not repeated.

DEFAULT = *AT-STREAM-STARTUP
Jobs belonging to this class are run again following each start of the job scheduler, provided the user has requested this in the LOGON or ENTER-JOB command.

DEFAULT = *WEEKLY
Jobs in this class are started weekly.
The exact starting time depends on the START operand value in the ENTER-JOB command.

DEFAULT = *DAILY
Jobs in this class are started daily.
The exact starting time depends on the START operand value in the ENTER-JOB command.

DEFAULT = *PERIOD(...)
Jobs are repeated each time the specified time interval has elapsed.

HOURS = 0 / <integer 0..23>
The time interval in hours may have a value between 0 and 23.

MINUTES = 00 / <integer 0..59>
A value between 0 and 59 minutes may be given.
The total time interval must be greater than 0. The exact start time depends on the START operand value in the ENTER-JOB command.

ALLOWED =
Specifies the values the user may give in the REPEAT-JOB or REPEAT operand of the ENTER-JOB or LOGON command.

ALLOWED = *NO
Repetition of jobs in this class is not possible, unless a value other than NO has been specified for DEFAULT.

ALLOWED = *AT-STREAM-STARTUP
Jobs may be repeated if required each time the job scheduler is started.

ALLOWED = *DAILY
Jobs in this class may be repeated daily.
The exact starting time depends on the START operand value in the ENTER-JOB command.

ALLOWED = *WEEKLY
Jobs in this class may be repeated weekly.
The exact starting time depends on the START operand value in the ENTER-JOB command.

ALLOWED = *PERIOD
Jobs can be repeated each time the specified time interval has elapsed.

JOB-PARAMETER =
Additional class attributes, evaluated by system exit 32.

JOB-PARAMETER = *NO
No additional class attributes are defined.

JOB-PARAMETER = <c-string 1..127>
This operand enables additional class attributes to be specified in 'string' in free syntax. Between 0 and 127 characters may be specified. Here, information specific to the computer system can be stored in each job class definition. System administration must create an exit routine to compare what the user specifies in the JOB-PARAMETER operand of the LOGON, ENTER-JOB or MODIFY-JOB command with what was specified in 'string' and to confirm its validity. The exit routine is called during the processing of these commands entered by the user.
The JOB-PARAMETER operand is not evaluated by the system.

Notes

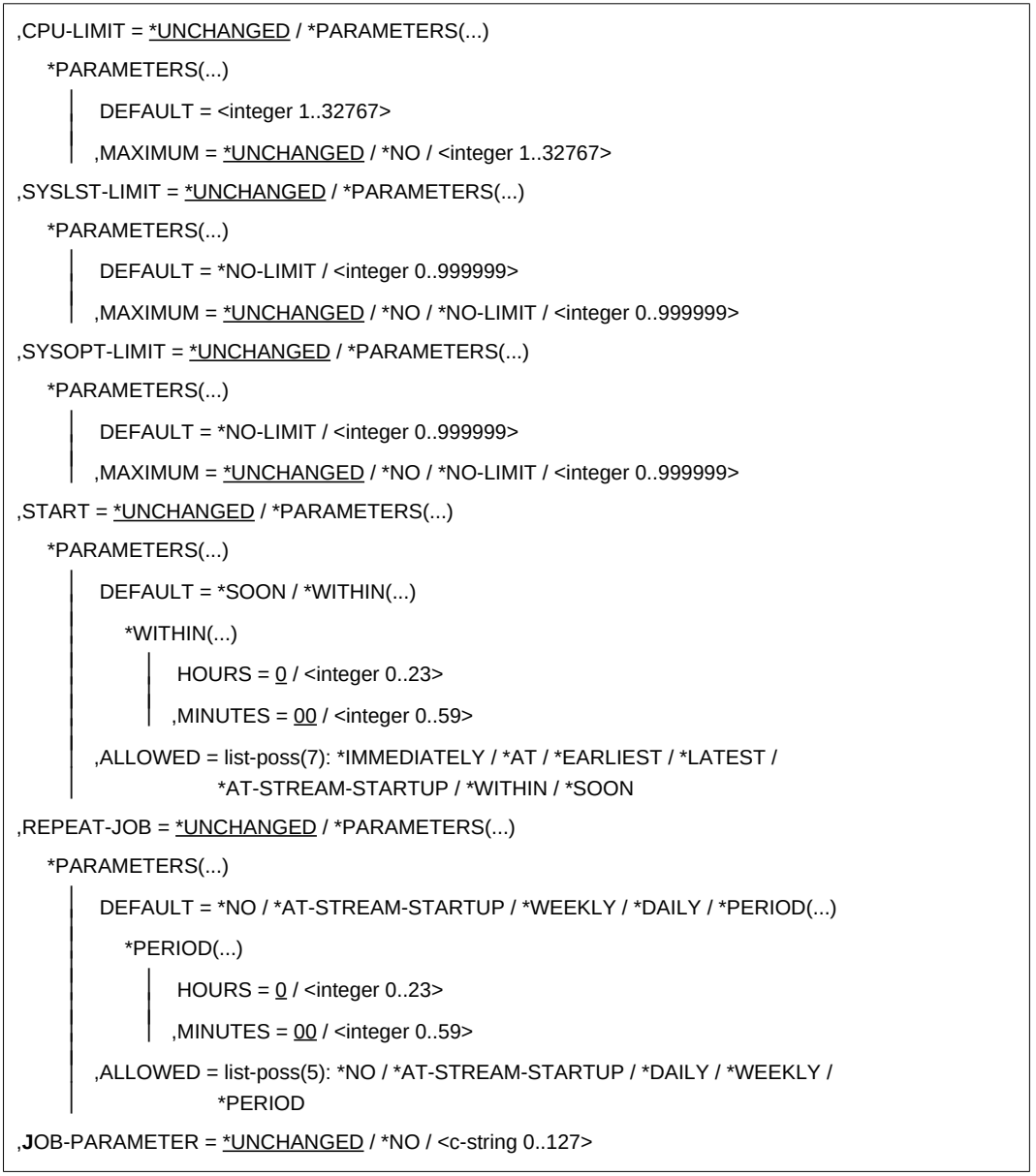
– The CLASS-LIMIT and CLASS-WEIGHT operands are evaluated by the operating system's class scheduler, which is independent of the job schedulers, in order to control the job-related portion of the system load (see "Commands, Volumes 1 - 3" [1], [2], [3]).
– The significance of the JOB-PRIORITY, START and JOB-PARAMETER operands depends on the job scheduling algorithm used by the stream, as defined in the STREAM operand (see "Commands, Volumes 1 - 3" [1], [2], [3]).
– The number of job classes is unlimited.

MODIFY-JOB-CLASS
Modify job class definitions

This statement is used to modify an existing job class definition in the SJMSFILE or the JMS database.

```
MODIFY-JOB-CLASS

 NAME = <name 1..8>
,STREAM = *UNCHANGED / *DEFAULT-STREAM / <name 1..8>
,CLASS-LIMIT = *UNCHANGED / <integer 0..4095>
,CLASS-WEIGHT = *UNCHANGED / <integer 1..9>
,CLASS-OPTIMUM = *UNCHANGED / <integer 0..4095>
,JOB-PRIORITY = *UNCHANGED / *PARAMETERS(...)
    *PARAMETERS(...)
        DEFAULT = <integer 1..9>
        ,MAXIMUM = *UNCHANGED / *NO / <integer 1..9>
,JOB-TYPE = *UNCHANGED / *BATCH / *DIALOG
,TP-ALLOWED = *UNCHANGED / *NO / *YES(...)
    *YES(...)
        CATEGORY = *TP / <name 1..7>
,DIALOG-ALLOWED = *UNCHANGED / *NO / *YES(...)
    *YES(...)
        CATEGORY = *DIALOG / <name 1..7>
,BATCH-ALLOWED = *UNCHANGED / *NO / *YES(...)
    *YES(...)
        CATEGORY = *BATCH / <name 1..7>
,START-ATTRIBUTE = *UNCHANGED / *BATCH / *DIALOG / *TP
,RUN-PRIORITY = *UNCHANGED / *PARAMETERS(...)
    *PARAMETERS(...)
        DEFAULT = <integer 30..255>
        ,MAXIMUM = *UNCHANGED / *NO / <integer 30..255>
,NO-CPU-LIMIT = *UNCHANGED / *NO / *YES
```

continued ➠


,CPU-LIMIT = *UNCHANGED / *PARAMETERS(...)
*PARAMETERS(...)
DEFAULT = <integer 1..32767>
,MAXIMUM = *UNCHANGED / *NO / <integer 1..32767>
,SYSLST-LIMIT = *UNCHANGED / *PARAMETERS(...)
*PARAMETERS(...)
DEFAULT = *NO-LIMIT / <integer 0..999999>
,MAXIMUM = *UNCHANGED / *NO / *NO-LIMIT / <integer 0..999999>
,SYSOPT-LIMIT = *UNCHANGED / *PARAMETERS(...)
*PARAMETERS(...)
DEFAULT = *NO-LIMIT / <integer 0..999999>
,MAXIMUM = *UNCHANGED / *NO / *NO-LIMIT / <integer 0..999999>
,START = *UNCHANGED / *PARAMETERS(...)
*PARAMETERS(...)
DEFAULT = *SOON / *WITHIN(...)
*WITHIN(...)
HOURS = 0 / <integer 0..23>
,MINUTES = 00 / <integer 0..59>
,ALLOWED = list-poss(7): *IMMEDIATELY / *AT / *EARLIEST / *LATEST /
*AT-STREAM-STARTUP / *WITHIN / *SOON
,REPEAT-JOB = *UNCHANGED / *PARAMETERS(...)
*PARAMETERS(...)
DEFAULT = *NO / *AT-STREAM-STARTUP / *WEEKLY / *DAILY / *PERIOD(...)
*PERIOD(...)
HOURS = 0 / <integer 0..23>
,MINUTES = 00 / <integer 0..59>
,ALLOWED = list-poss(5): *NO / *AT-STREAM-STARTUP / *DAILY / *WEEKLY /
*PERIOD
,JOB-PARAMETER = *UNCHANGED / *NO / <c-string 0..127>

For a description of the operands see the DEFINE-JOB-CLASS statement.

Note

Database updates are ignored for the following operands:

– STREAM
– JOB-TYPE
– START-ATTRIBUTE.

DELETE-JOB-CLASS
Delete class definitions

This statement can be used to delete an existing class definition from the SJMSFILE or the JMS database.

If the definition in the file is deleted, any jobs of the class in question that are in the job pool when the next system session starts are lost. If the definition is to be deleted in the system, its status is flagged as 'IN-DELETE'. The class definition is not removed from the database until no remaining jobs are assigned to the class. No new jobs are accepted for job classes flagged as 'IN-DELETE'.
Deletion of a default class is not permitted. If a default class is to be deleted, all access rights to it must first be rescinded.

DELETE-JOB-CLASS
NAME = <name 1..8>

NAME = <name 1..8>
Name, consisting of from 1 to 8 characters, of an existing class definition that is to be deleted.

GRANT-JOB-CLASS-ACCESS
Control access by user IDs to job class

This statement is used to control access by user IDs to a job class. If a job class is defined as the default class for a user (by means of the SET-JOB-CLASS-DEFAULT statement), it is not necessary to define access once again using the GRANT-JOB-CLASS-ACCESS statement.

If a new user ID is entered in the JOIN file during a session, it immediately receives access to all public classes.

```
GRANT-JOB-CLASS-ACCESS

 NAME = <name 1..8>
,ACTION = *ADD / *REMOVE
,USER = *ALL / list-poss(255): <name 1..8>
```

NAME = <name 1..8>
Name of a job class to which a user ID is to receive or be denied access.

ACTION =
Permits or denies access to a job class.

ACTION = *ADD
The user ID specified under USER= may access the job class.

ACTION = *REMOVE
The user ID specified under USER= is denied access to the job class specified for NAME =.

USER =
Controls access to a job class.

USER = *ALL
Access to the job class specified in NAME = is to be defined for all user IDs.

USER = list-poss(255)
Access is granted for the user IDs given in this list.

USER = <name 1..8>
Access is granted to this one user ID. The name consists of from 1 to 8 characters.
No check is run to ascertain whether the specified user ID is entered in the JOIN file unless a modification is made in the current session. All IDs are entered in the SJMSFILE.

SET-JOB-CLASS-DEFAULT
Specifies default classes for users

This statement is used to specify default classes for users. At the same time, the users are granted access.

If no public default class has been specified for a job type, $SYSJC is the system default class, which allows all privileges.
Since only privileged users have access to $SYSJC, JMU issues a warning message.

When a new user is entered in the JOIN file during a session, he or she receives immediate access to all system default classes.

```
SET-JOB-CLASS-DEFAULT

 NAME = <name 1..8>
,ACTION = *ADD / *REMOVE
,USER = *ALL / list-poss(255): <name 1..8>
```

For the meaning of the operands see the GRANT-JOB-CLASS-ACCESS statement.

Note

Changing the default class always consists of the two statements:
SET-JOB-CLASS-DEFAULT default-job-class.old,*REMOVE,*ALL
SET-JOB-CLASS-DEFAULT default-job-class.new,*ADD,*ALL.
It is not advisable to change the default class during the current session, because $SYSJC is entered as the default class between the two statements.

SET-MODIFICATION-MODE
Sets modification mode

This statement can be used to set a modification mode in the JMU utility routine that permits the following modifications to be made to JMS files in the current session:

– modify rights of access to job classes
– assign default job classes to new users
– create, delete and modify job class definitions

If desired, these modifications can be implemented with immediate effect.

Note

The SET-MODIFICATION-MODE statement is permitted only when called from the JMU and the caller possesses the TSOS privilege.

SET-MODIFICATION-MODE
SCOPE = *FILE / *SYSTEM / *ALL

SCOPE =
Specifies whether the modifications are to be implemented in the file, the database or both.

SCOPE =*FILE
The modifications are to be implemented in the file only, i.e. they will be effective only as of the next startup.

SCOPE=*SYSTEM
The modifications are to be implemented in the database only, i.e. they are effective only for the current session (for test purposes).

SCOPE=*ALL
The modifications are to be implemented in the file and in the database.

Notes

– If SCOPE = *SYSTEM / *ALL is specified, the STREAM, JOB-TYPE and START-ATTRIBUTE operands of the MODIFY-JOB-CLASS statement are ignored.
– Only the category names already defined can be used for assigning categories.
– If job classes are deleted in the current system, a class containing jobs that are still active is flagged JOB-CLASS-IN-DELETE. Jobs already accepted can be completed, but no new jobs are accepted.
– SHOW-JOB-CLASS as a JMU statement outputs only the contents of SJMSFILE. If SCOPE = *SYSTEM, the SHOW-JOB-CLASS statement is not executed.
– If SCOPE = *ALL is specified, a modification is made only if it is possible both in the SJMSFILE and in the database.

SHOW-JOB-STREAM
List contents of stream definitions or names of streams

This statement is used to list the contents of stream definitions or the stream names themselves.

SHOW-JOB-STREAM
NAME = *ALL / *ALL-NAMES / list-poss(255): <name 1..8> ,OUTPUT = *SYSOUT / *SYSLST

NAME =
Name of the job stream to be listed.

NAME =*ALL
All stream definitions are to be listed.

NAME = *ALL-NAMES
All stream names are to be listed.

NAME = <list-poss(255): <name 1..8>
All stream definitions whose names are specified in this list are to be listed. A maximum of 255 names may be given.

OUTPUT =
Defines the output destination.

OUTPUT = *SYSOUT
The stream definition(s) or stream names are to be output to SYSOUT.

OUTPUT = *SYSLST
Output is to be via SYSLST.

Example

```
REQUESTED DETAILS OF JOB STREAM: JSSTD
NAME.........:JSSTD
FILE.........:SYSENT.JOBSCHED.112
RUN PRIORITY..:125
DEFAULT.......:NO
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIORITY=Y,CPU-TIME=Y,WAIT-TIME=Y,JOB-QUOTA=50,LOGGING=NO


REQUESTED DETAILS OF JOB STREAM: JSSTD1
NAME.........:JSSTD1
FILE.........:SYSENT.JOBSCHED.112
RUN PRIORITY..:130
DEFAULT.......:YES
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIORITY=Y,CPU-TIME=Y,WAIT-TIME=Y,JOB-QUOTA=30,LOGGING=NO

REQUESTED DETAILS OF JOB STREAM: JSSTD2
NAME.........:JSSTD2
FILE.........:SYSENT.JOBSCHED.112
RUN PRIORITY..:150
DEFAULT.......:NO
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIO=Y,CPU-TIME=Y,WAIT-TIME=N,JOB-QUOTA=20,LOGGING=NO

REQUESTED DETAILS OF JOB STREAM: JSTSOS
NAME.........:JSTSOS
FILE.........:SYSENT.JOBSCHED.112
RUN PRIORITY..:120
DEFAULT.......:NO
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIORITY=Y,CPU-TIME=Y,WAIT-TIME=Y,JOB-QUOTA=50,LOGGING=NO
```

SHOW-JOB-CLASS
List contents of class definitions or names of classes

This statement enables the contents of class definitions or the names of classes to be listed. The listing of a class definition includes the names of all users having access to that class.

SHOW-JOB-CLASS
NAME = *ALL / *ALL-NAMES / list-poss(255): <name 1..8> ,OUTPUT = *SYSOUT / *SYSLST

NAME =
Name of the class to be listed.

NAME = *ALL
All class definitions are to be listed.

NAME = *ALL-NAMES
All names of classes are to be listed (without the contents of the class definitions).

NAME = list-poss(255): <name 1..8>
All class definitions whose names are specified in this list are to be listed. A maximum of 255 names may be given.

OUTPUT =
Defines the output destination.

OUTPUT = *SYSOUT
Output is to be via SYSOUT.

OUTPUT = *SYSLST
Output is to be via SYSLST.

Example

```
REQUESTED DETAILS OF JOB CLASS: JCBATCHF
NAME..........:JCBATCHF
STREAM........:JSSTD
CLASS LIMIT...:255
CLASS OPTIMUM.:0
WEIGHT........:3
JOB PRIORITY..:DEFAULT=3            MAXIMUM= 1
JOB ATTRIBUTES:JOBTYPE=BATCH        ST-ATTR= BATCH
TP ALLOWED....:TP
DIALOG ALLOWED:NO
BATCH ALLOWED.:BATCH
RUN PRIORITY..:DEFAULT=180          MAXIMUM= 30
NO CPU LIMIT..:YES
CPU LIMIT.....:DEFAULT=32767        MAXIMUM= 32767
SYSLST LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
SYSOPT LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
START.........:DEFAULT=SOON         ALLOWED= SOON EARLY AT LATE IN IMM STUP
REPEAT JOB....:DEFAULT=NO           ALLOWED= NO STUP DAILY WEEKLY PERIOD
JOB PARAMETER.:UNDEFINED

JCBATCHF IS AVAILABLE TO:
ALL USERS

JCBATCHF IS A DEFAULT FOR:
NO USERS


REQUESTED DETAILS OF JOB CLASS: JCBSTD
NAME..........:JCBSTD
STREAM........:JSSTD1
CLASS LIMIT...:50
CLASS OPTIMUM.:0
WEIGHT........:8
JOB PRIORITY..:DEFAULT=9            MAXIMUM= 1
JOB ATTRIBUTES:JOBTYPE=BATCH        ST-ATTR= BATCH
TP ALLOWED....:TP
DIALOG ALLOWED:NO
BATCH ALLOWED.:BATCH
RUN PRIORITY..:DEFAULT=220          MAXIMUM= 180
NO CPU LIMIT..:YES
CPU LIMIT.....:DEFAULT=32000        MAXIMUM= 32767
SYSLST LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
SYSOPT LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
START.........:DEFAULT=SOON         ALLOWED= SOON EARLY AT LATE IN IMM
REPEAT JOB....:DEFAULT=NO           ALLOWED= NO STUP DAILY WEEKLY PERIOD
JOB PARAMETER.:UNDEFINED

JCBSTD   IS AVAILABLE TO:
ALL USERS

JCBSTD   IS A SYSTEM DEFAULT
```

REMOVE-USER
Prohibit access to private job classes

The REMOVE-USER statement is used to deny specified user IDs access to all private job classes. The statement is an extension of the system command of the same name which is used to delete user entries in the JOIN file.
However, the statement cannot be used to prevent special users accessing public job classes or system default classes.

REMOVE-USER
USER-IDENTIFICATION = *ALL / list-poss(8): <name 1..8>

USER-IDENTIFICATION =
Specifies the user IDs to be removed.

USER-IDENTIFICATION = *ALL
This causes all access lists to be deleted, thus locking all private job classes. Any ID-specific default job classes are reset to the system default settings.

USER-IDENTIFICATION = list-poss(8): <name 1..8>
The specified user ID(s) is (are) to be prevented from accessing all private job classes. Any default job classes specific to these IDs are reset. Up to 8 user IDs can be specified.

END
Terminate statement input

The END statement terminates input of statements to the JMU routine.

END

The program is to be terminated.

6 LMSCONV
Creating and administering libraries

Version: **LMSCONV V1.0B**

LMSCONV (Library Maintenance System Converter) is a routine for converting old library formats (MLU and LMR) to the new format (PLAM), and from K-ISAM to NK-ISAM format. It is also used for creating and maintaining libraries and for processing the library elements (also known as "members"). LMSCONV helps the user make the transition from MLU, LMR and COBLUR to LMS (on LMS see the "LMS" manual [11]).

LMSCONV performs the following functions:

- create libraries
- add elements to a library
- output library elements to files
- copy elements to other libraries
- list elements
- delete elements
- update elements
- rename elements
- output a library’s table of contents.

LMSCONV handles the following library formats:

- **program libraries** for storing macros, object and load modules, lists, etc. These libraries are processed by the PLAM access method.
- **source program libraries** for storing source programs
- **object module libraries** for storing object modules
- **macro libraries** for storing macros.

The following diagram illustrates the input/output facilities of LMSCONV:

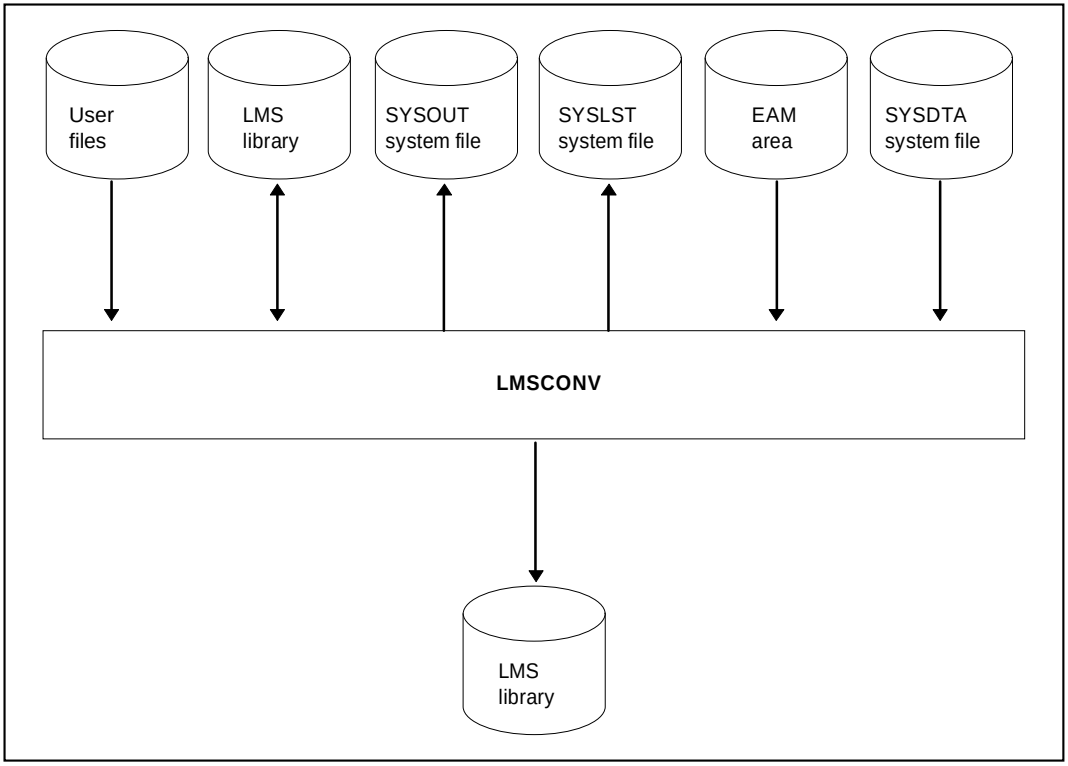


Figure 3: LMSCONV access facilities

The types of library elements accepted by LMSCONV include files, object modules from the EAM area, and elements from other libraries. Object and load modules and macros are separated by element type. The elements are referenced individually in the library by means of their element identifiers.

Example of an LMSCONV run

```
/START-PROGRAM FROM-FILE=$LMSCONV
%  BLS0500 PROGRAM 'LMSCONV', VERSION 'V01.0B' OF '1992-07-20' LOADED.
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991.
   ALL RIGHTS RESERVED.
%  LMC0310 LMSCONV VERSION V01.0B43 STARTED
CTL=(CMD)                                              PRT=(OUT) ----------  (1)

$LIB FILE=UEB.BIBL,BOTH,NEW ---------------------------------------------------  (2)

$ADDM A.QUELL.A ----------------------------------------------------------------  (3)

$PAR LOG=MED -------------------------------------------------------------------  (4)

$ADDM A.BEISPIEL>BSP -----------------------------------------------------------  (5)

INPUT  FILE
OUTPUT LIBRARY= :X:$RUES.UEB.BIBL,DEV=DISK
         ADD A.BEISPIEL AS (M)BSP/@(0001)/1990-11-19 ----------------------  (6)

$TOC* * ------------------------------------------------------------------------  (7)

INPUT  LIBRARY= :X:$RUES.UEB.BIBL,DEV=DIS
TYP NAME      VER (VAR#) DATE         NAME      VER (VAR#) DATE
(M) A.QUELL.A @   (0001) 1990-11-19   BSP . . . @   (0001) 1990-11-19    2
(M)-ELEMENT(S) IN THIS TABLE OF CONTENTS ---------------------------------  (8)

$END
%  LMS0311 LMS V01.0B43   TERMINATED NORMALLY -----------------------------  (9)
```

(1) LMSCONV is called.

(2) LMSCONV creates a new program library called UEB.BIBL, and assigns it as an input/output library.

(3) File A.QUELL.A is added to the library as a type M element with the element name A.QUELL.A.

(4) The processing operand LOG=MED means that LMSCONV outputs positive acknowledgments in addition to error messages.

(5) File A.BEISPIEL is added to the library as a type M element with the element name BSP.

(6) Positive acknowledgment: as the LOG=MED processing operand is set, LMSCONV confirms that file A.BEISPIEL has been included in the library as element BSP.

(7) The table of contents of program library UEB.BIBL is to be listed.

(8) TOC entry for program library UEB.BIBL.

(9) LMSCONV is terminated.

LMSCONV in interactive and batch mode

The LMSCONV utility is started with the command

```
START-PROGRAM FROM-FILE=$LMSCONV
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are also available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in "Commands, Volume 3" [3].

The END statement terminates the LMSCONV routine.

LMSCONV runs in both interactive and batch modes. In interactive mode, LMSCONV reads input statements by default from the terminal.

Before execution of LMSCONV, switch 1 (MODIFY-JOB-SWITCHES ON=1) must be set if LMSCONV has been called in a procedure and the LMSCONV statements are also to be read from the procedure. Otherwise, LMSCONV will still be called, but will expect the statements from the terminal.

In batch mode, LMSCONV reads the statements from the system file SYSDTA.

If a library is still closed, LMSCONV outputs the following message during batch operation:

```
FILE (ELEMENT or TYPE) IS LOCKED.NEXT ATTEMPT AFTER 6 SECONDS!
```

It then automatically goes on to the next statement.

The LMSCONV log is output to system file SYSOUT (i.e. the terminal, in interactive mode) or to the medium defined as PRT (system file SYSLST or library element). If LMSCONV is to output positive acknowledgments in addition to error messages, the processing operand PAR LOG=MED must be set.

6.1 Libraries

A library is a file with a particular substructure. It contains elements ("members") and a table of contents (TOC).

An element is a logically coherent set of data such as a file, an object module etc. Each element in the library can be referenced individually.

Storing files as elements in a library saves space in the system catalog, as only the library entry will appear in the system catalog. Space is also saved because there is just one default space allocation per library, and the elements are stored in the library in compressed form.

Storing object modules from the EAM area in a library eliminates the need to recompile the source programs every time.

Processing program libraries with LMSCONV offers a number of advantages over a source program, macro or object module library:

– Program libraries contain different types of elements, i.e. a program library allows a variety of elements for a particular application, such as object and load modules, to be concentrated in one place.

– Program libraries identify elements not only by their element names, but by the version identifier and type of element.

– A number of users can have simultaneous write access to program libraries.

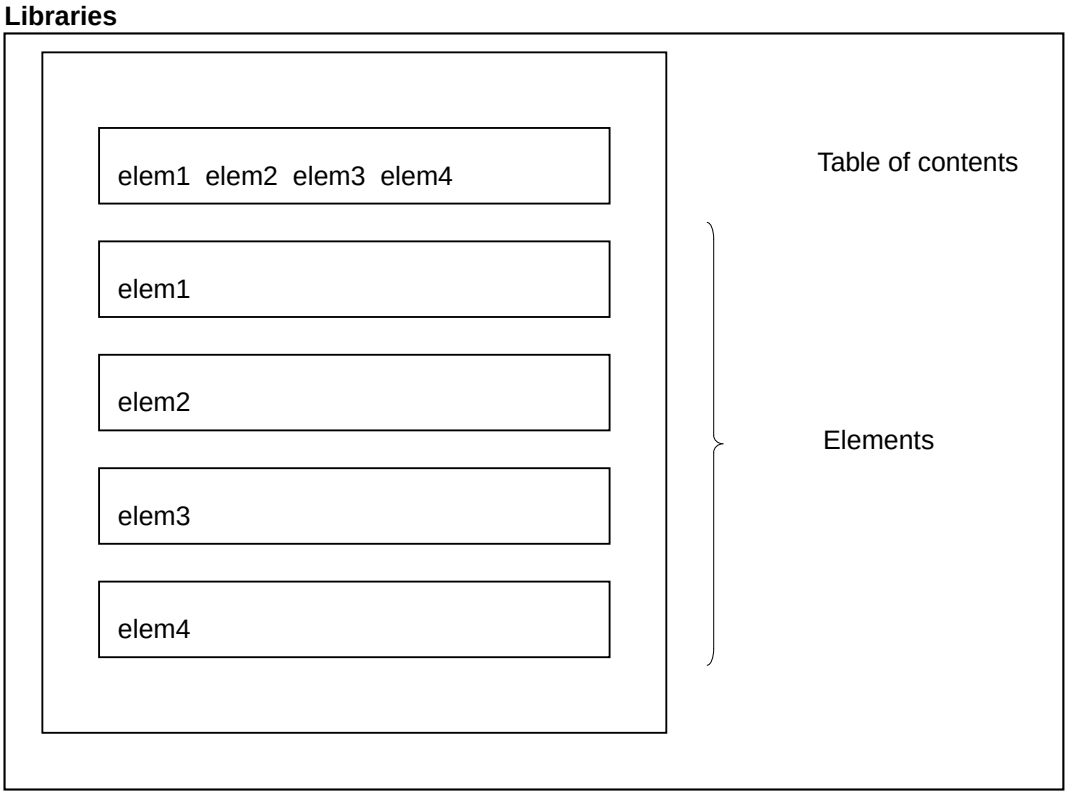


Figure 4: Structure of a library

Every library has an entry in the system catalog. As with any other BS2000 file, the user can define the name and other file attributes, such as the retention period and protection attributes.
LMSCONV opens an output library for reading and writing. An input library can be opened for writing with the DEL and NAM statements only (see section “Statements” on page 184). In all other cases it is read-only.

Types of libraries

LMSCONV processes various types of libraries:

– program libraries
– source program libraries
– macro libraries
– object module libraries
– sequential libraries

6.1.1 Program libraries (PLs)

Program libraries are PAM files processed by the PLAM library access method. For this reason, they are also called PLAM libraries.

These have some significant advantages over other library formats, including the following:

– synonymous elements can exist, differentiated by their type or version identifier

– several users can have simultaneous write access to the library.

There are two PLAM library formats. In addition to the existing NK2 format (NK2 PLAM file) this latest version supports NK2 disks with a minimum allocation unit of 8 or 64 Kbytes (abbreviated form: NK2(8K,64K)).
LMSCONV supports both library formats.
The user specifies the desired format by means of the command ADD-FILE-LINK ...,BUFFER-LENGTH=*STD(n). n can be either 1 or 2. The COPYLIB statement enables the user to convert from one format to the other; the ADD and SEL statements are available for both formats (see page 158f).

Multiple element types in one library

Program libraries may contain any element type supported by LMSCONV.

The element type dictates how the content of the element is to be interpreted by LMSCONV, and the type of storage unit to which it belongs:

Type	Content of element
S	Source programs 1)
M	Macros
R	Oject modules
C	Load modules
H	Compiler result information 2)
J	Procedures
P	Print data
D	Text data
X	Data in any format
L	LLM 2)
F	IFG format masks 2)
U	IFG-USER-PROFILE 2)

1) PRT elements are stored as type S in source program libraries.
2) Cannot be generated by LMSCONV.

Multiple versions for each element type and name

An element in a program library is uniquely identified by its type, name and version identifier.

If the user fails to specify a version identifier, LMSCONV automatically takes the following actions:

– When reading
  LMSCONV looks for the element of that name having the highest version identifier. The date is not taken into consideration.

– When writing
  The way the version is handled depends on the statement:

  ADD, PRT
  The element is created or overwritten with the highest possible version number X'FF'. LMSCONV represents this version with an @.

  DUP, NAM, UPD
  The output element is given the version identifier of the input element.

  If this causes an element of the same name to be overwritten, the internal variant number is incremented by 1. It acts as a write access count.

Note

A target version cannot be assigned.

Multiple access to program libraries

A library can be opened for read and write access by one or more users.

An element can be read simultaneously by several users, but can only be written by one. While an element is open for writing, the element cannot be accessed by any other user, even for reading, but any other element in the library can be accessed.

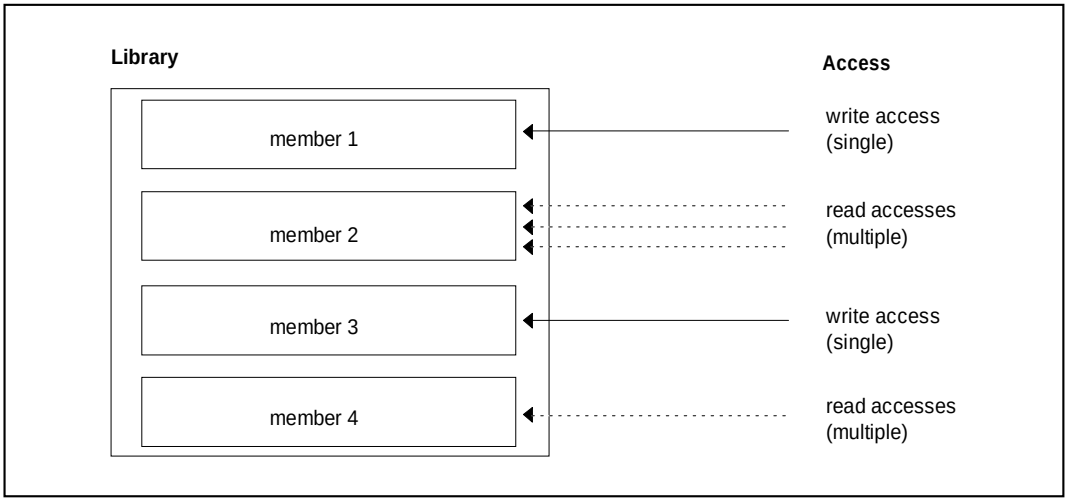


Figure 5: Multiple access to elements

Multiple access to a library means that an element may exist when the table of contents of the library are listed, but has ceased to exist when next accessed because another user has deleted it in the meantime.
Thus a table of contents list (see TOC) only gives a snapshot of the current state of the input library.

Restricting multiple access

LMSCONV always opens a program library with SHARED-UPDATE=*YES. An ADD-FILE-LINK command with SHARED-UPDATE=*NO issued for this library has no effect. The user can, however, restrict multiple access by means of the following commands:

`/SECURE-RESOURCE-ALLOCATION`

No other user can access the library for the duration of the task in which the command is issued.

`/MODIFY-FILE-ATTRIBUTES`

This command restricts multiple access as required by allocating write and read passwords (WRITE-PASSWORD, READ-PASSWORD), or by specifying the operand
...,PROTECTION=*PAR(USER-ACCESS=*OWNER-ONLY) or
...,PROTECTION=*PAR(ACCESS=*READ).
For a description of these commands see "Commands, Volumes 1 - 3" [1], [2], [3].

6.1.2 Single-type libraries

The following libraries can only hold one element type:

– source program libraries
– macro libraries
– object module libraries

Unlike program libraries, these types of library cannot hold multiple elements with the same name.

LMSCONV stores no file protection attributes for libraries or elements. If a retention period is specified in the ADD-FILE-LINK command when a library is created, LMSCONV places it in the catalog entry for the library.

Parallel access to source program, macro and object module libraries is permitted. They can be read simultaneously by different tasks.

LMSCONV opens a library for reading and writing (OPEN=INOUT) if it was assigned as a default output library for the LMSCONV run (LIB ...,USAGE=OUT) or was specified in DEL, NAM or PRT. The other statements open libraries for reading only (OPEN=INPUT).

Source program libraries/macro libraries (OSMs)

Source program libraries are ISAM files (KEY-POS=5,KEY-LEN=8) and recognize element type S only. Source programs and listings are stored as type S elements in source program libraries.
Macro libraries are ISAM files (KEY-POS=5,KEY-LEN=8) and recognize element type M only.

Object module libraries (OMLs)

Object module libraries (OMLs) are files in PAM format. They hold the object modules generated by language processors (compilers, assemblers) from the EAM area as type R elements.

OMLs can hold up to 3380 modules. The number of CSECTs/ENTRYs/COMMONs is limited (between 380 and 800 depending on the use of the library).
The size of an object module library cannot exceed 32500 PAM pages.

6.1.3 Sequential libraries (archive libraries)

Sequential libraries are held on magnetic tape. These libraries are tape files with standard labels and a block size of 2048 bytes.

LMSCONV processes sequential libraries by means of the BTAM access method.

A sequential library may contain phases (load modules), object modules, macros and source programs (also procedures and other text). Elements of the same type are combined in one library section. The order of sections is mandatory:
phases, object modules, macros, source programs.

The maximum record length for object modules, macros and source programs is 80 bytes.

The table of contents for sequential libraries is different from that of other libraries:
each element is preceded by a block containing the element identifier.

A sequential library can contain several elements of the same type and the same name. If the version number and/or the date are different, LMSCONV can read them individually.

The following restrictions apply to sequential libraries:

– They cannot be processed by DEL and NAM (see section 6.5, "Statements").
– List elements cannot be held in tape libraries because they have a record length > 80 characters.
  Records longer than 80 bytes are truncated.
– BS2000 does not support continuation tapes.

LMSCONV does not enter any file protection attributes for libraries or elements. If a retention period is specified in the ADD-FILE-LINK command, LMSCONV places it in the file header label (HDR1) when a library is created.

Rules for element names in sequential libraries

elementname — Maximum 8 characters

Character set:

Letters	: A-Z
Special characters	: $, #, @, &, %, -, _
Digits	: 0-9

The first character must be a letter or one of the special characters $, # or @.

version — Three-character version identifier

Character set:

Letters	: A-Z
Special characters	:
Digits	: 0-9

Letters may be used only as the first character.

Specifying the date in the element identifier

The user date maintained for each element can also be used to select the element. The user date is optional.
If the user date is not specified in a statement, LMSCONV inserts the current date by default (or the date of the source element when an element is copied or renamed). If DATE is specified, the current date is inserted instead of the user date.

Rules for specifying the date for sequential libraries:

YYMMDD — *Meaning:*

YY	: Year
MM:	: Month
DD	: Day

LMSCONV adds a reference year to make the four-digit century:

YY < 60: LMSCONV fills to 20YY.
YY ≥ 60: LMSCONV fills to 19YY.

Use of versions

The update function UPD increments the variant number of the updated element by 1.

6.2 Elements

6.2.1 Contents of a program library

LMSCONV handles the following element types in program libraries:

Type	Element content
S	Source programs
M	Macros
R	Object modules
F	IFG format masks
U	IFG user profile
C	Load modules
H	Compiler result information
J	Procedures
P	Print data
D	Text data
X	Data in any format
L	LLM

Using an asterisk (*) as the element type in DUP and TOC allows all element types stored in the library to be referenced. To list all the elements in a program library, either TOC or TOC* */* must be specified.

Description of the element types

Element type S - source programs

Source programs in libraries are used as input to compilers and to the assembler in compilation and assembly runs.

Element type M - macros

The assembler takes the macro elements referenced in the program from the library that has been assigned.

Element type R - object modules

Object modules generated by compilers and by the assembler are normally placed in the temporary EAM area. LMSCONV can store these object modules as type R elements in a program library. Object modules generated by compilers and by the assembler can also be stored directly in a program library.

Element type F - IFG format masks

In future, elements of this type will be generated by IFG and stored in program libraries. This element type cannot be generated by LMSCONV, however.

Element type U - IFG user profile

In future, elements of this type will be generated by IFG and stored in program libraries. This element type cannot be generated by LMSCONV, however.

Element type C - load modules

A load module created by the linkage editor TSOSLNK is normally stored in a file. This file can be stored by LMSCONV in a program library as a type C element. Load modules created by the linkage editor can optionally also be stored directly in a program library.

Element type H - compiler result information

Elements of this type are created by compilers and by the assembler, and stored in program libraries. You will be able to find more details in the relevant user manuals. This type of element cannot, however, be created by LMSCONV.

Element type J - procedures

This type of element is used for BS2000 procedures and LMSCONV statements.

Note

Task switch 1 must be set when LMSCONV is called in a DO or CALL procedure and the statements are to be read from SYSDTA.

Element type P - list elements

Data edited for printing is known as list elements. The first character of the record must be a valid feed character for print control.
Elements of this type can be created with ADDP, DUPP and PRT (see section “Statements” on page 184).

List elements are printed via LST, utilizing the print control character when PRT (LST) was previously specified.

Element type D - text data

Any type of text can be written to elements of type D. The same functions are possible as for element type S. This type of element cannot, however, be created by LMSCONV.

Element type X - data in any format

Data of any format can be written to type X elements.

Element type L - LLMs

The linkage editor BINDER stores the link and load modules it creates in type L elements. LMSCONV cannot create elements of this type.

Element identifier

Elements in program libraries are referenced by means of their element type and element identifier.

The element identifier consists of the name, version and date, and is specified in the following form:

```
elementname[/version[/date]]
```

or

```
elementname[//date]
```

Specification of version and date is optional on input, but **on output, the version must not be specified**. If no version is specified in a statement, the element with the highest version is selected by default. If no date is specified in a statement, by default today's date is assumed.

The element identifier is specified as an operand:

operation x	elem

operation	Statement name.
x	Element type.
elem	Element identifier, consisting of elementname and optionally version and date.

Rules for element identifiers in program libraries

elementname

Up to 64 characters.

Character set:

Letters	: A-Z
Special characters	: $, #, @, . (period), - (hyphen), _ (underscore).
Digits	: 0-9

The hyphen, underscore and period characters must not be used as the first or last character, nor may two of these special characters appear consecutively.
The hyphen must not appear immediately to the right of a $, @, #, underscore or period. The element name must contain at least one letter or special character @, # or $.

version

Maximum 24 characters

Character set:

Letters	: A-Z
Special characters	: . (period), - (hyphen), @ (comercial at)
Digits	: 0-9

The special characters period and hyphen must not be used as the first or last character. The same special character must not appear in consecutive positions.
The hyphen must not appear to the immediate right of a period.

If the @ character is specified explicitly, no other character may be specified in the version identifier.

On output (write access) no version may be specified.

The target element contains either:
– version number X'FF' if the source is a file or *EAM and the output library is a PLAM library (version number X'FF' is the highest possible version and is represented as "@"), or
– the version number of the source element.

Specifying the date in the element identifier

The user date maintained for each element can be used for selecting elements. The user date is optional.
When an element is copied or renamed, the date of the original element is transferred if a new date is not explicitly specified.
If DATE is specified, the current date is inserted instead of the user date.
When an element is created, LMSCONV inserts the current date by default if no other date is specified explicitly.

Rules for specifying date for program libraries:

YYMMDD *Meaning:*

YYYY :	Year
MM :	Month
DD :	Day

Use of the version and variant number

With source program, macro and object module libraries, the version is incremented automatically when update and numbering functions are performed. This is not the case with program libraries.

Instead, a variant number (max. 4 numeric characters) is maintained that fulfils the function of a write access count.

Thus with program libraries, no version incrementation takes place, but a variant for the element is incremented by 1 for each write access to the element. This means that for ADD, NAM and UPD the variant is incremented whenever an element of the same type, name and version is overwritten.

Listing the element identifier

Element identifiers are listed as follows:

```
(type)elementname/version[(variantnumber)]
```

The variant number is only maintained for elements in program libraries. It is set to (0001) by default and incremented by 1 by the ADD, NAM and UPD statements.

With TOC, the date is also output:

```
(type)elementname/version[(variantnumber)]/date.
```

Assigning a character set to a PLAM library element (XHCS: eXtended Host Code Support)

A character set can be assigned to each element in a PLAM library by allocating a CCSN (Coded Character Set Name).
This character set is passed to interfaces and taken into account for outputs. If XCHS is not supported at a given interface, the ’no code’ default is always used.
LMSCONV itself does not require a particular character set, nor does it evaluate the default of the user ID.
A character set can be explicitly assigned or an existing assignment modified by means of SDF statements in the LMS utility routine.

LMSCONV offers implicit support:

A character set is implicitly assigned to an element when a file is added to the library by means of the ADD statement, by appending the catalog attribute CCS (coded character set). In order to avoid inconsistencies, the coded character set name (CCSN) is not saved in the attribute record (record type 164). When a module from the * file is added to the library, its attribute is ’no code’.
When a library element is output to a file, the corresponding coded character set name (CCSN) of the element is assigned to the file. (When an element is copied, the CCSN of the source element is assigned to the target element.) The coded character set (CCS) assigned to the element in question is used when records from a library element are output to SYSOUT (even in edited form).
If SYSOUT is assigned to a file, the user must also assign the desired coded character set to the file by means of the MODIFY-FILE-ATTRIBUTES command.
No CCSN is evaluated for output to SYSLST. If SYSLST is assigned to a file, the user can impose the desired character set on the output by assigning the CCS in question to the file by means of the MODIFY-FILE-ATTRIBUTES command.
If output was redirected to a library element by means of the LMSCONV command PRT, this element contains the value ’no code’ for CCSN.
’no code’ is always assumed as the value for CCSN when tables of contents and other items of element information created by LMSCONV are output.
If PAR OVERWRITE = EXTEND is specified, LMSCONV checks the CCSNs of source and target. If the two CCSNs do not tally, the job is rejected and an error message issued.

NK4 disks

There are two formats for PLAM libraries:

– the original NK2 format (NK2 PLAM file)
– the new NK2 format with a minimum allocation unit of 8 or 64 Kbytes (NK4 PLAM file).

LMSCONV supports both library formats.

The COPYLIB statement provides the means of conversion from one format to the other. The user specifies the desired format by means of the command ADD-FILE-LINK ...,BUFFER-LENGTH=*STD(1) or ...,BUFFER-LENGTH=*STD(2).
NK4 disks are also supported by the ADD and SEL statements.

Adding files with ADD

ADD can be used to add files of any BUFFER-LENGTH to a PLAM library.

File output with SEL

A distinction must be drawn between the following cases:

1.) The element contains an attribute record with the original BUFFER-LENGTH specification (i.e. after ADD with PAR KEY=YES or for original UPAM files or PLAM library files)

1.1) BUFFER-LENGTH is explicitly specified for the target file (either by ADD-FILE-LINK in the TFT or directly in the catalog):
The default (BUFFER-LENGTH) is always used, which means that the following problems may occur:

– SAM / ISAM files:
If the element records are too long for the specified BUFFER-LENGTH, a DMS error results.

– UPAM files:
When UPAM files are created, LMSCONV packs a logical block (other than the last block) with 2K units and outputs this block only with UPAM.
If BLK-CONTR=DATA is specified, each logical block (BUFFER-LENGTH) starts with a 12-byte check field (CF). If the specified BUFFER-LENGTH does not tally with that stored, data from DMS may be overwritten with the check field, thus rendering the file unusable.
If BLK-CONTR=NO is specified, unusable files may be generated if the BUFFER-LENGTH is changed (e.g. PLAM files).
On account of these possibilities for error, LMSCONV always issues a warning if there is any discrepancy in the BUFFER-LENGTHs (between user specification and stored value).
The system always attempts to create the file nevertheless.

1.2) No explicit BUFFER-LENGTH is specified or known for the target file. In this case, the value from the attribute record is used.

If n in STD(n) is odd, LMS increments to n+1.

2.) The element does not contain an attribute record (e.g. for phase elements).

2.1) BUFFER-LENGTH is specified explicitly for the target file: → same procedure as in 1.1. above.

When phases are generated, BUFFER-LENGTH specifications ≠ STD(1) or STD(2) produce errors.

2.2) BUFFER-LENGTH for the target file not explicitly specified or unknown.

- Phases:
  BUFFER-LENGTH is derived from the current environment, i.e on NK2 disks:
  BUFFER-LENGTH = STD(1)
  on NK4 disks: BUFFER-LENGTH = STD(2).
  As regards content, there is no difference between the phases.
- In all other instances, BUFFER-LENGTH is calculated on the basis of the maximum record length.

3.) In brief, it is advisable to proceed as follows when taking files from an NK2 disk to an NK4 disk via a library:

- Extract all "critical" elements in the library as files. "Critical" elements are PAM elements under type X which as files:
  - have BUFFER-LENGTH = STD(n) and n odd
  - are phases with PAM keys
  - are NK2-oriented PLAM files.
- Convert all files with BUFFER-LENGTH odd (except PLAM files) to NK4 format with PAMCONV.
- Convert all phases with PAM keys into NK phases with PAMCONV.
- Convert NK2 PLAM files into NK4 PLAM files by means of the LMSCONV statement COPYLIB.
- After conversion, use ADD to add the NK4 files to an NK4 PLAM file and transfer this file to the NK4 disk.

6.2.2 Contents of single-type libraries

Single-type libraries contain just one element type.

Type	Content of elements	Library
S	Listings, procedures, source programs and text data	Source program library
M	Macros	Macro library
R	Object modules	Object module library

Description of element types

Element type S

This element type contains listings, procedures, source programs and text data.

If a listing is to be output by LST using print control characters, PRT (LST) and the processing operand PAR FORMAT=P must first be specified.

Element type M

The assembler extracts the macro elements referenced in the program from the assigned macro library.

Element type R

The object modules generated by the compilers are stored in the object module library, and are used as input to the linkage editors TSOSLNK and BINDER and to the dynamic binder loader DBL.

Element identifier

The element identifier enables each element in a library to be individually addressed. The element identifier consists of the name, version and date, and is specified in the following form:

```
elementname[/version[/date]]
```

or

```
elementname[//date]
```

Specification of the version and date is optional. If no version is specified in a statement, the element with the highest version is selected by default. If no date is specified, today's date is used by default.

The element identifier is specified as an operand:

operation x elem

operation	Statement name.
x	Element type.
elem	Element identifier consisting of elementname and optionally version and date.

Rules for element identifiers in source program, macro and object module libraries

elementname — Up to 8 characters

Character set:

Letters	: A-Z
Special characters	: $, #, @, - (hyphen), _ (underscore)
Digits	: 0-9

The first character must be a letter, $, # or @.

version — Three-character version identifier

Character set:

Letters	: A-Z
Special characters	: none
Digits	: 0-9

Letters may be used only as the first character.

On output (write access), however, no version specification may be made.
The target element contains either:

– version number X'001' if the source is a file or *EAM and the output library is a macro, source program or object module library, or
– the version identifier of the source element.

Specifying the date in the element identifier

The user date maintained for each element can be used for selecting elements. The user date is optional.
When an element is created or renamed, the user can specify the date. LMSCONV inserts the current date by default.

If DATE is specified, the current date is inserted instead of the user date.

Rules for specifying the date in source program, macro and object module libraries:

YYMMDD *Meaning:*

YYYY	: Year
MM	: Month
DD	: Day

LMSCONV adds a reference year to make the four-digit century:
YY < 60: LMSCONV fills to 20YY.
YY ≥ 60: LMSCONV fills to 19YY.

Use of the version

Object module libraries:

The update function UPD increments the version number of the corrected element by 1.

Selectors for element identifiers

An element identifier refers to just one element. It can however be structured as a variable, enabling several elements to be selected for processing.
This type of element identifier is known as a selector.

LMSCONV provides a number of symbols for variable element referencing. These symbols have the following meanings in selectors.

Symbol	Meaning in selectors
‘ (single quote)	Any character permitted in the element name, version and date. All elements are selected that have any character in the corresponding position in the element identifier. A single quote represents just one character. A space can also be used instead of the final quote, i.e. the selected names can also be shorter than the selector.
* (asterisk)	The asterisk as an element type: The statement refers to all element types in the assigned input library. The asterisk in the element identifier: -in the element name The asterisk can be used alone or in combination with other characters. When used in combination with other characters, it must be the last character in the name. From this position in the character string, the name may be of any length and content. If * is used alone, the element name may be of any length and have any contents. - as the version The statement refers to all versions of the specified element name. If a single * is specified for the complete element identifier, the statement refers to the current highest version of all the elements.
< (smaller > (greater) = (equal # (not equal)	These symbols allow limit values to be used for version and date in element selection. These characters always appear between the slash and the version or between the slash and the date. In conjunction with the specified values for version and date, these limit the selection of the elements to be processed.
-elem	This represents a ' minus element', i.e. it excludes an element that would have been selected by the preceding selector. elem can also be a selector, but it must indicate that it refers to an element or group of elements within the previous selector. There must be a comma between selector and -elem.

Examples of selectors

- Selectors

AB’C*/*	All elements whose name begins with AB, have any character in position 3, and C in position 4 are selected. From position 5 onwards, the element name can have any contents.
’”/B*	All elements with a name length of a maximum of 3 characters and which have a B in the first position of the version number are selected.
//>1983*	All elements entered since 1.1.84 are selected.
B//*	The highest version of all elements named AB and with any date is selected.

- Selectors with limit values

*/>402	All elements whose version number is greater than 402 are selected.
A*//<1982*	The highest version of all elements whose name begins with A and which have a date earlier than 1.1.1982 is selected.
A*/#B*	All elements whose name begins with A and whose version number does not begin with B are selected.
AB’/=107	All elements whose name begins with AB, which are a maximum of 3 characters long and which have version number 107 (equivalent to specifying AB’/107) are selected.

- Selectors excluding elements

AB*,-ABC,C*	The highest version of all elements whose name begins with AB or C, except for element ABC, is selected.
L”’/*,-L”’/001	All elements whose name begins with L and are a maximum of 4 characters long, except for those with a version number of 001, are selected.

Constructors for element identifiers

Some LMSCONV statements, in addition to the identifier of the element (elem) to be processed, also use a further element identifier elemu specified after a separator (,>=) in the statement. In the statements

ADD, DUP, NAM

elemu represents the element identifier of the new element (>elemu).

These element identifiers can be formulated as variables by using constructors.

The variable positions in the constructors are derived from the element being processed (elem). LMSCONV uses the same format for these constructors as for the selectors.

The symbols used for the element identifier elemu in these constructors have the following meanings:

Symbol	Meaning in constructors
‘ (singel quote)	The position indicated by a single quote in the constructor is replaced by the character in the corresponding position in the element identifier elem. A single quote defines a position in the element identifier. If the quote is in the last position in the constructor, no more characters are taken from the element identifier elem.
* (asterisk)	The asterisk can only be used as the last character in the constructor. It means that from this position onwards, all characters in the element identifier elem are used. If * is specified on its own, the complete element identifier elem is used.

All other positions remain unchanged.

Examples of constructors

A library contains 3 elements called

1. ABC/001
2. ABCD/234
3. ABCDE/101

Statement to be executed	Name constructed by LMSCONV
NAMn ABC>“X	1. ABX/001 2. is not renamed 3. is not renamed
NAMn AB*>XY*	1. XYC/001 2. XYCD/234 3. XYCDE/101
DUPn AB“>‘X‘Y	1. AXCY/001 2. AXCY/234 3. is not duplicated

Notes

The following points should be noted when using selectors and constructors:

– Different input identifiers may be mapped onto the same output identifier. Different data is overwritten depending on the value of the processing operand OVERWRITE (example: NAMx A*>B).

– If a constructor causes a lexicographically higher element identifier to be generated that also satisfies the selector, LMSCONV will find this element again when it processes the table of contents sequentially.

– If an element/selector occurs more than once (even beyond the limits of sublists), only the last selector is processed, for example:

```
NAMn A>B,A>C
```

Only A>C is processed.

6.3 Functions of LMSCONV

This section summarizes the functions of LMSCONV. The functions are activated by means of statements.
Processing operands are used to control and modify not only the statements, but also LMSCONV execution. Task switches, set when LMSCONV is called, also influence LMSCONV execution. For LMSCONV statement information, see section 6.5.4, "Description of the individual statements", and for processing operands see section 6.5.5, "Processing operands".

After LMSCONV has been called, all processing operands are set to default values. If different values are required for processing operands in individual statements, the processing operand must be set.

If, in addition to error messages, all LMSCONV statements are to be logged when successfully executed (positive acknowledgments), the value of the LOG operand must be set to LOG=MED or LOG=MAX.

Before a library can be accessed, it must first be assigned. Assigned libraries may already exist or be newly created. Elements cannot be added and/or processed until the library has been successfully assigned.

6.3.1 Assigning libraries

All LMSCONV functions require an input or an output library, or both. LMSCONV reads elements from an input library and outputs elements to an output library.

LIB assigns a library as an input, output or input/output library. The assignment remains in force until the next LIB or the end of LMSCONV execution.

The library ID can be used to assign a different input library temporarily for a statement. This assignment applies to this statement only, after which the LIB assignment reapplies.

Assigning libraries with LIB

LIB defines

– which library is to be assigned or closed
– whether the library is assigned as an input, output or input/output library
– whether the library is a program, source program, macro or object module library
– whether the library already exists or is to be created.

LIB references libraries by the file name of the library, the file link name or the library ID.

If the file link name or library ID is used, there must have been a previous ADD-FILE-LINK command establishing the link to the file name of the library.

```
/ADD-FILE-LINK LINK-NAME=filelinkname,FILE-NAME=libraryname
```

The library ID is defined in an ADD-FILE-LINK command using the file link name LIBlib:

– lib must be the three digits which represent the library ID in LMSCONV statements.
– Leading zeros may be omitted in statements, but not in the ADD-FILE-LINK command.
– lib must be enclosed in parentheses in statements.

Example

```
/ADD-FILE-LINK LINK-NAME=BETA,FILE-NAME=A.BIB
/ADD-FILE-LINK LINK-NAME=LIB001,FILE-NAME=B.BIB
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB BETA,IN
$LIB (1),OUT
.
.
.
```

Library A.BIB is given the file link name BETA, is referenced during the LMSCONV run by the file link name, and is opened as an input library.

Library B.BIB is given library ID (1), is referenced during the LMSCONV run by the library ID and is opened as an output library.

The user may only use one file link name for each library.

Sequential libraries

Sequential libraries must be assigned by an ADD-FILE-LINK command containing the operand ACCESS-METHOD=*BTAM. If the tape library has no catalog entry and is to be read, the IMPORT-FILE command must be specified. The following commands must be specified:

a) File does not yet exist

```
/CREATE-FILE FILE-NAME=filename,SUPPORT=*TAPE(VOLUME=vsn,
DEVICE-TYPE=device)

/ADD-FILE-LINK LINK-NAME=LIBlib,FILE-NAME=filename,ACCESS-METHOD=*BTAM
```

b) File available on tape, but not yet cataloged

```
/IMPORT-FILE SUPPORT=*TAPE(VOLUME=vsn,DEVICE-TYPE=device,
FILE-NAME=filename)

/ADD-FILE-LINK LINK-NAME=LIBlib,FILE-NAME=filename,ACCESS-METHOD=*BTAM
```

c) File exists and is cataloged

```
/ADD-FILE-LINK LINK-NAME=LIBlib,FILE-NAME=filename,ACCESS-METHOD=*BTAM
```

Sequential libraries cannot be created by LIB. They must be assigned using LIBOUT and the library ID.

Temporary library assignment using the library ID

Libraries assigned by LIB remain assigned as input or output libraries until LIB assigns new libraries. If no new assignments are made, the assignments apply until the end of the LMSCONV run.

LMSCONV also offers the option of opening another input library for just one statement. In this case, the library ID is specified as an operand in the statement. This library is available for the duration of this statement only.

The input library assigned by LIB also applies in the next statement which contains no library ID as an operand.

The library ID is declared using the file link name LIBlib in an ADD-FILE-LINK command (see "Assigning libraries with LIB" above).

Example

```
/ADD-FILE-LINK  LINK-NAME=LIB001,FILE-NAME=EIN.BIB
/ADD-FILE-LINK  LINK-NAME=LIB002,FILE-NAME=AUS.BIB
/ADD-FILE-LINK  LINK-NAME=LIB003,FILE-NAME=PLA1.BIB
/ADD-FILE-LINK  LINK-NAME=LIB004,FILE-NAME=PLA2.BIB ———————————————  (1)

/START-PROGRAM  FROM-FILE=$LMSCONV ———————————————————————————————  (2)

$LIB(1),IN
$LIB (2),OUT ————————————————————————————————————————————————————  (3)

$DUPS ELEM1>ELEM01 ——————————————————————————————————————————————  (4)

$LSTS ELEMENT1(3)
$TOCS (4) ———————————————————————————————————————————————————————  (5)

$LSTS ELEM1 —————————————————————————————————————————————————————  (6)

$END ————————————————————————————————————————————————————————————  (7)
```

(1) The input and output libraries are assigned using the ADD-FILE-LINK command, as they are referenced by means of library IDs in the LMSCONV run.

(2) LMSCONV is called.

(3) The default libraries for all statements are assigned using LIB.

(4) The element ELEM1 is duplicated and is available as element ELEM01 in the corresponding output library.

(5) The libraries assigned using LIB apply to this statement as it contains no library IDs. This lists specified elements and table of contents of the library.

(6) Element ELEMENT1 in library PLA1.BIB, for which library ID (3) was defined in the ADD-FILE-LINK command, is listed; the table of contents for type S elements in library PLA2.BIB is output.

(7) LMSCONV is terminated.

For a description of the statements used see section “Statements” on page 184.

6.3.2 Processing elements

The following section summarizes the different ways in which LMSCONV processes elements.

LMSCONV can

- add elements to libraries
- output elements to files
- output elements to other libraries (duplicate)
- list elements
- delete elements
- rename elements
- update elements
- output the table of contents for a library.

Adding elements to a library

The following statements output elements to the assigned output library: ADD, DUP, PRT, UPD.

The processing operand OVERWRITE determines whether or not an existing element with the same name in the output library is overwritten.

ADD takes files and modules from the EAM area and adds them as elements to the assigned output library.

When a SAM/ISAM file is added, the processing operand KEY determines whether the file attributes and the ISAM key are also added.

Setting the KEY processing operand means that files with RECORD-FORMAT = FIXED can also be added.

Only files with RECORD-FORMAT = VARIABLE can be added to source program and macro libraries.

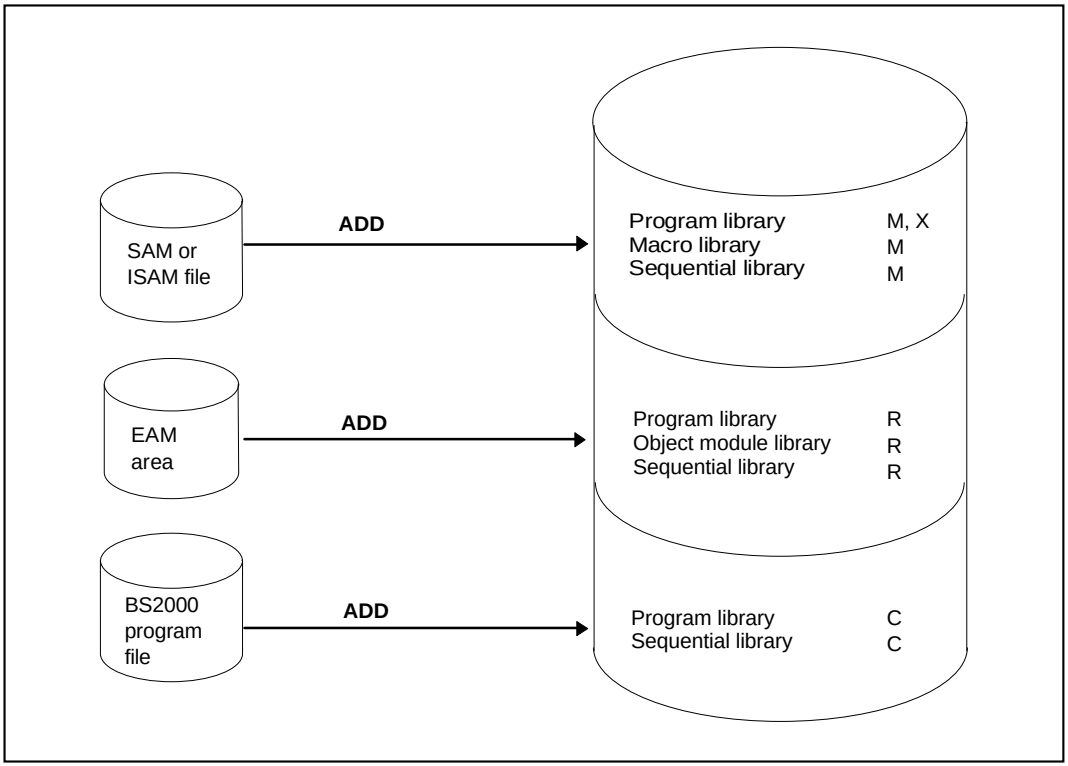


Figure 6: Adding elements with ADD

PRT can store the LMSCONV log in a list element.

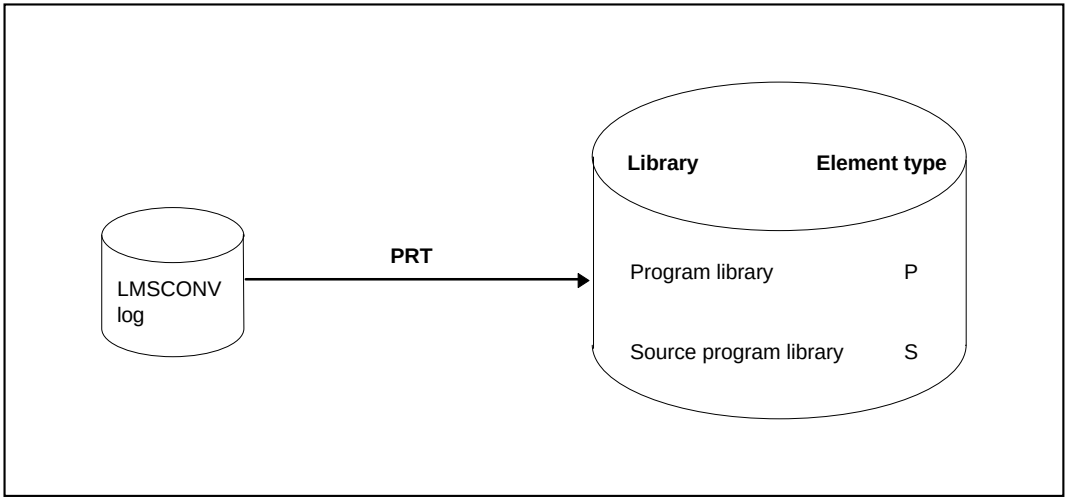


Figure 7: Adding elements with PRT

DUP copies elements from the input library to the output library, storing them under a different element identifier if required.

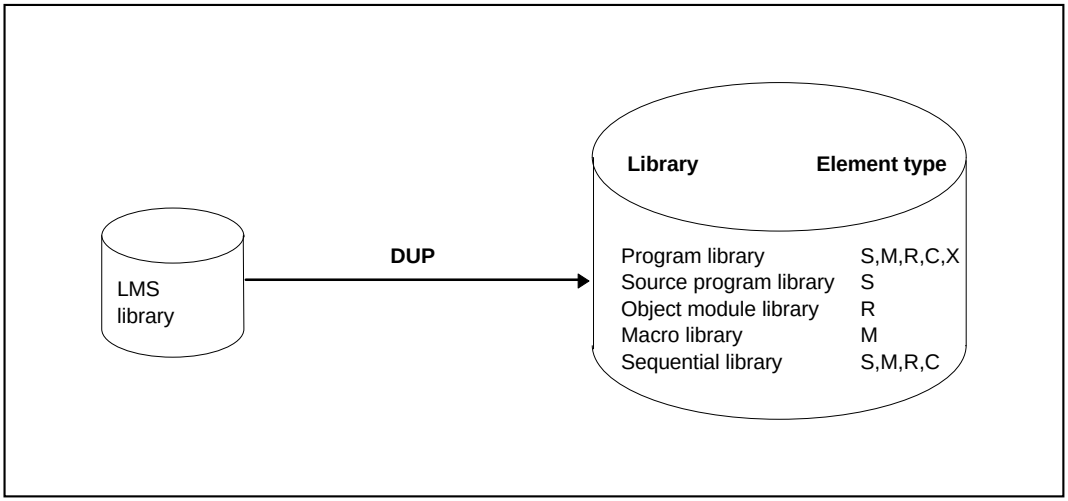


Figure 8: Adding elements with DUP

Outputting elements

Elements in an input library are output

– to files using SEL or
– to the output library using DUP.

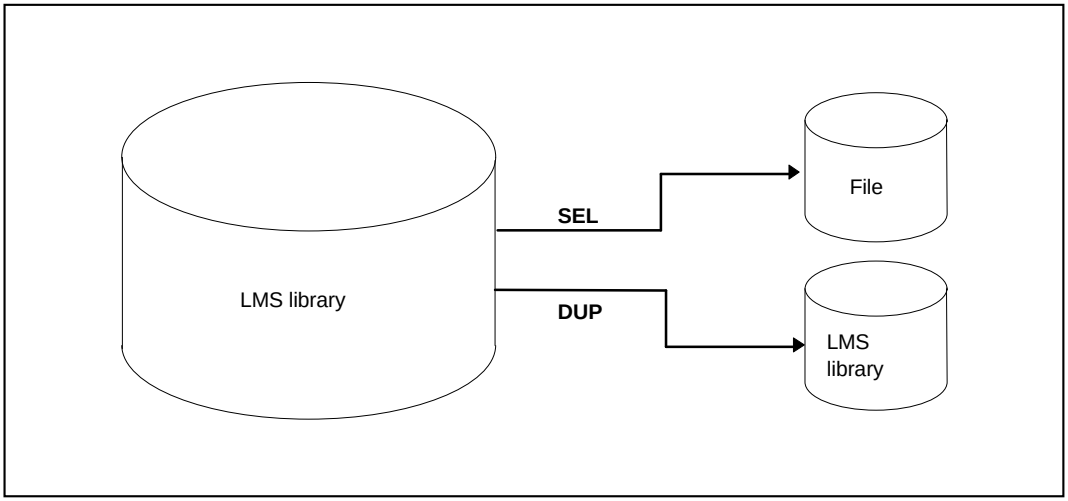


Figure 9: Outputting elements

Listing elements

Elements can be listed using the **LST** and **PRT** statements. LST defines the elements and the library from which the elements are to be listed.
PRT defines the output medium.
The format and scope of the output are defined by processing operands.

Deleting elements

DEL deletes one, several or all elements in the input library (program, source program, macro or object module library).
Data in program libraries can be deleted in two ways: with or without destroying the data:

– Deletion without destroying the data
  The entries in the table of contents are deleted and the space occupied by the corresponding element is released.
– Deletion with destruction of data
  In addition to the above, the space occupied by the corresponding element is overwritten with binary zeros.

With elements in program libraries, the data is only overwritten if the processing operand DESTROY=YES is set or if the element contains a data destruction indicator.

Updating elements

LMSCONV provides the update statement UPD for modifying elements. UPD enables object and load modules (element types R and C) and link and load modules (element type L) to be updated (corrected).

UPD updates the specified element in the assigned input library. The updated element is then written to the assigned output library with a new element identifier if required.

UPD has various substatements for updating object and load modules and LLMs. The substatements are read from the statement stream immediately following UPD, up to the *END statement.

The functions of substatements for object modules are:

– updating text records
– reversing text updates
– converting updates, i.e. converting REP records to text updates or vice versa
– inserting REP records
– inserting INCLUDE records
– modifying control section attributes
– excluding record types from the input element
– renaming symbols
– defining check numbers
– defining identifications
– defining the base address.

The functions of substatements for load modules are:

– updating text records
– reversing text updates
– deleting update journal records
– defining check numbers
– defining identifications
– defining segments
– defining the base address.

The functions of the substatement for LLMs are:

– updating text records
– reversing text updates
– deleting update journal records
– defining identifications.

Renaming elements

NAM renames the specified elements in the assigned input library. This statement also allows the renaming of elements whose identifiers do not conform to LMSCONV conventions.

Outputting a library's table of contents

TOC lists the table of contents (TOC) entries for the specified elements or for the complete input library.

The SORT processing operand defines whether the table of contents list is to be output unsorted or sorted by name, version number, date or reference names. By default, the element identifiers are output sorted by name, version number and date.

Table of contents lists for program libraries are always output sorted alphabetically by element type and name. Use the LMSCONV statement TOC or TOC* */* to output the complete table of contents of a program library.

Storing and calling procedures

LMS enables the user to store BS2000 procedures and ENTER procedures as elements in libraries (element type J).

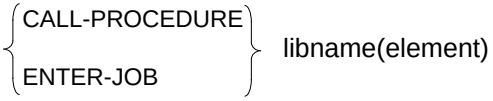
Existing procedure files can be added to libraries as elements with the aid of the ADD-ELEMENT statement.

Space can be saved in memory by storing procedures, particularly if the command files are small. The number of catalog entries is reduced.

Note, however, that if ISAM keys are stored as part of elements, these keys are removed before the procedure is called.

A library element can also be assigned as a system input file (SYSDTA) by means of the BS2000 command ASSIGN-SYSDTA (see the "Commands" manual [1]).

The call for procedures stored as library elements is as follows:



6.4 Controlling LMSCONV execution

LMSCONV execution is controlled by processing operands and job switches.

6.4.1 Control via processing operands

Processing operands influence not only execution of LMSCONV, but also individual functions. They are set by means of **PAR**.
The following table shows which processing operands operate on which statements.

LMSCONV statements / Processing operands	ADD	DEL	DUP	LST	NAM	PRT	SEL	TOC	UPD	RST
CS[ECT]				*					*	
DEC[OMPRESSED]	*		*							
DES[TROY]	*	*	*		*	*			*	
FO[RMAT]				*						
I[NFO]				*						
K[EY]	*									
O[VERWRITE]	*		*		*	*	*		*	
PA[TH]				*					*	
PH[ASE]							*			
RE[FERENCE]		*	*	*				*		
SL[ICE]				*					*	
SO[RT]								*		
STRIP			*						*	

The processing operands LOG and TERMINATE remain in effect throughout LMSCONV execution. They have no special effect on any individual statements.

The processing operands determine

– whether the data is destroyed when elements are deleted (DESTROY)
– whether elements are overwritten (OVERWRITE)
– the scope and format of output for listings and logs (FORMAT, INFO, LOG, PATH, SLICE, CSECT)
– for elements of type R or C, which record types are not moved from the input to the output element (STRIP)
– whether the table of contents of a library is to be output unsorted or sorted by name, version number, date or reference name (SORT)

– what response is to be made in the event of an error (TERMINATE)
– whether only elements that satisfy a reference condition are to be processed (REFERENCE)
– the load module format to be created (PHASE).

Controlling log output

The log of the LMSCONV run may include:

– input statements
– execution or termination of the statement
– assigned input and output libraries
– generated listings.

The log is written to the system file SYSOUT, SYSLST or to a library element. The output medium is defined by PRT.

If the log is written to an element, LMSCONV creates a type P element for program libraries and a type S element for source program libraries. If the library to which the element is to be written is a source program library, it cannot be the default input or output library for this LMSCONV run.

The scope and format of the log are governed by processing operands and job switches.

If job switch 4 was set when LMSCONV was called, LMSCONV's start and end messages are suppressed. The LMSCONV log is also restricted to a minimum (equivalent to setting processing operand PAR LOG = MIN).

If job switch 8 was set when LMSCONV was called, the access routine messages (AMCB, DMS) are not logged.

The following table gives an overview of the effect of processing operands on the LMSCONV log:

Processing operand	**Function**
LOG	Defines whether all statements, statements in error or messages only are to be logged.
FORMAT	Defines the record format when an element is listed.
INFO	Defines the scope of the output when an element is listed.
SORT	Defines sort criteria for table of contents listings.

Positive and negative acknowledgments

If the value of the LOG processing operand is LOG=MAX or LOG=MED, the execution of every LMSCONV statement affecting an element is logged. If the statement is successfully executed, LMSCONV outputs a positive acknowledgment.

If the statement cannot be executed, it is logged by LMSCONV together with a negative acknowledgment and any corresponding LMSCONV error message.

Interrupting LMSCONV execution by the user

The user can interrupt LMSCONV execution by pressing a program interrupt key (e.g. K2).

LMSCONV can be continued by means of the SEND-MSG command, with or without input text. This input text is then interpreted by LMSCONV's interrupt handling. The function currently executing is informed of the type of termination.

The following actions are carried out on an interrupt:

– LMSCONV is interrupted by BREAK/ESCAPE (K2 or similar).
  If LMSCONV is running interactively, the appropriate STXIT routine is activated.
– LMSCONV executes a BKPT macro (only in interactive mode).
– Input of an SEND-MSG command causes the BREAK-STXIT routine to be exited and the INTR-STXIT routine to be activated.
– LMSCONV analyzes the text entered with the SEND-MSG command, and informs the current function of the type of termination required.
– The INTR-STXIT routine is exited via the EXIT macro.
– The interrupted function is terminated in the required manner.

SEND-MSG command

The SEND-MSG command is used to control the way in which the execution of functions is terminated. A text entry can be specified in the command. This text is passed to the interrupted function by the interrupt handler. The following texts are permitted:

SEND-MSG ‘NI’	The next input buffer is processed (NEXT INPUT). All outstanding LMSCONV statements are ignored. This means that statements currently being processed are terminated and any statements remaining in the input buffer are ignored. All outstanding activities are ignored.
SEND-MSG ‘NS’	Execute next statement (NEXT STATEMENT). The current statement and any statements brought forward to this point are ignored; the next statement from the statement buffer is executed. NS is also the default if no operands are specified for the SEND-MSG command.
SEND-MSG ‘NE’	Stop processing this element (NEXT ELEMENT). Processing continues with the next element that satisfies the current selector.

Termination of LMSCONV execution due to error

Error handling is also performed by the STXIT routine.

Program termination is caused by

– Program error ("P error", SVC error).
– Abnormal end due to START-PROGRAM, LOAD-PROGRAM, CANCEL-JOB, EXIT-JOB or a connection failure.
– timeout (program or task timeout).

In all these cases, precautions are taken to ensure that libraries remain consistent. LMSCONV primarily ensures that all libraries are correctly closed.

The following statements are true of all program terminations:

– All STXIT routines in LMSCONV are deactivated to avoid irregular processing by SEND-MSG.
– LMSCONV simulates an END. This causes all open libraries to be closed.

If any libraries are still open at program termination, they are closed.

Program error

Before END is simulated, the following message is output to SYSOUT:

```
PROGRAM ERROR AT loc (IW=iw)
```

loc specifies the interrupt address and iw the interrupt weight.

LMSCONV terminates with a dump.

Program error handling always ensures the following:

– Diagnostic documentation is always produced.
– No continuation of LMSCONV is possible using RESUME-PROGRAM after a program error (this also applies to other types of interrupt) as this would be pointless.

Diagnostic tools

Setting job switch 31 causes a test condition to be set in LMSCONV. If a program error occurs when LMSCONV is running in batch mode, it is aborted with a dump. The registers have the same contents as at the time of the interrupt. If job switch 31 is set in interactive mode, the query:

```
DO YOU WISH A BKPT (Y/N)?
```

is output. If the answer is yes, the registers are loaded as they were at the time of the interrupt, and a BKPT macro is issued. The INTR-STXIT interrupt routine is deactivated to prevent LMSCONV being continued by SEND-MSG.

In all cases a dump is initiated before a simulated END is executed and LMSCONV is terminated.

There is no point in issuing a BKPT macro other than in interactive mode. It is requested by setting job switch 31.

6.4.2 Using job switches

BS2000 job switches enable the user to influence the LMSCONV session. They must be set before LMSCONV is loaded, using the system command MODIFY-JOB-SWITCHES ON = (no,...).

The following job switches can be used to influence execution of LMSCONV:

Job switch 1:
In interactive mode, the LMSCONV statements are read by default from the terminal using the WRTRD macro. If job switch 1 is set, the statements are read from the logical system file SYSDTA using the RDATA macro.

When LMSCONV is called in BS2000 procedures, job switch 1 must be set if the LMSCONV statements are to be read from the procedure file.

Job switch 4:
When job switch 4 is set, LMSCONV's start and end messages are suppressed. At the same time, LMSCONV's execution log is restricted to minimum scope (cf. LOG=MIN).

Job switch 8:
Setting job switch 8 causes the messages from the access routines (AMCB, DMS) to be suppressed.

Job switch 9:
Setting job switch 9 enables the user to request more space. This enables the TOC function to sort/merge comprehensive table of contents lists.

Job switch 31:
Job switch 31 enables a test condition to be set, which can be used for diagnostic purposes.

The job switches are only interrogated at start time. Setting or resetting them subsequently has no effect on LMSCONV.

6.4.3 PAM key elimination

For detailed information on PAM key elimination, see the section of the "LMS" manual [11] immediately preceding the description of the statements.

6.5 Statements

6.5.1 Statement syntax

The following metacharacters are used to represent the format of the statements and processing operands:

Notational conventions	**Explanation**	**Example**
UPPERCASE LETTERS and special characters	Uppercase letters and special characters denote constants which must be entered by the user exactly as shown.	NAMx elem(lib)> elemu Enter: NAMR MODLA(1)> AMOD
lowercase letters	Lowercase letters denote variables that must be replaced with actual values by the user.	
{ }	Braces enclose alternatives, i.e. one of the options must be selected.	PRT { (LST) / (SYSOUT) / (BOTH) / elem[(lib)] / ? } Enter: PRT (LST) or PRT (SYSOUT) or PRT (BOTH) or PRT PROT(1) or PRT ?
[ ]	Square brackets denote optional entries.	LSTx elem[(lib)] Enter: LSTM MACRO or LSTM MACRO(3)
...	Ellipses denote repetition; the preceding unit can be repeated several times in succession.	elem(lib),... Enter: A(1) or A(1),B(2) or A(1),B(2),C(3) etc.

Notational conventions	Explanation	Example
_	The underscore highlights the default value. This is the value that is used if no other value is specified by the user. If nothing is underscored, either there are different default values in batch and interactive modes, or there is no default assignment.	PAR DEC={YES / NO} Enter: PAR DEC=YES or PAR DEC=NO or PAR DEC=, which corresponds to PAR DECO=NO.

6.5.2 Statement format

LMSCONV statements consist of three parts:

– operation
– operands
– comments

General format:

```
[$]operation_operands_comments
```

A dollar sign ($) sign may appear as the first character of the statement, but this is not essential.

Operation

The operation must appear at the beginning of the statement. It consists of the statement name and, if the statement is to process elements, the element type.

Example

ADDx Statement syntax.

ADD Statement name.

x x should be replaced by the element type, e.g. ADDS for source programs

The element types supported by each statement are listed in the statement description.

Operands

The operation is followed by the operands, separated from it by at least one space. Operands are separated by commas. In many statements, the separators "=" and ">" are also used to separate operands. The ">" character is the symbol for an arrow indicating the direction of processing.

There must be no separator before the first or after the last operand of the statement as a whole. No spaces may appear within operands, or between operands and separators. Statements may be a maximum of 2028 bytes long.

Example

```
ADDM SRCE.DAT>SRCE.ELEM
```

The file SRCE.DAT is stored in the library as element SRCE.ELEM.

Comments

Comments may follow the operands, separated from them by at least one space.

No comments may be included in statements without operands.

If the comment is to extend over a complete line (comments lines) these lines must be flagged by an asterisk (*) in column 1 and a space in column 2.

The characters ! and X'15' (NEW-LINE) must not occur in comments, as they are interpreted as statement separators.

Continuation lines

A statement can consist of more than one line. The operation section must be at the beginning of the first line. The operand section can extend over several lines. Substatements such as *COR must not be separated.

To indicate a continuation, a continuation character or a space must immediately follow one of the separators. The continuation character must be within the column range 1 to 72.

The statement can be continued at any position in the continuation line.

Continuation characters

The continuation character can be represented by a hyphen (-) or by the plus (+) character.

Separators

The separators are "," or ">" or "=".

Entering blocked statements

Statements can also be entered in blocks.

In interactive mode, this means that each statement does not have to be sent individually; instead, data transmission can be initiated for a group of statements together.

Thus several statements in one line, separated by the exclamation mark (!), and statements extending over more than one line, separated by logical end-of-line, can be sent off collectively.

Logical end-of-line is the NEW-LINE character (NL) applicable for the relevant terminal type. Depending on the terminal, it is normally represented by the character \ or <.

6.5.3 Overview of all LMSCONV statements

Statement	Operands	Usage
ADDx	{ filename,... / LINK=filelinkname / [prefix.]name[.suffix] / [prefix.](name,...)[.suffix] } [>elemu]	Stora data in a library
ADDR	*OMF[(module1,...)][>elemu]	Store modules from the EAM area
COPYLIB	(lib1)>(lib2)	Copy a library
DELx	{ elem[(lib)] / (lib) }[,...]	Delete elements
DUPx	{ elem,...[(lib)] / (lib) }[>elemu][,{...}[>...]]	Duplicate (copy) elements
END		Terminate LMSCONV execution
LIB	{ FILE=libraryname / LINK=filelinkname / [LIBRARY=]name / [LID=](lib) } [,[USAGE={ IN / OUT / BOTH }] [,[FORMAT={ PL / OML / OSM }][,[STATE={ O[LD] / N[EW] / A[NY] }]	Assign libraries
LIB	C[LOSE][,{ FILE=libraryname / LINK=filelinkname / [LIBRARY=]name / [LID=](lib) }]	Close libraries
LIB	?	Display assigned libraries

Statement	Operands	Usage
LIBOUT	[{ (lib)[, { (vsn) | NEWLIB | NEWLIB(vsn) }] | libname | LINK=linkname }	Assign a sequential output library
LSTx	{ elem[(lib)] | (lib) } [,...]	List elements
NAMx	{ elem,...[(lib)] | (lib) } >elemu[,{...}>...]	Rename elements
PAR	{ parname=[{ parval | ? }] | ? } ,{...}]	Set processing operands
PRT	{ (LST) | ([SYS]OUT) | (CON) | (BOTH) | elem[(lib)] | ? }	Control log output
RST	[STOP]	Restart after error
SELx	elem[> { [prefix.](name)[.suffix] | filename | LINK=filelinkname }]	Output elements to files
SYS	[{ 'systemcommand' | systemcommand }]	Initiate system commands or switch to system mode
TOCx	[{ elem[(lib)] | (lib) } ,...]	List table of contents of a library
UPDx	elem[(lib)][>elemu]	Update object and load modules

Table of permissible element types for each statement

The following table summarizes which element types are permissible in each of the statements:

Element type / Statement	S	M	R	J	P	C	D	X	H	L	F	U
ADD	+	+	+	+	+	+	+	+				
DEL	+	+	+	+	+	+	+	+	+	+	+	+
DUP	+	+	+	+	+	+	+	+	+	+	+	+
LST	+	+	+	+	+	+	+	+	+	+	+	+
NAM		+	+			+				+		
SEL	+	+	+	+	+	+	+	+				
TOC	+	+	+	+	+	+	+	+	+	+	+	+
UPD			+			+				+		

+: Element type permitted for statement

empty box : Element type not permitted for statement

Table of required libraries

The following table shows which libraries are required by the individual LMSCONV statements:

Function	LMSCONV requires	
	Input library	Output library
ADD	no	yes
DEL 1)	yes	no
DUP	yes	yes
LST	yes	no
NAM 1)	yes	no
PRT	no	yes
TOC	yes	no
SEL	yes	no
UPD	yes	yes

1) The input library is opened for writing.

6.5.4 Description of the statements

ADD
Add files to library

ADD stores the following as library elements:

– files
– modules from the EAM area

ADD has two different formats corresponding to these different functions:

ADD (Format 1): Adding files

This ADD format takes files and stores them as elements in the open output library. Files of any buffer length can be added to a PLAM library.
The output library must previously have been assigned using LIB. File generation groups can only be stored using LINK= and an element identifier supported by LMSCONV. When the ADD statement is used, LMSCONV uses the CCS catalog attribute (CCS = coded character set) of the file as the element attribute.

Operation	Operands
ADDx	{ filename,... / LINK=filelinkname / [prefix.]name[.suffix] / [prefix.](name,...)[.suffix] } [>elemu]

ADDx — Statement name including code for element type (x).
The following element types are permitted:
M, R, C, X, S, J, P, D

filename — Fully qualified file name or selector.
“pathname” can be specified instead of “filename”.
If temporary files are being stored, selectors are not permitted.
Multiple file names are only meaningful where * or a constructor is specified for elemu.

LINK=filelinkname — File link name pointing to the file.

{[prefix.]name[.suffix]
[prefix.](name,...)[.suff}

This entry enables one or more files to be selected for storing in the library.

prefix
Specifies the common prefix of the files to be selected. 'prefix' must end with a period.

suffix
Specifies the common suffix of the files to be selected. 'suffix' must start with a period.

name
Specifies the partial file name(s) which, with prefix and/or suffix, become(s) one or more fully qualified file names.
The elements created are stored in the library with these partial names even if elemu is not specified. A selector is also permitted for 'name'.

Partially qualified file names can also be specified in the form prefix.(*).suffix.

elemu — Name of the element to be generated.
Where [prefix.](name,...)[.suffix] or a selector is specified for filename, constructors are also permitted. elemu can be omitted, in which case the element generated is given the name of the input file.

The element identifiers of the elements to be generated are subject to the various syntax rules for program libraries (see section "Contents of a program library") and other library types (see section "Contents of single-type libraries").

Processing operands

DESTROY — Defines whether the output element contains a flag indicating data destruction on deletion. (Only possible for program libraries.)

KEY — Defines whether file attributes and any existing ISAM keys are to be stored.

OVERWRITE — Defines whether an existing element of the same name in the output library is to be overwritten.

DECOMPRESSED Defines whether data in macro/source program libraries is compressed.

Example 1

Given these files:

A.B.C.A A.B.C.B A.B.C.C B.B.C.A C.B.C.A

the statement

```
ADDx A.B.(*)            selects      A.B.C.A    →  element name C.A
(only possible for      files        A.B.C.B    →  element name C.B
program libraries                    A.B.C.C    →  element name C.C
ADDx (A,C).B.C.A        selects      A.B.C.A    →  element name A
                        files        C.B.C.A    →  element name C
```

Example 2

```
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB TESTLIB,OUT
$ADDM M.MAC>MMAC
$END
```

The macro with file name M.MAC is stored in program library TESTLIB as a type M element with the name MMAC.

Example 3

```
/ADD-FILE-LINK LINK-NAME=PROG,FILE-NAME=PROG.DAT
/ADD-FILE-LINK LINK-NAME=TEST,FILE-NAME=TEST.COB
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB TESTLIB,OUT
$ADDM LINK=PROG>DDAT
$ADDM LINK=TEST>TCOB
$END
```

The files PROG.DAT and TEST.COB are stored in program library TESTLIB as elements DDAT and TCOB. In ADD they are referenced by their file link names.

ADD (Format 2): Adding modules from the EAM area

This ADD format takes modules from the EAM area of the current task and stores them as type R elements in the assigned output library. Output libraries here can be program or module libraries or sequential libraries.
When the ADD statement is used, LMSCONV uses the CCS catalog attribute (CCS = coded character set) of the file as the element attribute. If *OMF is read, 'no code' is the CCS value assigned to the elements.

Operation	Operands
ADDR	*OMF[(module1,...)][>elemu]

ADDR — Statement name with element type R.

module — Name (up to 8 characters in length) of a module in the EAM area. If there are several modules of the same name in the EAM area, LMSCONV adds the most recently compiled module to the library. If the module entry is omitted, LMSCONV adds all the modules in the EAM area.
A selector is permitted.

elemu — Element identifier of the element to be created.

A constructor is permitted.

elemu can be omitted, in which case modules will be given the name they had in the EAM area.

Note

If an element identifier is specified for elemu, the name of the module to be added must be specified for 'module' if there are several modules in the EAM area.
Otherwise, all modules from the EAM area will be stored as 'elemu', with each new module overwriting the one just added.

Example

```
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB TESTLIB,OUT
$ADDR *OMF(MOD1,MOD2)>ELM*
 .
 .
 .
$END
```

Object modules MOD1 and MOD2 from the EAM area are added to output library TESTLIB as elements ELM1 and ELM2.

COPYLIB
Copy library

COPYLIB copies the specified library in its entirety, i.e. complete with all data, library, type and element attributes.
The target library may not exist or must have FILE-STRUC=NONE (only a catalog entry exists for the library, it was not opened beforehand). The target library is assigned a library format in accordance with its buffer length.
If an error occurs while the COPYLIB statement is being processed, the target library will be incomplete.

Operation	Operands
COPYLIB	(lib1)>(lib2)

lib1 — Abbreviation for the library to be copied (zzz in the file link name LIBzzz, which was assigned beforehand by means of an ADD-FILE-LINK command).

lib2 — Abbreviation for the target library (zzz in the file link name LIBzzz which was assigned beforehand by means of an ADD-FILE-LINK command).

To be successful, the COPYLIB statement must be called by a user having the following access rights:

- Read permission for the source library.
  In addition, one of the following conditions must be satisfied:
  - the user is the owner of the source library
  - or the source library is not protected against TP-UPAM access (i.e. the library file is not protected by ACL, BACL, etc.)
  - or no protection attributes exist on the library, type and element levels.
- Write permission for the target library.

Example

An NK2 PLAM file is converted into an NK4 PLAM file:

```
/ADD-FILE-LINK FILE-NAME=source,LINK=lib000
/ADD-FILE-LINK FILE-NAME=target,LINK-NAME=lib999
/ADD-FILE-LINK FILE-NAME=4ktarget,LINK-NAME=lib004,BUFFER-LENGTH=STD(2)
/START-PROGRAM FROM-FILE=$LMSCONV
$COPYLIB (000)>(004) ------------------------------------------------------------ (1)
$LIB     (999),NEW
$DUP*     */*(000) ------------------------------------------------------------ (2)
$
$END
```

(1) The library is copied in its entirety (including the attributes).

(2) The elements for which the user has read permissions are copied.

DEL
Delete elements

DEL deletes the specified elements in the assigned input library. The TOC entries are also deleted and the storage space is released.

In program libraries, the data contained in the elements is also destroyed, i.e. overwritten with binary zeros, if

– the element contains a data destruction indicator;
– the processing operand DESTROY=YES is set.

Operation	Operands
DELx	$\left\{\begin{array}{l}\text{elem[(lib)]}\\\text{(lib)}\end{array}\right\}$[,...]

DELx — Name of the statement including element type:
S, M, R, C, J, P, D, X, H, L, F, U

elem — Element identifier or selector.

lib — Library ID of the input library.If no element is specified, the function applies to all the elements in the highest version of the named library, i.e. the whole library is deleted unless it contains elements with the same name but different versions.

Processing operands

DESTROY — Specifies whether data is also to be destroyed on deletion. (Only possible for program libraries.)

REFERENCE — Specifies the reference names that the elements to be deleted must contain. If the reference condition is not satisfied, the element is not deleted. REFERENCE is only permitted for type R elements. If a REFERENCE is given, DEL must specify element type R.

Example

```
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB LS.LIB,BOTH
$DELM DATA
$DEL* TPROG
$END
```

The type M element DATA in library LS.LIB is deleted.

DUP
Duplicate elements

DUP duplicates the specified elements in the assigned input library, or a complete library, placing the copy in the open output library. The duplicated elements can be given new identifiers.

An output library must have previously been assigned by means of LIB.

Elements can be selected for duplication by means of their reference names, i.e. if a reference condition is specified using the REFERENCE processing operand, only the elements fulfilling this condition are duplicated.

Operation	Operands
DUPx	{ elem,...[(lib)] / (lib) } [>elemu][,{...}[>...]]

DUPx — Name of the statement including the element type.

The following element types are permitted:
S, M, R, C, L, X, J, P, D, H, F, U * can be used to indicate all element types (only allowed for program libraries).

elem — Element identifier of the element to be duplicated, or selector.

Names that do not satisfy LMSCONV conventions can be specified for elem in order to allow such elements to be processed.

lib — Library ID of the input library.

elemu — Element identifier of the output element or constructor.

DUPx */* duplicates <u>all</u> elements of a type, retaining the original element identifiers.

Processing operands

DECOMPRESSED — Specifies whether the data in macro/source program libraries is compressed.

DESTROY — Specifies whether the output elements contain a data destruction indicator.
(Only possible for program libraries.)

OVERWRITE — Governs the overwriting of elements with the same name in the output library.

REFERENCE	Determines which reference names the elements to be duplicated must contain. If the reference condition is not fulfilled, the element is not duplicated. REFERENCE is only permitted for type R elements. If a REFERENCE is given, DEL must specify element type R.
STRIP	Only for type R and C elements. Determines which record types are to be excluded from duplication.

Example

```
/ADD-FILE-LINK LINK-NAME=LIB002,FILE-NAME=DUP.LIB
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB OLD.LIB,IN
$LIB (2),NEW,OUT
$DUPM OLD1>DUP1,OLD2>DUP2
$END
```

Elements in library OLD.LIB are duplicated into a new program library called DUP.LIB. The file link name LIB002 is assigned to the new output library DUP.LIB. The input library is explicitly assigned by LIB. Element OLD1 is duplicated as DUP1, and OLD2 as DUP2 with unchanged version and date.

END
End LMSCONV run

END terminates the LMSCONV routine. Any libraries that are still open are closed.

In interactive mode, LMSCONV always terminates normally. In batch mode and procedures, the type of termination depends on the termination flag (see the TERMINATE processing operand). If an internal LMSCONV termination flag is set, the flag is output with the LMSCONV termination message (LMSCONV-TERM-MSG:).

Operation	Operands
END	

The termination code indicates the most serious error that has occurred. This is stored in the monitoring job variable specified in the call

```
/START-PROGRAM FROM-FILE=$LMSCONV,MONJV=<name>
```

Termination code	Meaning
0	No error has occurred
1	Warning messages have been issued
2	Error has occurred with termination flag (see the TERMINATE processing operand)
3	Internal LMSCONV error (with dump)

Following LMSCONV termination, the first byte of the status indicator of the monitoring job variable contains the termination code (see above) and the fourth byte contains the internal termination flag (see PAR TERM).

LIB
Assign and close libraries

LIB enables input and output libraries to be created, opened and closed. LMSCONV reads elements from input libraries and writes elements to output libraries.

Exceptions to this rule are DEL and NAM, which operate on the assigned input library.

There are 3 formats of the LIB statement, as described on the following pages.

LIB (Format 1): Assigning libraries

LIB assigns libraries. It can be used to specify the following:

– It defines the library as an input or output library or both.

– It defines the library type
(program, source program, macro or object module library).

– It defines whether the library is to be created, whether it must already exist or whether it is to be created if required.

When LIB assigns a new input library, it closes any input library previously assigned by a LIB statement. The same applies to output libraries.

The newly assigned library is opened.

Operation	Operands
LIB	{ FILE=libraryname | LINK=filelinkname | [LIBRARY=]name | [LID=](lib) } [,[USAGE=]{ IN | OUT | BOTH }] [,[FORMAT=]{ PL | OML | OSM }][,[STATE=]{ O[LD] | N[EW] | A[NY] }]

LIB	Statement name.
FILE=libraryname	Fully qualified file name of the library. libraryname can also be pathname (see the manual "Commands, Volume 1" [1]).
LINK=filelinkname	Specifies the file link name with which the library was assigned in an ADD-FILE-LINK command.

LIBRARY=name	LMSCONV first tries to interpret "name" as a file link name. If no such file link name was previously assigned using an ADD-FILE-LINK command, "name" is interpreted as libraryname. The keyword 'LIBRARY=' may be omitted. In this case the library name must not be 'CLOSE', otherwise LIB format 2 will be assumed.
LID=(lib)	Specifies the library ID, consisting of a maximum of 3 digits. The library ID must previously have been defined with the file link name LIBlib in an ADD-FILE-LINK command.
USAGE	This operand specifies whether the library is to be used for input, output or both. LMSCONV processes a library as an input and/or output library. The **input library** is used by LMSCONV as an input medium; it outputs elements to the **output library**.
=IN	In this LMSCONV run, the library is the input library. Default value with STATE=OLD.
=OUT	In this LMSCONV run, the library is the output library. Default value with STATE=NEW or STATE=ANY.
=BOTH	In this LMSCONV run, the library is both input and output library.
FORMAT	This operand defines the type of library to be assigned. The operand is needed only where a library is to be created and the default value PL is not appropriate. LMSCONV knows the library type of existing libraries.
=PL	Program library.
=OML	Object module library.
=OSM	Source program or macro libraries (Old source/macro library).
STATE	Defines whether the library is to be created, whether it must already exist or whether it is to be created if necessary.
=O[LD]	The library must already exist.
=N[EW]	The library must be newly created. If the library already exists, the statement is rejected with an error message.
=A[NY]	The library is newly created if it does not already exist.

With DEL, NAM, UPD, DUP, LST and TOC, the library ID can be used to assign a library other than the default input library. This input library only applies while this statement is being executed; thereafter the default input library is assigned again.

Example

```
/ADD-FILE=LINK LINK-NAME=BEILINK,FILE-NAME=LMSCONV.BEI
/ADD-FILE-LINK LINK-NAME=LIB001,FILE-NAME=LMSCONV.EINB
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB FILE=LMSCONV.AUS,OUT,NEW
$LIB LINK=BEILINK,IN
 .
 .
 .
$LIB LID=(001),USAGE=IN
 .
 .
 .
$END
```

The library LMSCONV.BEI is assigned in an ADD-FILE-LINK command and linked as an input library with LMSCONV by means of LIB, using the file link name BEILINK and the operand value IN.

Output library LMSCONV.AUS is to be created (OUT,NEW). Its file name is specified in the LIB statement.

The third LIB statement, which assigns library LMSCONV.EINB as an input library using its library ID, closes library LMSCONV.BEI.

Assigning sequential libraries

Sequential libraries can only be assigned by LIB as input libraries. To create sequential libraries, it is necessary to use LIBOUT (lib),NEWLIB.

Existing sequential output libraries are assigned using LIBOUT (lib).

LIB (Format 2): Closing libraries

This format of LIB is used to close libraries. The input and output library assignments are lost.

Operation	Operands
LIB	C[LOSE][,{FILE=libraryname / LINK=filelinkname / [LIBRARY=]name / [LID=](lib)}]

LIB	Statement name.
C[LOSE]	requests the closing of libraries.
FILE=libraryname	specifies the fully qualified file name of the library to be closed. pathname can also be used instead of libraryname (see the "Commands, Volume 1" manual [1]).
LINK=filelinkname	specifies the file link name of the library to be closed. The library must have been assigned in an ADD-FILE-LINK command and must be known to LMSCONV.
LIBRARY=name	LMSCONV first tries to interpret "name" as the assigned file link name of the library to be closed. If it can find no such file link name, "name" is interpreted as libraryname. The file link name or library name must be known to LMSCONV.
LID=(lib)	specifies the library ID of the library to be closed. The library must have been assigned in an ADD-FILE-LINK command with the file link name LIBlib and must be known to LMSCONV.

Example

```
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB LMSCONV.TEST,BOTH
 .
$LIB C,LMSCONV.TEST
 .
$END
```

The library LMSCONV.TEST is assigned as an input/output library in the first LIB statement, and closed in the second.

LIB (Format 3): Displaying assigned libraries

This format of LIB provides information about the libraries used in the LMSCONV run.

The following information is output:

– library usage (input or output or both)
– library status (open or closed)
– library format
– any library ID assigned to it
– any file link name assigned to it
– file name of the libraries

Operation	Operands
LIB	?

LIBOUT
Assign sequential output library

The LIBOUT statement closes the current output library and assigns a new one.

Operation	Operands
LIBOUT	[{ (lib)[, { (vsn) / NEWLIB / NEWLIB(vsn) }] / libname / LINK=linkname }]

LIBOUT	Statement name.
lib	Library ID of the output library.
vsn	Volume serial number (6 characters).
NEWLIB	Creates a new library.
libname	Fully qualified file name of the output library.
linkname	File link name referring to the output library.

The LIBOUT statement

- closes the current output library
- cancels the assignment of the implicit output library
- defines the library specified as libname, lib or linkname as an output library.

This implicitly defined output library remains in force until the next LIBOUT. At the start of the LMSCONV run, and following a LIBOUT with no operands, the implicit output library remains undefined.
After an error, the implicit output library becomes undefined and must be reassigned.
The (vsn) suffix allows a volume serial number to be specified. It must be the same as the VSN of the assigned volume.
If a library ID (lib) or a file link name (LINK=...) is used in LIBOUT, an ADD-FILE-LINK command must be specified (see LIB).
A sequential library cannot be both input and output library at the same time. LIBOUT (without operands) causes the current output library to be closed. The same library can then be assigned as an input library.
If a tape is assigned for lib, a new library tape is always opened, i.e. new file header labels are written for the library.

For sequential libraries, NEWLIB must always be specified for LIBOUT.

The NEWLIB option is only permitted for files to be created from scratch. It causes the output library to be created as an empty library. The following attributes must be specified for the file in the ADD-FILE-LINK command:

Library	type	ACCESS-METHOD=	KEY-POSITION=	KEY-LENGTH=
Source	library	ISAM	5	8
Macro	library	ISAM	5	8
Modul	library	PAM		
Sequential	library	BTAM		

The first time an ISAM file is used as an output library, its type is defined as a macro or source program library.

LST
List elements

LST lists the specified elements of the assigned input library.

Depending on the value of PRT, the output medium is either

– the terminal and/or
– the system file SYSLST or
– a library element.

The scope and format of the listing are determined by the processing operands CSECT, FORMAT, INFO, PATH, REFERENCE and SLICE.

Operation	Operands
LISTx	{ elem[(lib)] / (lib) } [,...]

LSTx	Name of the statement, including the element type: S, M, R, C, J, P, D, X, H, L, F, U
elem	Element identifier of the element to be listed, or selector.
lib	ID of the input library.

As list elements (type P in program libraries and type S in source program libraries) can only be output to the system file SYSLST, PRT (LST) must also be specified.

Processing operands

CSECT	For type L, determines which control section is to be listed.
FORMAT	Defines whether the records are to be output in character, hexadecimal or mixed format.
INFO	Defines whether all records, specific record types, specific areas, or just the most significant element data are to be output.
PATH	For type L, defines the sub-LLM to be listed.
REFERENCE	Defines the reference names which the elements to be listed must contain. If the reference condition is not satisfied, the element is not listed. REFERENCE is only permitted with element type R. If REFERENCE is defined, R must be specified in LST.
SLICE	For type L, defines which slice of the element is to be listed.

Example

```
/ADD-FILE-LINK LINK-NAME=LIB003,FILE-NAME=TEST.LIB
/START-PROGRAM FROM-FILE=$LMSCONV
$PAR INFO=SUMMARY
$LSTS (3)
$END
```

The most significant data concerning all elements of library TEST.LIB is listed.

NAM
Rename elements

NAM renames the specified elements in the assigned input library. The name is only changed in the table of contents of the input library.

The NAM statement cannot be used for sequential libraries.

Operation	Operands
NAMx	{ elem,...[(lib)] / (lib) }>elemu[,{...}>...]

NAMx — Name of the statement, including the element type: M, R, C, L

elem — Identifier of the element to be changed, or selector.

elem can also be a name that does not satisfy the LMSCONV conventions, in order to enable such elements to be processed.

elemu — New element identifier; a constructor is also permitted.

lib — ID of the input library.

Processing operands

DESTROY — Defines whether a data destruction indicator is set in the renamed elements.
(Only permitted for program libraries.)

OVERWRITE — Governs the overwriting of elements of the same name.

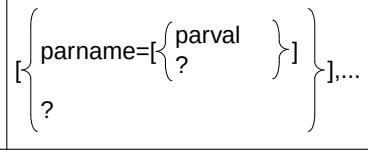
Example

```
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB PROD.LIB,BOTH
$NAMM MAXOUT>MACOUT
$END
```

The MAXOUT macro is renamed MACOUT.

PAR
Set processing operands

PAR sets the processing operands. A processing operand can define both a processing mode and also the values required for processing.

Operation	Operands
PAR	

PAR	Statement name.
parname	Name of the processing operand (see section 6.5.5, "Processing operands")
parval	Values of the processing operand (see section 6.5.5, "Processing operands")
?	The current values of all processing operands, or of the processing operand specified as parname, are listed.

Default values are defined for all processing operands. At the start of an LMSCONV run, all operands are set to these default values. (The defaults are specified in the description of each operand under the heading "Processing operands".) If no value is specified for an operand in the PAR statement, or if the value is incorrect, the corresponding default value is used. The processing operands can be specified in any order. If any operand is repeated, the last value specified is always used. The values specified remain in force until a new value is specified explicitly or until the defaults are reset by specifying a PAR without operands. The descriptions of the individual functions specify which processing operands affect the function.

You can find a summary of the processing operands under "Table of all processing operands".

PRT
Control log output

PRT defines the output medium for LMSCONV logs.

The output media are:

– the terminal
– the system file SYSLST
– a library element

If the log is output to an element in a program library, LMSCONV creates a type P element; if the element is in a source program library, LMSCONV creates a type S element.
If the library to which the element is written is a source program library, it must not be the default input or output library for the current LMSCONV run.

Operation	Operands
PRT	{ (LST) | ([SYS]OUT) | (CON) | (BOTH) | elem[(lib)] | ? }

PRT	Statement name.
LST	Outputs the log to the system file SYSLST.
{ SYSOUT | CON }	Outputs the log to the system file SYSOUT (i.e. to the terminal in interactive mode).
BOTH	Outputs the log both to the terminal and to the system file SYSLST.
elem	Outputs the log to the library element elem.
lib	ID of the library to which the element is to be written.
?	The current value is listed.

Processing operands

LOG	Controls the scope of the log; to obtain a log, you must set PAR LOG=MED or PAR LOG=MAX (default LOG=MIN).
OVERWRITE	Governs the overwriting of elements with the same name in the output library.

DESTROY — Defines whether a data destruction indicator is to be set in the list element.
(Only possible for program libraries.)

Example

```
/ADD-FILE-LINK LINK-NAME=LIB002,FILE-NAME=PROT.LIB
/START-PROGRAM FROM-FILE=$LMSCONV
$PRT PROELEM(2)
$LIB EINAUS.LIB,BOTH
  :
  :
$END
```

The log for this LMSCONV run is written to element PROELEM in library PROT.LIB.

RST
Exit TEST mode

RST switches back to RUN mode after TEST mode has been operating due to an error. The current input and output libraries are closed. They become "undefined" and must be reassigned.

Operation	Operands
RST	[STOP]

RST — Statement name.

STOP — RUN mode is resumed.
The internal LMSCONV termination flag is not deleted.

If the termination bit is set when LMSCONV terminates, LMSCONV ends with TERMJ instead of TERM.

If this operand is omitted and the termination flag has been set, the flag will be deleted.

SEL
Output elements to files

SEL outputs library elements to files.

LMSCONV creates

– files in accordance with the stored file attributes (PAR KEY=YES)
– files in accordance with the entry in the task file table (TFT) if file link names are being used
– files in accordance with the catalog entry.

If the ISAM keys for a file were included in the element (PAR KEY=YES), the ISAM keys are also output by SEL.

Elements of type C and PAM files stored as type X are selected into PAM files, while other elements are normally output to ISAM files with KEY-POS=5 and KEY-LEN=8.

The file created receives the CCSN of the source element as its CCS (coded character set) catalog attribute.

Operation	Operands
SELx	elem[> { [prefix.](name).[suffix] | filename | LINK=filelinkname }]

SELx — Name of the statement, including the element type:
M, R, C, X, S, J, P, D.

C applies only to BS2000 load modules.

elem — Element identifier of the element to be output.
A selector is permitted if
– “filename” is not specified
– in the expression [prefix.](name)[.suffix], a constructor is specified for "name".

filename — Fully qualified file name of the file to be created.
Constructors are not permitted.
pathname can also be specified for filename.

LINK=filelinkname — File link name referencing a file assigned in an ADD-FILE-LINK command.

[prefix.](name)[.suffix]	If a selector was specified for the element identifier, the expression may select several elements from different files. In this case, "name" must be a constructor.
prefix.	Specifies the common prefix of the files to be created. "prefix" must finish with a period.
.suffix	Specifies the common suffix of the files to be created. "suffix" must start with a period.
name	Specifies the partial name which combines with "prefix" and/or "suffix" to specify one or more fully qualified file names. If a constructor is specified for "name", this part of the file names is made up of the element names.

Note

Permissible element names are not always permissible file names.

Processing operands

OVERWRITE	Defines whether a file of the same name is overwritten, not overwritten, or extended by the records from the input element.
PHASE	Defines whether an NK or PK phase (load module) is created.

Creating ISAM files

If elements are output to ISAM files and no ISAM keys exist yet, LMSCONV generates the ISAM keys as follows:

LMSCONV generates the ISAM keys with an initial value of 1000 and an increment of 1000. The generated values are stored right-justified in the ISAM key.

Notes

- Elements of type R are output up to the END record. Any subsequent records are ignored.
- Update journal records (TXTP) are not output with type C elements.
- The record length is only specified for fixed-length records. For variable-length records it has a value of 0.
- Where SEL and PAR OVERWRITE=YES are specified, and a file already exists with the same name as the file to be created, the existing cataloged file must have the same file attributes as the file to be created.

Example 1

```
/ADD-FILE-LINK LINK-NAME=AUSD,FILE-NAME=AUSDAT,ACCESS-METHOD=*SAM,
               RECORD-FORMAT=*VARIABLE
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB EIN.BIB,IN
$SELS ELEM1>LINK=AUSD
$END
```

SEL causes element ELEM1 to be output to file AUSDAT.

Example 2

If all elements in a library are to be output with their names, the following statement must be specified:

```
$SEL[x] *>(*)
```

Possible errors

There are two PLAM library formats, one being the original NK2 format and the other being the new NK2 format with a minimum allocation unit of 8 or 64 Kbytes. In this context, the following possible errors must be borne in mind with regard to the use of the SEL statement for file output:

1a) If the library element contains an attribute record with the original BUFFER-LENGTH specification (e.g. following an ADD with PAR KEY=YES or for original UPAM files such as PLAM library files) and if BUFFER-LENGTH was explicitly specified for the target library in the TFT by means of ADD-FILE-LINK or directly in the catalog - i.e this specification is always evaluated - the following problems may occur:

SAM/ISAM file

If the element records are too long for the specified buffer length, a DMS error results.

UPAM file

When UPAM files are created, LMSCONV packs a logical block (other than the last block) with 2-Kbyte units and outputs this block only with UPAM.

If BLK-CONTR=DATA is specified, each logical block (BUFFER-LENGTH) starts with a 12-byte check field (CF). If the specified buffer length does not tally with that stored, data from DMS may be overwritten with the check field, thus rendering the file unusable.

If BLK-CONTR=NO is specified, unusable files may be generated if the buffer length is changed (e.g. PLAM files).

LMSCONV always issues a warning if there is any discrepancy in the buffer lengths (between user specification and stored value).
The system always attempts to create the file nevertheless.

1b) If the library element includes an attribute record with the original BUFFER-LENGTH specification (e.g. following an ADD with PAR KEY=YES or for original UPAM files such as PLAM library files), and if **no** buffer length is explicitly specified for the target file or the value is unknown, i.e. the buffer length from the attribute record is used, the following must be borne in mind:

If n in STD(n) is odd, LMSCONV increments to n+1.

2a) The element **does not** contain an attribute record (e.g. for phase elements) and BUFFER-LENGTH is explicitly specified for the target file, the same applies as described in 1a above.
When phases are generated, BUFFER-LENGTH specifications ≠ STD(1) or STD(2) produce errors.

2b) If the element **does not** include an attribute record (e.g. for phase elements) and **no** buffer length is explicitly specified for the target file or the value is unknown, the following applies:

- For phases, BUFFER-LENGTH is derived from the current environment, i.e. BUFFER-LENGTH = STD(1) on NK2 disks and BUFFER-LENGTH = STD(2) on NK4 disks. As regards contents, there is no difference between the phases.
- For all other types of element, BUFFER-LENGTH is calculated on the basis of the maximum record length.

SYS
Issue system commands

SYS allows system commands to be issued that are subsequently processed by the CMD macro. The program status is not exited. If no operand is specified, SYS operates as a HOLD-PROGRAM command. Program mode is re-entered by means of the RESUME-PROGRAM command.

Operation	Operands
SYS	[{'systemcommand' / systemcommand}]

SYS — Statement name.

systemcommand — Name of a command with the mandatory or desired operands. The system command is passed unchanged to the command processor.

TOC
List table of contents of library

TOC outputs the table of contents entries for the specified elements or for the complete library. The processing operand REFERENCE enables the table of contents listing to be restricted to those elements containing a particular reference name.

Operation	Operands
TOCx	[{ elem[(lib)] / (lib) } ,...]

TOCx — Name of the statement, including the element type:
S, M, R, C, P, J, D, X, H, L, F, U

* Indicates all element types (not permitted for tape libraries).

elem — Element identifier or selector.

lib — ID of the input library.

Processing operands

REFERENCE — Specifies the reference names that the elements to be output must contain. If the reference condition is not satisfied, the element is not output. REFERENCE is only permitted for type R elements. R must be specified with TOC if REFERENCE is specified.

SORT — Defines the sort criteria for the table of contents list.

Default value: The table of contents list is output sorted by name, version number and date.

Tables of contents that are too large to be sorted can be sorted as a whole by setting job switch 9.

Example

```
/ADD-FILE-LINK LINK-NAME=LIB001,FILE-NAME=A.LIB
/START-PROGRAM FROM-FILE=$LMSCONV
$LIB EINAUS.LIB,BOTH
$TOC* *
$PAR SORT=V
$TOCS (1)
$END
```

The ADD-FILE-LINK command assigns the source program library A.LIB. The program library EINAUS.LIB is assigned as the default input/output library. Since it is a program library, TOC can specify * as the element type. With other types of library, TOC must specify one particular element type. All type S elements in library A.LIB are listed sorted by version.

Note

To output the complete contents of a library (all elements and all versions) either TOC or TOC* */* must be specified. This works for all types of library for which the type '*' is permitted.

UPD
Update object/load modules and LLMs

UPD updates the specified element in the assigned input library. The updated element is then written to the assigned output library. It can be given a new element identifier.

UPD has various substatements for updating object modules, load modules and link and load modules. The substatements are read from the statement stream immediately following the UPD statement up to the *END statement.

The input and output libraries may be identical.

Three formats are available for UPD:

– UPDR (Format 1)

– UPDC (Format 2)

– UPDL (Format 3)

Format 1

Updating object modules

Operation	Operands
UPDR	elem[(lib)][>elemu]

UPDR	Name of the statement, including element type R.
elem	Full element identifier of the element to be updated, or a selector.
lib	ID of the input library.
elemu	Element identifier of the output element or constructor.

Processing operands

OVERWRITE	Specifies whether elements with the same name in the output library are to be overwritten. However, this processing operand is not evaluated if: ➤ input library=output library and elem=elemu. In this case the input element is overwritten.
DESTROY	Specifies whether a data destruction indicator is set in the output element. (Only permitted for program libraries.)
STRIP	Specifies which record types are to be excluded during updating.

LMSCONV first collects the UPD substatements, at the same time checking their syntax.

After the *END statement, LMSCONV checks that the substatements can be executed and then executes them.

Overview of the update statements for UPDR

Sub-statement	Operands	Meaning
*BAS	baseaddr	Defines a base address
*CON	checkno	Defines the cross-check number
*COR	[csectname,][baseaddr+]address, [[{C / X / B}] ‘checktext‘=[:=]][{C / X / B}] ‘corrtext‘ [,ID=‘ident‘][,CONTROL=number]	Corrects text records
*DEL	{rectype / (rectype,...) / TXTP[,ID=‘ident‘]}	Deletes parts of object modules
*END		Terminates update entries
*ID	‘ident‘	Defines the identification
*INS	INCLUDE (module,...)[,library]	Enters an INCLUDE record
*INV	{REP / COR[,ID=‘ident‘]}	Converts updates
*NAM	{CSECT: / ENTRY: / EXTERN: / COMMON:} nameold,nemenew	Renames symbols
*REM	[ID=‘ident‘]	Undoes updates

Sub-statement	Operands	Meaning
*REP	[csectname,][baseaddr+]address, [[<{C / X / B}] 'checktext'=[:=]][{C / X / B}] 'corrtext' [,CONTROL=number]	Inserts a REP record
*SET	{csectname / *} [,PRIV={Y / N}][,PUBLIC={Y / N}] [,VISIBLE={Y / N}][,READONLY={Y / N}][,PAGE={Y / N}] [,RESIDENT={Y / N}][,RMODE={24 / ANY}][,AMODE={24 / 31 / ANY}]	Modifies control section attributes

Description of the update statements for UPDR

*BAS
Define base address

*BAS defines a base address. This base address is then added to the address in a following *COR unless an explicit base address is specified.

Operation	Operands
*BAS	baseaddr

baseaddr — Defines the base address (hexadecimal).
$0 \leq$ baseaddr $\leq$ 7FFFFFFF

*CON
Define cross-check number

*CON defines the cross-check number for the complete update run. If more than one *CON is specified, the last value always applies.

Operation	Operands
*CON	checkno

checkno — Defines the cross-check number (hexadecimal).
$0 \leq$ checkno $\leq$ 7FFFFFFF

Note

Check numbers provide extra security when performing or transferring updates. During the update run, LMSCONV calculates a check number from the characters in each update statement, and from this calculates the cross-check number. These values are also calculated in TEST mode. The check numbers are output as each update statement is logged, and the cross-check number is output at the end of the update log. When the update is transferred or actually executed in RUN mode, the check numbers and cross-check number should also be transferred. When updates are to be executed, LMSCONV compares the numbers specified with those calculated. If they do not match, no updates are performed.

*COR
Correct text records

*COR corrects text records within an object module and generates an update journal record (TXTP record) containing the original contents of the text area.

Operation	Operands
*COR	[csectname,][baseaddr+]address, [[{C / X / B}]'checktext'=[:=]][{C / X / B}]'corrtext' [,ID='ident'][,CONTROL=number]

csectname — Specifies the name of a CSECT.
If a csectname is specified, corrections are only made within that CSECT. With prelinked modules in the new format (with full ESD), corrections must always be made using a specific CSECT name.

baseaddr — Specifies the base address (hexadecimal). This base address only applies to this *COR. If no base address is specified, the value specified for *BAS is used.

$0 \leq$ baseaddr $\leq$ 7FFFFFFF

address — Specifies the relative address.
baseaddr + address specifies the absolute address in the object module, or the CSECT-relative address if csectname is specified.

$0 \leq$ baseaddr+address $\leq$ 7FFFFFFF

C'checktext' — Specifies the check text in character format. Any quotes in the text must be written twice.
checktext must not be more than 50 characters.

X'checktext' — Specifies the check text in hexadecimal form.
checktext must not be more than 50 bytes long.

B'checktext' — Specifies the check text in binary form.
checktext must not be more than 50 characters long.

The original text for comparison with the check text is taken from the existing TXT records for this area. If there is more than one text for the same address, the last one applies.

=:= — Check text and correction text must be the same length.

=	Check text and correction text may be different lengths.
C’corrtext’	Specifies the correction text in character format. Any quotes in the text must be written twice. corrtext must not be more than 50 characters long.
X’corrtext’	Specifies the correction text in hexadecimal form. corrtext must not be more than 50 bytes long.
B’corrtext’	Specifies the correction text in binary form. corrtext must not be more than 50 characters long.
ID=’ident’	Specifies an identification in character format. ident must not be more than 8 characters long. This identification only applies to this *COR. If this operand is omitted, the specification made in the *ID substatement is used.
CONTROL=number	Defines a local check number (hexadecimal). $0 \leq$ number $\leq$ FFFF

If no TXT record exists, or if it exists only for part of the correction, LMSCONV generates a TXT record and inserts it in the module.

Several text records may exist for one address (e.g. through ORG statements). In such cases, more than one text record may have to be modified.

No *REP statement may be specified within the set of correction statements, otherwise the text corrections may be overwritten later during linkage.

*DEL
Delete sections of object modules

*DEL excludes the following record types from the input element:

– ISD records
– LSD records
– REP records
– INCLUDE records
– TXTP records
– DSDD records

Operation	Operands
*DEL	{ rectype (rectype,...) TXTP[,ID='ident'] }

rectype
Specifies the record type to be excluded from transfer from the input element to the output element.
The permitted record types are:

– ISD
– LSD
– REP
– INCLUDE
– TXTP
– DSDD

ID='ident'
Specifies an identification in character format. This identification only applies to this *DEL.
ident must not be more than 8 characters long.

If this operand is omitted, the specification made in the *ID substatement is used.

Note

REP, INCLUDE and TXTP records should not be deleted without thorough consideration of the consequences.

*END
Terminate update input

*END terminates the sequence of update statements. LMSCONV then checks that all statements are executable, and executes the set of statements.

Operation	Operands
*END	

*ID
Define identification

*ID defines a global identification. It applies to all statements in which no local identification is specified.

Operation	Operands
*ID	['ident']

'ident' Specifies the global identification.
ident must not be more than 8 characters long.

If this operand is not specified, 8 spaces are used as the default.

*INS
Insert INCLUDE record

*INS inserts an INCLUDE record in an object module. The INCLUDE record is evaluated by the dynamic binder loader (DBL).

Operation	Operands
*INS	INCLUDE (module,...)[,library]

module Name of the object module to be included. If only one object module is specified, the parentheses can be omitted. module must not be more than 8 characters long.

library Name of the program or object module library containing the specified object module(s). If this operand is omitted, DBL assumes the TASKLIB.

Notes

- The complete "(module,...)[,library]" specification must not be more than 71 characters long.
- LMSCONV checks neither the existence of the specified object modules nor the library specification.

*INV
Convert updates

*INV converts either REP records to text corrections or text corrections to REP records.

INV (Format 1): Converting REP records to text corrections

All REP records in the object module are converted to text corrections. This means that REP records need not be processed during linking and loading. The converted REP records are removed from the object module. LMSCONV creates update journal records for corrected text records. If specified for prelinked modules in the new format, this statement is rejected with an error message.

Operation	Operands
*INV	REP

Note

With OML libraries, only 409 TXTP records can be created by *INV REP.

INV (Format 2): Converting text corrections to REP records

All text corrections, or text corrections with a particular identification in an object module, for which an update journal record exists, are converted into REP records (for possible problems in prelinked modules see *REP). The update journal records are then deleted. If specified for prelinked modules in the new format, this statement is rejected with an error message.

Operation	Operands
*INV	COR[,ID=‘ident‘]

ID=’ident’ Specifies an identification. ident must not be more than 8 characters long. If this operand is not specified, the specification made in the *ID substatement is used. If nothing was specified for *ID either, all text corrections are converted.

*NAM
Rename symbols

*NAM modifies the name of a CSECT, ENTRY, EXTRN or COMMON. Every time renaming takes place, ESD records are modified. LMSCONV checks the uniqueness of the names within all ESD records and, unlike LMR, rejects the rename attempt if the new name already exists.

Operation	Operands
*NAM	{ CSECT: / ENTRY: / EXTERN: / COMMON: } nameold,namenew

nameold — Old name of the symbol. The name must be specified in full.

namenew — New name of the symbol. The name must be specified in full. namenew must not be more than 8 characters long.

Note

Masked CSECT names can also be renamed.

*REM
Undo updates

*REM undoes (reverses) all updates or text corrections of a given identification for which an update journal record exists. Once an update has been undone, the corresponding journal record is deleted.

Operation	Operands
*REM	[ID='ident']

ID='ident' — Specifies the local identification character by character. ident must not be more than 8 characters in length.

If this operand is not specified, the *ID specification is valid.

*REP
Insert REP record

*REP inserts REP records into the object module. These REP records are evaluated by the dynamic binder loader (DBL).

Operation	Operands
*REP	[csectname,][baseaddr+]address, [[{C / X / B}]'checktext'=[:=]][{C / X / B}]'corrtext' [,CONTROL=number]

csectname
: Specifies the name of a CSECT.
This entry is not valid for prelinked modules in the new format. They can only be updated via absolute addresses.

baseaddr
: Specifies the base address (hexadecimal). This base address applies to this *REP only. If baseaddr is not specified, the value specified for *BAS is used. If the baseaddr operand is used, the value specified for *BAS is ignored.

  $0 \leq$ baseaddr $\leq$ FFFFF

address
: Specifies the relative address.
baseaddr + address specifies the absolute address in the object module.

  $0 \leq$ baseaddr + address $\leq$ FFFFF

  The address must lie within the module's text area.

C'checktext'
: Specifies the check text in character format. A quote in the text must be written twice.
checktext must not be more than 50 characters long.

X'checktext'
: Specifies the check text in hexadecimal form.
checktext must not be more than 50 bytes long.

B'checktext'
: Specifies the check text in binary form.
checktext must not be more than 50 bytes long.

  The original text for comparison with the check text is formed from the TXT records that exist for these areas. If there are several texts for the same address, the last of the texts applies.

=:=	checktext and corrtext must be the same length.
=	checktext and corrtext may be of different lengths.
C’corrtext’	Specifies the correction text in character format. A quote in the text must be written twice. corrtext must not be more than 50 characters long.
X’corrtext’	Specifies the correction text in hexadecimal form. corrtext must not be more than 50 bytes long.
B’corrtext’	Specifies the correction text in binary form. corrtext must not be more than 50 characters long.
CONTROL=number	Specifies the local check number (hexadecimal). $0 \leq$ number $\leq$ FFFF The REP record is only inserted if the check number calculated by LMSCONV is equal to the check number specified here.

Note

Unlike with *COR, LMSCONV does not check whether REP records already exist for the update area. A prelinked module should always be corrected without a CSECT and check text being specified, i.e. it should be corrected by means of relative addresses within the prelinked module. The full functionality can only be used for object modules resulting from a compiler or assembler run.

*SET
Modify control section attributes

*SET modifies control section attributes.

Operation	Operands
*SET	{csectname | *} [,PRIV={Y | N}][,PUBLIC={Y | N}] [,VISIBLE={Y | N}][,READONLY={Y | N}][,PAGE={Y | N}] [,RESIDENT={Y | N}][,RMODE={24 | ANY}][,AMODE={24 | 31 | ANY}]

csectname — Name of the control section whose attributes are to be modified.

* — Specifies that the attributes are to be modified in all control sections.

PRIV

=Y — Specifies that only privileged system routines may access the specified control sections.

=N — No access restrictions.

PUBLIC

=Y — The specified control sections are shareable.

=N — The specified control sections are not shareable.

VISIBLE

=Y — The specified control sections are not masked. A secondary name record is created for these sections, and the names are entered in the secondary name directory.

Note

For an explanation of masking, and primary and secondary names, see the "BINDER" manual [3].

=N — The specified control sections are masked. In this case, no secondary name record is created, nor are the names placed in the secondary name directory. Any existing secondary name record is deleted.

If all the control sections of an object module are masked, a library element without a secondary name entry is created. This object module can only be retrieved via primary names. LMSCONV outputs a warning.

The module name can, however, be derived from the first control section name by means of all the ESD records, as masked control sections are also used here.

Note

Object modules only having masked control sections cannot be processed by the linkage editor (e.g. exclusion of an object module in the autolink function).

READONLY

=Y — Specifies that the specified control sections can only be read at program execution time.

=N — Allows the specified control sections to be written to during program execution.

PAGE

=Y — Specifies that the specified control sections are to be aligned on a page boundary, i.e. the load address must be a multiple of decimal 4096 or hexadecimal 1000.

=N — Ignores page boundaries. The control sections start at the next double word address available at link time.

RESIDENT

=Y — Specifies that the specified control sections are to be loaded into class 3 memory and remain resident there.

=N — Specifies that the specified control sections are not to be loaded into class 3 memory.

RMODE

=24 — Specifies that the specified control sections are to be loaded into the address space below the 16-Mb boundary.

=ANY — No restriction applies.

AMODE

=24	Specifies that the specified control sections are to be executable in 24-bit mode.
=31	Specifies that the specified control sections are to be executable in 31-bit mode.
=ANY	Any mode.

Note

At least one attribute must be specified, otherwise an error message is issued.

Other format of UPDR

Operation	Operands
UPDR	elem[(lib)][;number]

For a description of UPDR, elem and lib see format 1 of the UPD statement.

number	Specifies the cross-check number in hexadecimal form. 0 ≤ number ≤ FFFFFF

For the operation of the processing operands see the new format of UPDR.

Update journal

LMSCONV writes an update journal record (TXTP record) for every existing record that is changed. This update journal record contains the original text of the updated record.

Description of the update statement in the old format

```
L(address)[bitnumber][,[[v]'text1'=[:=]][v]'text2'][,number]

or the short form:

L(address)bitnumber[.number]
```

If just one bit is to be set to 1, the short form can be used. Update statements consisting only of an address are not permitted.

Address type:	Module address.
address	Hexadecimal address, max. 8 characters. This address can be taken from the compiler listing or module listing without recalculation. Leading zeros can be omitted.
bitnumber	Decimal number, max. 3 positions. This specifies the number of a bit in the field defined by "address". The bits are numbered from the left, starting at 1.
v	X: the text is in hexadecimal form. B: the text is in binary form. v entry omitted: the text is alphanumeric or consists of special characters.
text1	Check text, max. 50 characters. A quote in alphanumeric text must be written twice. If the text is hexadecimal, there must be an even number of characters. The check text is always compared with the original text in the object module.
=:=	Check text and correction text must be the same length.
=	Check text and correction text may be of different lengths.
text2	Correction text, as text1.
number	Check number, 4 hexadecimal characters.

Note

If the module to be updated by this update statement is a prelinked module in the new format, the statement is rejected with an error message.

If a bit number is specified, v=B must be set and vice versa.

The check/correction text area is formed from the specified address and the length of the text specified in the update statement. If a check text is specified, it is compared with the check text area in the element; if it matches, the update is executed and, depending on the STRIP operand, an update journal record may be written. If an address appears more than once in the element, the last text addressed is used for checking; if it matches, all text positions in the element that fall in the update text area are modified.

UPDC (Format 2): Updating load modules (BS2000 phases)

Operation	Operands
UPDC	elem[(lib)][>elemu]

UPDC — Name of the statement with element type C.

elem — Full element identifier of the element to be updated. A selector is not permitted.

lib — ID of the input library.

elemu — Element identifier of the output element or constructor.

Processing operands

OVERWRITE — Specifies whether elements with the same name in the output library are to be overwritten.
However, this processing operand is not evaluated if

- input library=output library and elem=elemu.

In this case the input element is overwritten.

DESTROY — Specifies whether a data destruction indicator is set in the output element.
(Only permitted for program libraries.)

STRIP — Specifies which record types are to be excluded during updating.

Overview of update statements for UPDC

Sub-statements	Operands	Meaning
*BAS	baseaddr	Defines a base address
*CON	checkno	Defines the cross-check number
*COR	[segment,][baseaddr+]address, [[{C / X}]'checktext'=[:=]][{C / X}]'corrtext' [,ID='ident'][,CONTROL=number]	Corrects text records
*DEL	TXTP[,ID='ident']	Deletes update journal records
*END		Terminates update input
*ID	'ident'	Defines identification
*REM	[ID='ident']	Undoes updates
*SEG	{segment / %ROOT}	Defines a segment

Description of the update statements

*BAS
Define base address

*BAS defines a base address. This base address is then added, with the address in a following *COR, to the absolute address in the load module unless an explicit base address is specified in *COR.

Operation	Operands
*BAS	baseaddr

baseaddr — Defines the base address (hexadecimal).
$0 \leq$ baseaddr $\leq$ 7FFFFFFF

*CON
Define cross-check number

*CON defines the cross-check number for the complete update run. If more than one *CON is specified, the last value always applies.

Operation	Operands
*CON	checkno

checkno — Defines the cross-check number (hexadecimal).

The cross-check number is calculated by LMSCONV and logged at the end of the UPD run.

$0 \leq$ checkno $\leq$ 7FFFFFFF

*COR
Correct text records

*COR corrects text records within a segment and generates an update journal record containing the original contents of the text area.

Operation	Operands
*COR	[segment,][baseaddr+]address, [[{C|X}]'checktext'=[:=]][{C|X}]'corrtext' [,ID='ident'][,CONTROL=number]

segment — Specifies the name of the segment to be updated. Updates can then only be made within the segment, and not outside segment boundaries. segment must not be more than 8 characters long.

The segment name specified here has precedence over the name defined in *SEG. %ROOT selects the root segment.
If this operand is not specified, the specification made in the *SEG substatement is used.

baseaddr — Specifies the base address (hexadecimal). This base address only applies to this *COR. If no base address is specified, the value specified for *BAS is used.

$0 \leq$ baseaddr $\leq$ 7FFFFFFF

address — Specifies the relative address.
baseaddr + address specifies the absolute address in the load module.

$0 \leq$ baseaddr+address $\leq$ 7FFFFFFF

C'checktext' — Specifies the check text in character format. Any quotes in the text must be written twice.
checktext must not be more than 50 characters.

X'checktext' — Specifies the check text in hexadecimal form.
checktext must be more than 50 bytes long.

=:= — Check text and correction text must be the same length.

= — Check text and correction text may be different lengths.

C’corrtext’	Specifies the correction text in character format. Any quotes in the text must be written twice. corrtext must not be more than 50 characters long.
X’corrtext’	Specifies the correction text in hexadecimal. corrtext must not be more than 50 bytes long.
ID=’ident’	Specifies an identification in character format. ident must not be more than 8 characters long. This identification only applies to this *COR. If this operand is omitted, the specification made in the *ID substatement is used.
CONTROL=number	Defines a local check number (hexadecimal). The check number is calculated by LMS for every *COR statement. $0 \leq$ number $\leq$ FFFF

*DEL
Delete update journal records

*DEL excludes the update journal records (TXTP) from the input element.

Operation	Operands
*DEL	TXTP[,ID='ident']

ID='ident' Specifies an identification in character format.
ident must not be more than 8 characters long.

This identification only applies to this *DEL. If this operand is omitted, the specification made in the *ID substatement is used.

*END
End update input

*END terminates the sequence of update statements. LMSCONV then checks that all statements are executable, and executes the set of statements.

Operation	Operands
*END	

*ID
Define identification

*ID defines a global identification. It applies to all statements in which no local identification is specified.

Operation	Operands
*ID	['ident']

'ident' Specifies the global identification.
ident must not be more than 8 characters long.

If this operand is not specified, 8 spaces are used as the default.

*REM
Undo corrections

*REM undoes (reverses) all text corrections, or all text corrections for a particular identification, for which an update journal record exists. The update journal records are then deleted.

Operation	Operands
*REM	[ID='ident']

ID='ident' Specifies a local identification in character format.
ident must not be more than 8 characters long.

If this operand is omitted, the specification made in the *ID substatement is used.

*SEG
Define segment

*SEG defines a segment of a load module that is to be updated by a subsequent *COR.

Operation	Operands
*SEG	{ segment / %ROOT }

segment Specifies the name of the segment to be corrected.
segment must not be more than 8 characters long.

%ROOT Specifies that the root segment is to be corrected.

UPDL (Format 3): Updating link and load modules

Operation	Operands
UPDL	elem[(lib)]>[elemu]

UPDL	Name of the statement, including element type L.
elem	Full element identifier of the element to be updated, or selector.
lib	ID of the input library.
elemu	Element identifier of the output element or constructor.

Processing operands

CSECT	Defines the base for the displacement in the *COR substatement.
OVERWRITE	Specifies whether elements of the same name in the output library are to be overwritten. However, this processing operand is not evaluated if ➤ input library=output library and elem=elemu. In this case the input element is overwritten.
PATH	Defines the base for the displacement in the *COR substatement.
SLICE	Defines the base for the displacement in the *COR substatement.
STRIP	Defines which record types are to be excluded during the update.
LMSCONV	Collects the UPD substatements, checking – the syntax of the substatements – that there are no conflicts in the updates (overlapping) – the uniqueness of the symbols when renaming.

After the *END statement, LMSCONV checks that the substatements entered are executable, and then executes them.

Overview of update statements for UPDL

Sub-statements	Operands	Meaning
*COR	[csect,]displ, [[{C / X / B}]'checktext'=[:=]][{C / X / B}]'corrtext' [,ID='ident'][,CONTROL=number]	Corrects text records
*DEL	TXTP,[ID='ident']	Deletes update journal records
*END		Terminates update input
*ID	['ident']	Defines identification
*REM	[ID='ident']	Undoes updates

*COR
Correct text records

*COR corrects text records in a link and load module.

Operation	Operands
*COR	[csect,]displ, [[{C / X / B}]'checktext'=[:=]][{C / X / B}]'corrtext' [,ID='ident'][,CONTROL=numberl]

csect

32-character CSECT name.
If csect is specified, all CSECTs with the specified name are corrected.

The priority for evaluating the names is as follows:

1. csect was specified in *COR: the CSECT processing operand is ignored.

2. PAR CSECT=xxx: only the PATH or SLICE processing operand may be set.
3. PAR PATH=xxx: the SLICE processing operand must not be set.
4. PAR SLICE=xxx: the PATH processing operand must not be set.

displ — Defines the relative address. displ specifies the absolute address in the LLM or, if csect is specified, the CSECT-relative address.

$0 \leq$ displ $\leq$ 7FFFFFFF

C'checktext' — Specifies the check text in character format. Any quotes in the text must be written twice.
checktext must not be more than 50 characters.

X'checktext' — Specifies the check text in hexadecimal form.
checktext must not be more than 50 bytes long.

B'checktext' — Specifies the check text in binary form.
checktext must not be more than 50 characters long.

The original text for comparison with the check text is taken from the existing TXT records for this area. If there is more than one text for the same address, the last one applies.

=:= — Check text and correction text must be the same length.

= — Check text and correction text may be different lengths.

C'corrtext' — Specifies the correction text in character format. Any quotes in the text must be written twice.
corrtext must not be more than 50 characters long.

X'corrtext' — Specifies the correction text in hexadecimal form.
corrtext must not be more than 50 bytes long.

B'corrtext' — Specifies the correction text in binary form.
corrtext must not be more than 50 characters long.

ID='ident' — Specifies an identification in character format.
ident must not be more than 8 characters long.

This identification only applies to this *COR. If this operand is omitted, the specification made in the *ID substatement is used.

CONTROL=number — Defines a local check number (hexadecimal).

The check number is calculated by LMS for each *COR statement.

$0 \leq$ number $\leq$ FFFF

*DEL
Delete update journal records

*DEL excludes the update journal records (TXTP) from the input element.

Operation	Operands
*DEL	TXTP[,ID='ident']

TXTP — Update journal records are to be excluded from the input element.

ID='ident' — Specifies an identification in character format.

ident must not be more than 8 characters long.

This identification only applies to this *DEL. If this operand is omitted, the specification made in the *ID substatement is used.

*END
Terminate update input

*END terminates the sequence of update statements. LMSCONV then checks that all statements are executable, and executes the set of statements.

Operation	Operands
*END	

*ID
Define identification

*ID defines a global identification. It applies to all statements in which no local identification is specified.

Operation	Operands
*ID	[‘ident‘]

’ident’ Specifies the global identification.
ident must not be more than 8 characters long.
If this operand is not specified, 8 spaces are used as the default.

*REM
Undo updates

*REM undoes (reverses) all text corrections, or all text corrections for a particular identification, for which an update journal record exists. The update journal records are then deleted.

Operation	Operands
*REM	[ID=‘ident‘]

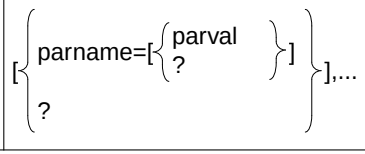
ID=’ident’ Specifies a local identification in character format.
ident must not be more than 8 characters long.
If this operand is omitted, the specification made in the *ID substatement is used.

6.5.5 Processing operands

Processing operands influence the execution of LMSCONV.

Where a processing operand is required to operate on a function, it must be set prior to issuing the corresponding statement.

Processing operands are set by means of PAR:

Operation	Processing operands
PAR	

The following two sections contain an alphabetical table of the processing operands followed by a detailed description of each of them.

The names of the processing operands and operand values may be abbreviated providing the abbreviation is unique within the processing operands.

The following processing operand influences the operation of LMSCONV, but not the individual functions:

TERMINATE Termination behavior in the event of an error

The following processing operand influences the operation of both LMSCONV and individual functions:

LOG Statement logging

The other processing operands only influence individual functions. The function descriptions indicate which operands affect which functions. You will find a summary in table form of the effect of the processing operands in section 6.4.1, "Control via processing operands".

If PAR is specified with no processing operands, all the processing operands are set to their default values.

If a processing operand is specified with no operand value (e.g. PAR LOG=), that processing operand is reset to its default value.

Table of all processing operands

Processing operands	Function
CS[ECT]=[{NAME / ?}]	Define CSECT
DEC[OMPRESSED]=[{YES / NO / ?}]	Define compression
DES[TROY]=[{YES / NO / ?}]	Control physical deletion
FO[RMAT]=[{C / X / SYMBOLIC / XC / REC / P / ?}]	Define record format
I[NFO]=[{ALL / SUMMARY / (rectype,...) / rectype[([#recbeg][{-#recend / :#number}])] / TXT[([addbeg][{-addend / :length}])] / TXTP[(['identu'][-'idento'])] / {PHY[SICAL] / LOGICAL[({NEXT / ALL})]} / ?}]	Define scope of output

Processing operands	Function
K[EY]=[<YES / NO / ?>]	Transfer file attributes and existing ISAM keys
LO[G]=[<MAX / MED / MIN / ?>]	Log statements
O[VERWRITE]=[<YES / ONLY / NO / V / D / EXTEND / ?>]	Overwrite elements with same name
PA[TH]=[<NAME / ?>]	Define path name
PH[ASE]=[<PK / NK / ?>]	Define phase format
RE[FERENCE]=[<name / ([name],[<CSECT / ENTRY / ALL>]) / ?>]	Define reference conditions
SL[ICE]=[<NAME / ?>]	Define slice
SO[RT]=[<U / [N] [V] [D] / R / ?>]	Sort table of contents

Processing operands	Function
STRIP=[{rectype / (rectype,...) / YES / NO / ?}]	Suppress records
TER[MINATE]=[{1 / 2 / 3 / 4 / 5 / 6 / 7 / ?}]	Termination behavior in event of error

Description of the processing operands

PAR CSECT
Define CSECT name

This processing operand defines a CSECT name for UPDL and LSTL.

Operation	Processing operands
PAR	CS[ECT]=[{NAME / ?}]

name — CSECT name, max. 32 characters.

With LSTL, only the CSECT with this name is listed.

With UPDL, this name is the base for the displacement in the *COR substatement.

? — The current value is listed.

PAR DECOMPRESSED
Control compression of macros and source programs

This operand has no effect on program libraries. For other libraries, it controls the compression of macros and source programs for output to libraries.

Operation	Processing operands
PAR	DEC[OMPRESSED]=[{YES / NO / ?}]

YES — All elements are output in decompressed form.

NO — All elements are output in compressed form.

? — The current value is listed.

The DECOMPRESSED processing operand affects: ADD, DUP

PAR DESTROY
Control physical deletion

The DESTROY processing operand determines whether data is overwritten with binary zeros on deletion, or whether space is simply released.

To work fully at the deletion stage, the operand must have been set when the element was originally stored. If set later, only variants created after the operand was set will be physically deleted.

Operation	Processing operands
PAR	DES[TROY]=[{YES / NO / ?}]

YES — All deleted data is overwritten with binary zeros before the storage space is released.

This means that

– data is destroyed when elements in program libraries are deleted.

– when elements are written to program libraries, a data destruction indicator is set so that when such elements are deleted later, the data is destroyed.

This operand value can only be used with elements in program libraries.

NO — Whenever data is deleted, only the space is released; data is not destroyed.

? — The current value is listed.

The DESTROY processing operand affects ADD, DEL, DUP, NAM, PRT, UPD.

PAR FORMAT
Define record format

The FORMAT processing operand defines the record format used when listing elements with LST. The possible formats are

– character
– hexadecimal
– character and hexadecimal side by side
– character and hexadecimal one above the other
– character, with the first character of the record contents treated as a print control character.

Operation	Processing operands
PAR	FO[RMAT]=[C / X / SYMBOLIC / XC / REC / P / ?]

C — Character format.

X — Hexadecimal format, only for elements of types

– R and C
– X where these elements contain PAM files. 2*4 4-byte blocks are output to the system file SYSLST and 2*3 4-byte blocks to the system file SYSOUT. For all other elements, this operand value has the same effect as the operand value REC.

SYMBOLIC — Formats the records differently for different element types:

Element type	Record format
S, M, J, D, X	Character format
P	Character format, with the first character of each record interpreted as a print control character.
R, L	The ESD, ISD, RLD, TXT, TXTP, REP and END information familiar to LMSCONV is output in edited form. Other information such as LSD and DSDD is output unedited, i.e. the record length field and, where applicable, the record number are also output. Continuous text information is not subdivided.
C	Character and hexadecimal side by side.

XC — Character and hexadecimal format side by side, only for elements of types

– R, L and C
– X where these elements contain PAM files.

For all other elements this operand value has the same effect as the value REC.

REC — Character and hexadecimal format one above the other, i.e. two lines are output for each element record, with the hexadecimal format in the second line. For type L, REC has the same effect as SYMBOLIC.

P — Character format, with the first character of the record contents treated as a print control character.

This type of output is intended only for printable text elements or list elements.

Elements of types R, C and L are represented as for operand value XC.

? — The current value is listed.

The FORMAT processing operand affects LST.

PAR INFO
Define scope of output

The INFO processing operand defines the range of records to be selected when listing elements using LST. The possible options are

– all records
– the most significant element data
– specific record types
– a particular range within a record type.

Operation	Processing operands
PAR	I[NFO]=[{ ALL | SUMMARY | (rectype...) | rectype[([#recbeg][{-#recend | :#number}])] | TXT[([addbeg][{-addend | :length}])] | TXTP[([‘identu‘][-‘idento‘])] | {PHY[SICAL] | LOGICAL[({NEXT | ALL}]} | ? }]

ALL — All element records are output.

SUMMARY — The most significant element data is output, i.e.

– for text type elements (element types S, M, J, P, D, X) the user record types are output in tabular form followed by the number of records.
– for type R elements, the length of the object module together with the names, lengths and addresses of the CSECTs are output.
– for type C elements, the length of the load module together with the names, lengths and addresses of the segments are output.
– for type L elements, the complete logical structure is output.

rectype	The specified record type of a type R, L or C element is output. For text type elements (element types S, M, J, P, D, X) any record type entry is ignored, i.e. all records in the element are output. Possible record types are – for type R elements: ESD, ISD, LSD, TXT, RLD, TXTP, REP, INCLUDE, DSDD, REF, END. If REF is specified, all reference names associated with the object module are output. – for type C elements: ESD, ISD, LSD, TXT, RLD, TXTP. – for type L elements: ESVD, ESVR, TXT, LRLD, TXTP.
recbeg	Specifies the first record at which output of the specified record type is to start. This applies to type R elements only. 1 ≤ recbeg ≤ 2147483647
recend	Specifies the last record at which output of the specified record type is to stop. This applies to type R elements only. 1 ≤ recend ≤ 2147483647
number	Specifies the number of records of the specified record type that is to be output. This applies to type R elements only. 1 ≤ number ≤ 2147483647, where recbeg+number must be ≤ 2147483647.
addbeg	Specifies the start address of the area, in hexadecimal, from which the specified record type is to be output. This applies to type L, R and C elements only. 0 ≤ addbeg ≤ 7FFFFFFF
addend	Specifies the end address of the area, in hexadecimal form, up to which the specified record type is to be output. This applies to type L, R and C elements only. 0 ≤ addend ≤ 7FFFFFFF
length	Specifies the length, in hexadecimal form, of the address area to be output for the specified record type. This applies to type L, R and C elements only. 1 ≤ length ≤ 7FFFFFFF
identl	Specifies the lower identification boundary.
identu	Specifies the upper identification boundary.
PHYSICAL	The physical LLM structure is logged.

LOGICAL The logical LLM structure is logged.

NEXT	Only the next level down is logged.
ALL	The complete structure is logged.

? The current value is listed.

The INFO processing operand affects LST.

PAR KEY
Transfer file attributes and ISAM keys

The KEY processing operand specifies whether the existing ISAM keys and file attributes are to be transferred to the output element.

Operation	Processing operands
PAR	K[EY]=[<(YES / NO / ?)>]

YES The following file attributes are stored unchanged in the new element to be added: ACCESS-METHOD, BUFFER-LENGTH, PERFORMANCE, RECORD-FORMAT, RECORD-SIZE, USAGE, USER-ACCESS. If ACCESS-METHOD=ISAM, the ISAM keys and the following file attributes are stored in the new element to be added: LOGICAL-FLAG-LENGTH, PADDING-FACTOR, PROPAGATE-VALUE-FLAG, VALUE-FLAG-LENGTH.
This operand value is only allowed when processing program libraries, but not when processing source program libraries.

NO The file attributes and ISAM keys are not transferred. In this case, only ISAM files with KEY-POS=5, KEY-LEN≤16 and RECFORM=V can be transferred into the output element.

? The current value is logged.

Note

When LST is used to list an element containing ISAM keys, the values of KEY-POS and KEY-LEN are also output.

The KEY processing operand affects ADD.

PAR LOG
Log statements

The LOG processing operand governs the scope of the LMS log.

Operation	Processing operands
PAR	LO[G]=[{MAX / MED / <u>MIN</u> / ?}]

MAX	Complete log.
MED	Statements are only logged when in error. Positive acknowledgments are logged.
<u>MIN</u>	Only error messages, termination messages and negative acknowledgments are logged.
?	The current value is logged.

The LOG processing operand affects the whole LMSCONV run and UPD.

PAR OVERWRITE
Overwrite elements with same name

This processing operand controls the overwriting of elements with the same name in the output library, and files of the same name when using SEL. The OVERWRITE processing operand influences ADD, DUP, NAM, PRT, SEL and UPD.

Operation	Processing operands
PAR	O[VERWRITE]=[{YES / ONLY / <u>NO</u> / V / D / EXTEND / ?}]

YES	An element or file with the same name is overwritten. Note that the element name always includes the element version (see section called "Rules for element identifiers in program libraries").
ONLY	An element is only written if an element or file with the same name already exists.
NO	An element or file with the same name is not overwritten and the statement is not executed.
V	An element with the same name is overwritten only if new version number > old version number. For program libraries it has the same effect as NO.
D	An element with the same name is only overwritten if ➤ new date > old date.
EXTEND	The element or file is to be extended. This operand value only affects ADD and SEL. For other statements, EXTEND has the same effect as NO. An element or file is only extended if the element contains no ISAM keys and if the file attributes stored in the element correspond to the file attributes of the file, including the file name. Otherwise, ADD or SEL will be rejected with an error message.
?	The current value is logged.

PAR PATH
Define path name

This processing operand is used to define a path name for UPDL and LSTL.

Operation	Processing operands
PAR	PA[TH]=[{NAME / ?}]

name	Path name, max. 255 characters. With LSTL, only the sub-LLM with this path name is listed. With UPDL, the name is used as the base for the displacement in the *COR substatement.
?	The current value is logged.

The PATH processing operand affects UPDL and LSTL.

Note

If the SLICE processing operand is set, it can be reset by setting the PATH processing operand to 'UNDEFINED'.

PAR PHASE
Define phase format

This processing operand defines the format of the phase (load module) to be generated. Normally, PK phases are generated in the PAM key (PK) environment and NK phases in the non-key (NK) environment. This processing operand also allows NK phases to be generated in the PK environment.

Operation	Processing operands
PAR	PH[ASE]=[{PK / NK / ?}]

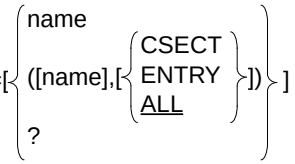
PK	The phase is generated in PK format if it is written to a PK disk, and in NK format if it is written to an NK disk.
NK	The phase is always generated in NK format.
?	The current value is listed.

The default value of the PHASE operand is defined by the class 2 option.

The PHASE processing operand affects SELC.

PAR REFERENCE
Define reference conditions

The REFERENCE processing operand defines conditions (reference conditions) under which elements are selected for processing. The reference condition consists of a reference name/attribute pair. This processing operand is only evaluated for type R elements in program libraries.

Operation	Processing operands
PAR	RE[FERENCE]=[]

name	Specifies the reference name. A selector is permitted. "name" must not be more than 32 characters long.
CSECT	Only elements having reference names with the CSECT attribute are to be processed.
ENTRY	Only elements having reference names with the ENTRY attribute are to be processed.
ALL	Only elements having reference names with any attribute are to be processed.
?	The current value is logged.

The entry PAR REFERENCE= resets any existing reference conditions to “undefined”.

The entry PAR REFERENCE=name has the same effect as
PAR REFERENCE=(name, ALL).

The entry PAR REFERENCE=(,...) has the same effect as PAR REFERENCE=(*,...).

The REFERENCE processing operand affects: LST, TOC, DUP, DEL.

Note

Element type R must be explicitly specified, otherwise the reference condition is ignored.

PAR SLICE
Define slice

This processing operand defines a segment for UPDL and LSTL.

Operation	Processing operands
PAR	SL[ICE]=[{NAME / ?}]

name — Slice name, max. 32 characters.

With LSTL, only the slice with this name is listed.

With UPDL, the name forms the base for the displacement in the *COR substatement.

? — The current value is listed.

Note

If the PATH processing operand is set, it can be reset by setting the SLICE processing operand to ’UNDEFINED’.

PAR SORT
Sort table of contents

The SORT processing operand defines the sort criteria for outputting the table of contents (cf. TOC).

Operation	Processing operands
PAR	SO[RT]=[{U / [N] [V] [D] / R / ?}]

U — Unsorted output:
the contents entries are output in the order in which they appear in the table of contents.

N — Output sorted by name.

V — Output sorted by version number.

D — Output sorted by date.

R — Output sorted by reference names as defined by the REFERENCE processing operand.

If no reference condition is defined using the REFERENCE processing operand, LMSCONV outputs the primary name directory.

? — The current value is logged.

N, V and D are the default values. They may be specified in any order and any combination.

If several sort criteria are specified, the TOC is output in the order in which they are specified.

The SORT processing operand affects TOC.

PAR STRIP
Suppress records

The STRIP processing operand defines which record types are not to be transferred from the input element to the output element. STRIP is evaluated for element types R, L and C only.

Operation	Processing operands
PAR	STRIP=[rectype / (rectype,...) / YES / NO / ?]

rectype
: Excludes the specified record type from transfer to the output element.

  Possible record types are

  – for type R elements:
    ISD, LSD, TXTP, REP, INCLUDE, DSDD
  – for type C elements:
    TXTP
  – for type L elements:
    TXTP

  Any other entry is ignored.

YES
: TXTP records are not transferred to the output element, i.e. the update journal is not transferred.

NO
: All records are transferred to the output element.

?
: The current value is logged.

The STRIP processing operand affects UPD for element types R, L and C, and DUP for element types R and C.

PAR TERMINATE
Termination action in event of error

The TERMINATE processing operand determines which situations are treated as errors and therefore set the internal LMSCONV abort flag.

If the abort flag is set on termination of LMSCONV, this leads in interactive mode to normal program termination, and in batch mode or procedures to a branch to STEP or ABEND or LOGOFF. In batch mode, TEST mode is always entered for serious errors, irrespective of the current value.

This processing operand also influences the behavior of LMSCONV after errors, i.e. it determines which errors cause LMSCONV to switch to TEST mode.

Operation	Processing operands
PAR	TER[MINATE]=[{1 / 2 / 3 / 4 / 5 / 6 / 7 / ?}]

1 TEST mode:
Set the abort bit on serious errors, i.e. errors which make continuation pointless (e.g. device error).

2 RUN mode, 3 TEST mode:
Set the abort bit on serious errors and also on other types of error (e.g. syntax errors in a statement, element could not be corrected).

4 RUN mode, 5 TEST mode:
As for 2 and 3, and also when a function cannot be executed because an element could not be found.

6 RUN mode, 7 TEST mode:
As for 4 and 5 and also when a function cannot be executed because an existing element was not allowed to be overwritten (OVERWRITE=NO) or the element could not be written because no element existed with the same name (OVERWRITE=ONLY).

? The current value is logged.

With the even values (2, 4, 6), the LMSCONV run continues in RUN mode after the occurrence of the TERMINATE condition; with the odd values (1, 3, 5, 7) TEST mode is entered except when LMSCONV is reading the statements interactively from the terminal.

Every TERMINATE condition that occurs is logged.

The TERMINATE processing operand influences the overall LMSCONV run.

6.6 Messages

The messages are supplied in a message file with the message class "LMC".

Messages are output as a 7-character code (LMSnnnn), together with their meaning (in English and German) and any action that can be taken by the user.

Messages can be viewed by means of `/HELP-MSG-INFORMATION LMCxxxx`.

In addition, the LMSCONV access routines output internal error codes (DMS, AMCB, PLAM).

The PLAM error codes can be interrogated using the command `/HELP-MSG-INFORMATION PLAxxxx`..

AMCB error codes are included in the meanings of the LMSCONV messages.

The messages can be found in the "System Messages" manual [16]/[17].

6.7 Comparison tables

6.7.1 LMSCONV compared with LMS

The following table compares the LMS and LMSCONV statements. LMSCONV has fewer statements than LMS. The functionality of the statements in terms of type handling is also more limited.

LMSCONV statement	Type												**LMS 3.0 statement**
	S	M	R	C	P	D	X	H	J	L	U	F	
ADD	=	=	=	=	=	=	=	-	=	-	-	-	ADD
	+	+	-	-	+	+	+	-	+	-	-	-	COM
	+	+	-	-	+	+	+	-	+	-	-	-	COR
	-	-	-	-	-	-	-	-	-	-	-	-	CTL
DEL	=	=	=	=	=	=	=	=	=	=	=	=	DEL
DUP	=	=	=	=	=	=	=	=	=	=	=	=	DUP
	+	+	-	-	+	+	+	-	+	-	-	-	EDR/EDT
END	-	-	-	-	-	-	-	-	-	-	-	-	END
LIB	-	-	-	-	-	-	-	-	-	-	-	-	LIB
LIBOUT	-	-	-	-	-	-	-	-	-	-	-	-	LIB
LST	=	=	=	=	=	=	=	=	=	=	=	=	LST
NAM	+	=	=	=	+	+	+	+	+	=	-	-	NAM
	-	-	-	-	-	-	-	-	-	-	-	-	NOP
	+	+	-	-	+	+	+	-	+	-	-	-	NUM
PAR	-	-	-	-	-	-	-	-	-	-	-	-	PAR
PRT	-	-	-	-	-	-	-	-	-	-	-	-	PRT
RST	-	-	-	-	-	-	-	-	-	-	-	-	RST
SEL	=	=	=	=	=	=	=	-	=	-	-	-	SEL
	-	-	-	-	-	-	-	-	-	-	-	-	SUM
	-	-	-	-	-	-	-	-	-	-	-	-	SUMADD
	-	-	-	-	-	-	-	-	-	-	-	-	SUMDEL
	-	-	-	-	-	-	-	-	-	-	-	-	SUMPRT
SYS	-	-	-	-	-	-	-	-	-	-	-	-	SYS
	-	-	-	-	-	-	-	-	-	-	-	-	TCH
TOC	=	=	=	=	=	=	=	=	=	=	=	=	TOC

LMSCONV statement	Type												**LMS 3.0 statement**
	S	M	R	C	P	D	X	H	J	L	U	F	
UPD	-	-	=	=	-	-	-	-	-	=	-	-	UPD
	-	-	-	-	-	-	-	-	-	-	-	-	USE
	-	-	-	-	-	-	-	-	-	-	-	-	$

\+ : applicable only to LMS

= : applicable to both LMS and LMSCONV

\- : not applicable

LMSCONV does not support all LMS processing operands. All the LMS processing operands are listed below with an indication of their usage in LMSCONV.

Processing operand	**LMS 3.0**	**LMSCONV**
BASE	+	-
CHECK	+	-
COMPARE	+	-
CSECT	+	+
DECOMPRESSED	+	+
DESTROY	+	+
ERRCONS	+	-
FCBTYPE	+	-
FORMAT	+	+
INFO	+	+
KEY	+	+
LCASE	+	-
LINE	+	-
LOG	+	+
LST	+	-
NEWFORM	+	-
OVERWRITE	+	+
PATH	+	+
PHASE	+	+
RANGE	+	-
REFERENCE	+	+

Processing operand	LMS 3.0	LMSCONV
SEGMENT	+	-
SLICE	+	+
SORT	+	+
STRING	+	-
STRIP	+	+
SUM	+	-
TERMINATE	+	+
TEST	+	-
TOC	+	-
TYPE	+	-
VALUE	+	-

- : not supported

+ : supported

The available LMSCONV statements do not support the following LMS functions:

– ADD from FMS libraries

ADDx FMS=fmslib(fmselem).....

– ADD from the LMSCONV statement stream

ADDx { CTL / CMD / SYSDTA }

– SEL in FMS libraries

SELx elem>FMS=fmslib(fmselem)

– Delta technique: the BASEVERSION operand must not be used (ADD, DUP).

– Target version: LMSCONV does not permit a version to be specified for the target element.

– Structured duplication: LMSCONV does not permit the structured duplication of delta trees (DUP...,STRUC=YES).

6.7.2 LMSCONV compared with LMR

LMR can only handle modules in OMLs (object module libraries). This corresponds to type R in PLAM libraries. The following table compares the LMSCONV and LMR statements. With LMSCONV statements that allow several types to be processed, only type R is mentioned.

LMSCONV statement	LMR statement	Function
ADDR	ADD, COPYALL	Add modules
DELR	DELETE	Delete modules
DUPR	ADD, COPYALL	Duplicate modules
END	END	Terminate program
LIB	MODLIB, SOURCE	Library assignment
PRT(LST)+LSTR	LIST	List modules via SYSLST
[PRT(OUT)+]LSTR	DISP	List modules via SYSOUT
NAMR	REVISE	Rename modules
PAR	PARAMS	Define processing operands
PRT	-	Define output medium
SELR	-	Select modules
[PRT(OUT)+]TOCR	DISP	Output table of contents to SYSOUT
PRT(LST)+TOCR	LIST	Output table of contents to SYSLST
RST	-	Restart following error
SYS	-	Issue system commands
UPDR	INCLUDE	Update modules
*INS	RENAME	- Insert INCLUDE
*NAM	REP	- Change internal name
*REP	TRAITS	- Insert REP record
*SET	-	- Modify attributes
*BASE	-	- Define base address
*CON	-	- Define cross-check number
*COR	-	- Correct text records
*DEL	-	- Delete record types
*END	-	- Terminate update statement
*ID	-	- Define identification
*INV	-	- Convert updates
*REM		- Undo updates

The following table compares the LMR and LMSCONV processing operands:

LMSCONV statement	LMR	Processing operand
CSECT	-	Define CSECT
DECOMPRESSED	-	Control compression
DESTROY	-	Control data destruction
FORMAT	-	Define record format
INFO	-	Define output scope on listing
KEY	-	Control transfer of file attributes and ISAM keys
LOG	CLIST	Define statement logging
OVERWRITE	OVERWRITE	Define overwrite
PATH	-	Define path name
PHASE	-	Define phase format
REFERENCE	-	Define reference condition
SLICE	-	Define segment
SORT	-	Sort contents listing
TERMINATE	-	Define abort action
STRIP	ISD/NOISD	Keep/delete ISD records
STRIP	INCL/NDINCL	Keep/delete INCLUDE records
STRIP	REP/NOREP	Keep/delete REP records
-	CLASS1/CLASS2	Define program class indicator
-	GENDATE	Define creation date

Certain LMR attributes cannot be reproduced in LMSCONV:

– task switches
– STXIT behavior
– log formats
– directory 2 information:
  Only part of the corresponding information is stored in PLAM libraries by the programs generating it, i.e. only what is needed for linking/loading is stored.

6.7.3 LMSCONV compared with MLU

MLU only processes macro libraries. Macros are stored by LMS/LMSCONV as type M elements in PLAM libraries.

The following table compares the LMSCONV statements that allow type M, and also the typeless statements, with the MLU statements.

LMSCONV statement	MLU statement	Function
ADDM	ADD	Add macros
-	COMP	Compress macros
-	DCOMP	Decompress macros
DELM	DEL	Delete macros
DUPM	COPY	Copy macros
END	END	Terminate program
LIB	MACLIB, COPYLIB	Assign library
[PRT(OUT)+]LSTM	LIST	List macros via SYSOUT
PRT(LST)+LSTM	PRINT	List macros via SYSLST
NAMM	-	Rename macros
PAR	-	Define processing operands
PRT	-	Define output medium
SELM	-	Select macros into file
[PRT(OUT)+]TOCM	LSDIR	Output table of contents to SYSOUT
PRT(LST)+TOCM	PRDIR	Output table of contents to SYSLST
-	PUNCH	Output macros to SYSOPT
RST	-	Restart following error
SYS	-	Issue system command

There are further differences:

– LMSCONV, unlike MLU, has processing operands (see section 6.5.5, "Processing operands").
– The storage format can only be selected for OSM libraries in LMSCONV. The DECOMPRESSED processing operand has been implemented for this purpose. No statement exists, however, to compress complete OSM libraries.

The MLU STXIT behavior and log formats have not been emulated.

6.8 Conversion tools

6.8.1 Converting from LMR to LMSCONV

LMR format

Libraries created by LMR can be processed by LMSCONV. As elements processed by LMR are not given a version number, LMSCONV assumes a version number of 0 (zero) for these elements. Non-empty object module libraries created and processed by LMSCONV can be processed by LMR. LMR cannot process program libraries.

Converting from LMR to LMS

General:

1. OVERWRITE default value.
   In LMR, the default value is OVERWRITE=YES, whereas in LMSCONV it is OVERWRITE=NO.
2. Completion log.
   In LMSCONV you must set PAR LOG=MAX to obtain a completion log.

Mapping of LMR functions

a) Include all modules from *EAM, overwriting elements with the same name

/START-PROGRAM FROM-FILE=$LMR MODLIB <library> COPYALL SOURCE=* END	/START-PROGRAM FROM-FILE=$LMSCONV $PAR OVERWRITE=YES $LIB <library>,BOTH $ADDR *OMF $END

b) Include one module from *EAM, overwriting elements with the same name

/START-PROGRAM FROM-FILE=$LMR MODLIB <library> ADD OBJMOD=(name),SOURCE=* END	/START-PROGRAM FROM-FILE=$LMSCONV $PAR OVERWRITE=YES $LIB <library>,BOTH $ADDR *OMF(name) $END

c) Copy one module from another library, overwriting elements with the same name

/START-PROGRAM FROM-FILE=$LMR	/START-PROGRAM FROM-FILE=$LMSCONV
	$PAR OVERWRITE=YES
	$LIB <in.lib>
MODLIB <library>	$LIB <library>,OUT
ADD OBJMOD=(name),SOURCE=	$DUPR name
(LIB,<in.lib>)	
END	$END

d) Copy all modules from another library, overwriting elements with the same name

/START-PROGRAM FROM-FILE=$LMR	/START-PROGRAM FROM-FILE=$LMSCONV
	$PAR OVERWRITE=YES
	$SYS ADD-FILE-LINK LINK-NAME=LIBzzz,
	FILE-NAME=in.lib
MODLIB <library>	$LIB <library>,BOTH
COPYALL SOURCE=(LIB,<lib>)	$DUPR *(<zzz>)
END	$END

e) List modules in a library

/START-PROGRAM FROM-FILE=$LMR	/START-PROGRAM FROM-FILE=$LMSCONV
MODLIB <library>	$LIB <library>
DISP D1	$TOCR *[/*]
DISP D2,ALL	$PAR INFO=REF!$LSTR *[/*]
DISP OBJMOD=(name,ALL)	$PAR INFO=ALL!$LSTR name[/*]
END	$END

f) Rename a module in the table of contents

/START-PROGRAM FROM-FILE=$LMR	/START-PROGRAM FROM-FILE=$LMSCONV
MODLIB <library>	$LIB <library>,BOTH
REVISE OBJMOD=(old,new)	$NAMR old>new
END	$END

g) Delete a module from the table of contents

/START-PROGRAM FROM-FILE=$LMR	/START-PROGRAM FROM-FILE=$LMSCONV
MODLIB <library>	$LIB <library>,BOTH
DELETE (name)	$DELR name
END	$END

h) Insert REP in a module

/START-PROGRAM FROM-FILE=$LMR MODLIB <library> REP ...,OBJMOD=name END	/START-PROGRAM FROM-FILE=$LMSCONV $LIB <library>,BOTH $UPDR name *REP ..... *END $END

6.8.2 Converting from MLU to LMSCONV

MLU format

Libraries created by MLU can be processed by LMSCONV. As elements processed by MLU are not given a version number, LMSCONV assumes a version number of 0 (zero) for these elements. Macro and source program libraries created by LMSCONV are in MLU format and can be processed by MLU. MLU cannot process program libraries.

Mapping of MLU functions

a) Copy a macro from another library, overwriting any element with the same name

/START-PROGRAM FROM-FILE=$MLU COPYLIB=<in.lib> MACLIB=<library> COPY macroname END	/START-PROGRAM FROM-FILE=$LMSCONV $PAR OVERWRITE=YES $SYS ADD-FILE-LINK LINK-NAME=LIBzzz FILE-NAME=in.lib $LIB <library>,BOTH $DUPM macroname(zzz) $END

b) Delete macros from a library

/START-PROGRAM FROM-FILE=$MLU MACLIB=<library> DEL macroname END	/START-PROGRAM FROM-FILE=$LMSCONV $LIB <library> $DELM macroname $END

c) Output table of contents

/START-PROGRAM FROM-FILE=$MLU MACLIB=<library> LSDIR END	/START-PROGRAM FROM-FILE=$LMSCONV $LIB <library> $TOCM */* $END

6.8.3 Converting from COBLUR to LMSCONV

Libraries created by COBLUR are only read by LMSCONV, which cannot make changes to these libraries. As the elements in a COBLUR library have neither version numbers or dates, zero is assumed for both. LMSCONV ignores the sections into which the library is divided, so it is not possible to tell from a contents listing which elements are held in which sections. If different sections contain elements with the same name, these names will be listed several times in a contents listing. If the LST statement specifies a name that exists in several sections, the first element that is found is listed.

6.8.4 Converting from LMSCONV to LMS

All LMSCONV statements and processing operands are understood by LMS (on LMS, see the "LMS (BS2000)" manual [11]). Thus the user can convert from LMSCONV to LMS without any problems.

All existing LMSCONV procedures also run under LMS and produce exactly the same results.

7 PAMCONV
Utility routine for converting file formats

Version: **PAMCONV V11.0A**

The PAMCONV routine is used for converting files from K format to NK format or vice versa.

In **K format**, the DMS management information is stored in a PAM key which is prefixed to the data block. This file format is referred to as key format, or K format for short.

In **NK format**, which does not use the PAM key, the DMS management information is either integrated in the data blocks or it is left out. This file format is known as nonkey format, or NK format for short.

Prior to BS2000 V10.0 the only nonkey format was NK2 format. Files with this format were generally referred to as NK files. As of BS2000/OSD-BC V1.0 there is now another NK file format, NK4. The term NK file now serves as a generic term for NK2 and NK4 files.
These file formats (K, NK2 and NK4) were created so as to be enable the data management system to make the best possible use of the existing disk formats. As of BS2000/OSD-BC V1.0, NK4 disks are also supported in addition to K and NK2 disks (BS2000 ≤ V10.0). The minimum transfer unit between the disk and main memory for this disk format is 4 Kbytes. The size of the smallest file (minimum allocation unit = min. AU) is either 8 Kbytes or 64 Kbytes, depending on the formatting by system administration.
Files on NK4 disks must always be in NK4 format. PAMCONV V11.0 offers the possibility of converting K or NK2 files to NK4 format.

A disk format is defined by the critera "with/without PAM key", "minimum allocation unit (min. AU)" and "minimum transfer unit (min. TU)".
The following disk formats are supported as of BS2000/OSD-BC V1.0:

Disk format	PAM key		min.AU			min.TU	
	with	w/o	6KB	8KB	64KB	2KB	4KB
K disk	x		x			x	
NK2 disk		x	x			x	
NK2 disk		x		x		x	
NK4 disk		x		x			x
NK2 disk		x			x	x	
NK4 disk		x			x		x

The disk format within any one pubset is homogeneous.

For private disks the only formats supported are the K and NK2 disks with a minimum allocation unit of 6 Kbytes.

The diagram below shows which file formats can be stored on the supported disk formats without the file formats first having to be converted using PAMCONV.

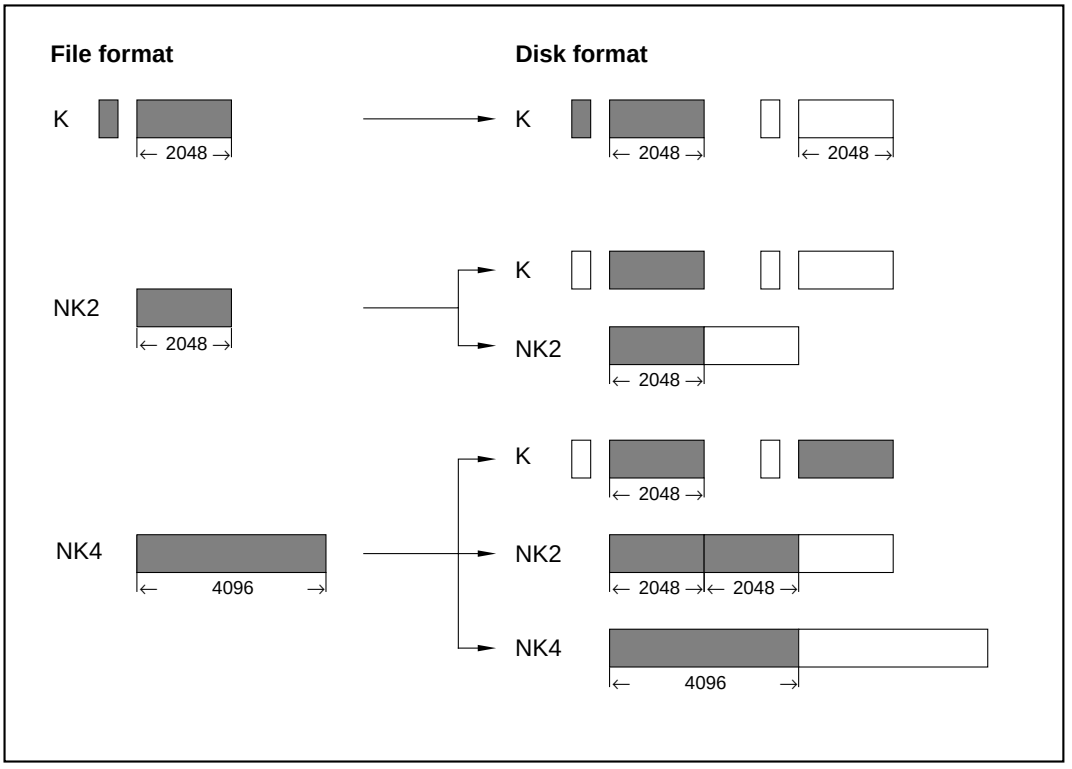
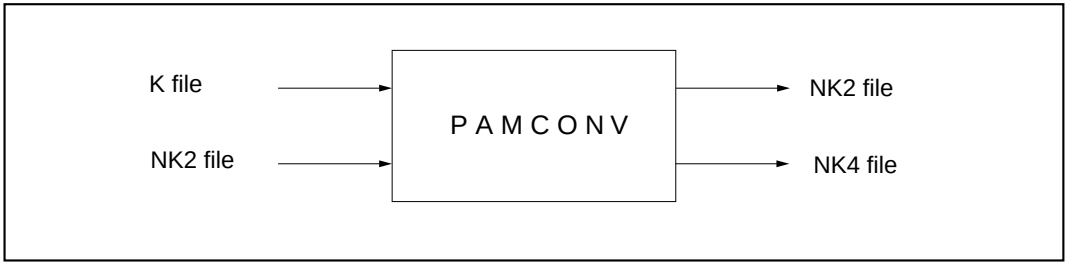


Figure 10: File formats that can be mapped to certain disk formats without conversion

The primary purpose of PAMCONV is to convert K files into NK files so that the latter can be stored in NK pubsets.



The conversion options provided by PAMCONV V11.0 for the individual file structures are listed in the table below.

Conversion options with PAMCONV V11.0

File structure	File format1		File format2
ISAM	K-ISAM	⟷	NK2-ISAM
	K-ISAM	⟷	NK4-ISAM
	NK2-ISAM	⟷	NK4-ISAM
SAM	K-SAM	⟷	NK2-SAM
	K-SAM	⟷	NK4-SAM
	NK2-SAM	⟷	NK4-SAM
PAM	K-PAM	⟷	NK-PAM
	K-PAM	⟷	NK4-PAM
	NK2-PAM	⟷	NK4-PAM
Load module	K module	⟷	NK2 module
	K module	⟷	NK4 module
	NK2 module	⟷	NK4 module

Calling PAMCONV

The routine can be started in two ways:

```
/START-PAMCONV
```

START-PAMCONV
CPU-LIMIT = **_*JOB-REST_** / <integer 1..32767>
,**MONJV** = **_*NONE_** / <full-filename without-gen-vers>
,**PROG**RAM-**MODE** = **_24_** / ***ANY**

The CPU-LIMIT, MONJV and PROGRAM-MODE operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in "Commands, Volume 3" [3].

```
/START-PROGRAM FROM-FILE=PAMCONV
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are also available for calling the routine. For descriptions of these operands, see the START-PROGRAM command in "Commands, Volume 3" [3].

Command statements of the routine can be entered once the program has run.

The following messages are output:

```
%  BLS0500 PROGRAM 'PAMCONV', VERSION '11.0A' OF '1993-03-11' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993. ALL
RIGHTS RESERVED
%  PEA7000 14:48:34/0.0859 PAMCONV-VERSION V11.0A64 IN BS2000 V11.2  STARTED
%  PEA7001 ENTER PAMCONV STATEMENTS
//
```

7.1 Functionality of PAMCONV

The PAMCONV utility routine provides the user with two basic functions for adapting files to the available disk format.

- **File format conversion**

  One of the functions of the PAMCONV routine is to convert files from K format to NK format (and vice versa). File format conversion is performed using statements, specifically the CONVERT-FILE statement.

- **Reblocking**

  The introduction of NK4 disks in BS2000/OSD-BC V1.0 means that it is possible to convert the blocking factor of a file from an odd number to an even number. Only files with an even-numbered blocking factor can be stored on an NK4 disk.

  PAMCONV provides the "reblocking" function for this purpose.

Functionality of PAMCONV V11.0 in the different BS2000 versions

The PAMCONV V11.0 utility routine can run under BS2000 V9.5 or higher, although its functionality differs according to the BS2000 version used. The following tables provide an overview of the conversion options for ISAM, SAM, PAM and load module files in BS2000 V9.5, V10.0 and BS2000/OSD-BC V1.0/V2.0.

In addition, the basic directions of conversion are shown in diagram form for each file structure, with the block structure of the file before and after conversion.

ISAM files

An ISAM file that is to be converted with PAMCONV may have one of the following three block structures.

- **PAMKEY**: the file is a K-ISAM file.
- **DATA2K**: the file is an NK2-ISAM file; the blocking factor n may be odd or even.
- **DATA4K**: the file is an NK4-ISAM file; the blocking factor n is even.

The table below summarizes all conversion options for an ISAM file:

Conversion options for an ISAM file

Conversion options			V9.5	V10.0	OSD-BC V1.0	OSD-BC V2.0
PAMKEY	→	DATA2K	0	X	X	X
PAMKEY	→	DATA4K			X	X
DATA2K	→	PAMKEY	X	X	X	X
DATA2K	→	DATA4K			X	X
DATA4K	→	PAMKEY	X	X	X	X
DATA2K	↔	DATA2K	0	X	X	X
DATA4K	↔	DATA4K			X	X
DATA4K	→	DATA2K	0	X	X	X

X Conversion is supported.

≙

0 If the NK-ISAM access method is available in the system, the file format can be converted to NK format.

≙

In BS2000 V9.5, NK-ISAM is optional. In order to convert K-ISAM files into NK-ISAM files in BS2000 V9.5, therefore, NK-ISAM must be loaded.

MLU macro libraries are ISAM files and must be converted as such.

There are three basic directions of conversion

PAMKEY	→	DATA2K
PAMKEY	→	DATA4K
DATA2K	→	DATA4K

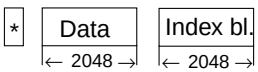
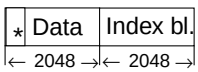
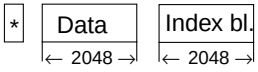
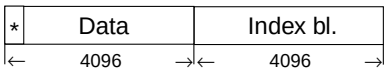
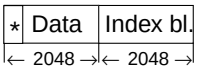
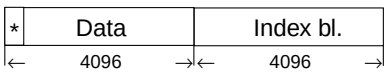
The block structure of an ISAM file before and after file format conversion in these three directions is shown below.

Key

* →16 bytes block management information

BLKSIZE → logical block length

Indexblock → index block

ISAM file before conversion		ISAM file after conversion
PAMKEY (K-ISAM file) 	→	**DATA2K** (BLKSIZE=(STD,1)) (NK2-ISAM file) 
PAMKEY (K-ISAM file) 	→	**DATA4K** (BLKSIZE=(STD,2)) (NK4-ISAM file) 
DATA2K (BLKSIZE=(STD,1)) (NK2-ISAM file) 	→	**DATA4K** (BLKSIZE=(STD,2)) (NK4-ISAM file) 

SAM files

A SAM file that is to be converted with PAMCONV may have either of the following two block structures:

- **PAMKEY**: the file is a K-SAM file.
- **DATA**: the file is an NK2-SAM file if the blocking factor n is odd, or an NK4-SAM file if n is even.

The table below summarizes all conversion options for a SAM file:

Conversion options for a SAM file

Conversion options			V9.5	V10.0	OSD-BC V1.0	OSD-BC V2.0
PAMKEY	→	DATA	X	X	X	X
DATA	→	PAMKEY	X	X	X	X
DATA	↔	DATA	X	X	X	X

X Conversion is supported.

≙

If SAM files are converted from K format to NK format in BS2000 V9.5, the target files have an internal NK format but the source file remains in K format.

The diagram below shows the following direction of conversion

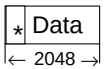
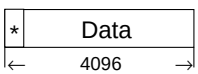
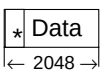
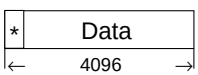
PAMKEY	→	DATA

This illustrates the block structure of a SAM file before and after file format conversion.

Key:

* → 16 byte block management information

BLKSIZE → logical block length; in the case of NK-SAM files the 16 bytes are deducted only once per logical block length.

SAM file before conversion		SAM file after conversion
PAMKEY (K-SAM file) 	→	**DATA** (BLKSIZE=(STD,1)) (NK2-SAM file) 
PAMKEY (K-SAM file) 	→	**DATA** (BLKSIZE=(STD,2)) (NK4-SAM file) 
DATA (BLKSIZE=(STD,1)) (NK2-SAM file) 	→	**DATA** (BLKSIZE=(STD,2)) (NK4-SAM file) 

PAM files

A PAM file that is to be converted with PAMCONV may have one of the following three block structures:

- **PAMKEY**: the file is a K-PAM file.
- **DATA**: the file is an NK2-PAM file if the blocking factor n is odd, or an NK4-PAM file if n is even.
- **NO**: the file is an NK2-PAM file if the blocking factor n is odd, or an NK4-PAM file if n is even. No block management information is stored.

The table below summarizes all conversion options for a PAM file:

Conversion options for a PAM file

Conversion options			V9.5	V10.0	OSD-BC V1.0	OSD-BC V2.0
PAMKEY	→	NO	X	X	X	X
NO	→	PAMKEY	X	X	X	X
NO	↔	NO		X	X	X
DATA	↔	DATA		X	X	X

X Conversion is supported.

≙

If PAM files are converted from K format to NK format in BS2000 V9.5, the target files have an internal NK format but the source file remains in K format.

PAM-DATA files cannot be converted into K format. If nonkey to key conversion is selected nevertheless, processing in PAMCONV V11.0 is rejected with message `PEA2212`.

PLAM libraries are PAM files that do not use the PAM key and must therefore be converted as such.

The diagram below shows the following three conversion directions:

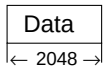
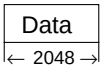
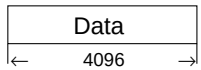
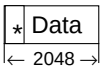
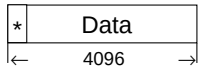
PAMKEY	→	NO
NO	→	NO
DATA	→	DATA

This illustrates the block structure of a PAM file before and after file format conversion.

Key:

* → 12 bytes block management information

BLKSIZE → logical block length

PAM file before conversion		PAM file after conversion
PAMKEY (K-PAM file) 	→	**NO** (BLKSIZE=(STD,1)) (NK2-PAM-NO file) 
NO (BLKSIZE=(STD,1)) (NK2-PAM-NO file 	→	**NO** (BLKSIZE=(STD,2)) (NK4-PAM-NO file) 
DATA (BLKSIZE=(STD,1)) (NK2-PAM-DATA file) 	→	**DATA** (BLKSIZE=(STD,2)) (NK4-PAM-DATA file 

Load module files

A load module is a specific type of PAM file. In the PAM file name (K-PAM or NK-PAM file), 'PAM' is replaced by 'load module'. Reference is therefore made to K, NK2 and NK4 load module files.

NK2 load module files have the file format with the block structure NO, i.e. no block control information is stored. The logical block length is (STD,1), i.e. 2048 bytes.
NK4 load module files have a logical block length of (STD,2), i.e. 4096 bytes.
Any other block length specifications are rejected with an error message.

Conversion options for load module files

Conversion options			V9.5	V10.0	OSD-BC V1.0	OSD-BC V2.0
PAMKEY	◄——►	NO	X	X	X	X
NO	◄——►	NO	X	X	X	X

7.2 File format conversion

7.2.1 Types of conversion

Standard conversion

It is assumed that the source and target files for a conversion are stored on public volumes (magnetic disk storage).

This might be called the standard case. Enough space for both files must be available. If this is not the case, conversion must be effected via an intermediate file.

Conversion via an intermediate file

- General

  During file conversion, the target file requires about the same amount of disk space as the source file. A "self-contained" conversion without the need for additional space is not supported by PAMCONV. This means that enough disk space must be available for the target file. If this is not the case, conversion can be effected using an intermediate file on magnetic tape or private disk.

  After successful conversion from the source file to the intermediate file, the source file is deleted to obtain space for the target file.

  When conversion takes place via an intermediate file, the FILE-DISPOSAL operand of the CONVERT-FILE statement (which specifies how the generated file is to be handled after conversion) is ignored.

  For this type of conversion, file convertibility is checked before an intermediate file is generated. This avoids a situation where the intermediate file would not be convertible into a target file.

- Two-step conversion using an intermediate file

  This type of conversion is performed explicitly by means of two CONVERT-FILE statements.
  The two statements may be issued in the same program run or in separate program runs.

  – Step 1
    Conversion of the source (disk) file on magnetic tape or private disk.

  – Step 2
    Conversion of the intermediate file from magnetic tape or private disk into the target file.

- One-step conversion using an intermediate file

  Two-step conversion as described above is combined into one operation here, which means that only one CONVERT-FILE statement is needed. This is achieved by entering DISK or TAPE for the DEVICE-FOR-TEMPFILE operand.

  The intermediate file is assigned a name with the following format:

  SYSTMP.tsn.PAMCONV.ss.cpusec

  This intermediate file is erased after the two (internal) conversion steps have been successfully completed; otherwise further processing of the file is possible via this name.

  Note

  Internally, conversion is implemented in two steps as mentioned above. However, the user is not requested to make a new input until both conversion steps have been concluded.

Specification of the conversion options

The conversion variants are selected on the basis of the source and target file specifications and the entry in the DEVICE-FOR-TEMPFILE operand of the CONVERT-FILE statement:

- DEVICE-FOR-TEMPFILE = NONE

  No intermediate file is stored on a private volume, unless this is specified via an ADD-FILE-LINK command.

  The following variants are possible:

  – The source file is specified as a disk file *and* there is no catalog entry or ADD-FILE-LINK command for the target file, or the target file is specified as a disk file.

    ⇨ Conversion from source file on magnetic disk to target file on magnetic disk (standard case).

  – The source file is specified as a disk file and the target file is specified as a tape file.

    ⇨ Conversion from source file on magnetic disk to intermediate file on magnetic tape (two-step conversion via intermediate file on tape: step 1).

– The source file is specified as a tape file *and* there is no catalog entry or ADD-FILE-LINK command for the target file, or the target file is specified as a disk file.

⇨ Conversion from source file on magnetic tape (must be an intermediate file generated by PAMCONV) to target file on magnetic disk (two-step conversion via intermediate file on tape: step 2).

Specifying both source file and target file as magnetic tape files is not permitted.

Note

This implies that a file on magnetic tape must always be a PAMCONV intermediate file, otherwise conversion is rejected.

- DEVICE-FOR-TEMPFILE = TAPE

An intermediate file on magnetic tape is generated in each case. The source and/or target file must not simultaneously be specified as a tape file via an ADD-FILE-LINK command.

The following options exist:

– The source file is specified as a disk file *and* there is no catalog entry or ADD-FILE-LINK command for the target file, or

– the target file is specified as a disk file.

⇨ Conversion from source file on magnetic disk to target file on magnetic disk (one-step conversion via intermediate file on tape).

- DEVICE-FOR-TEMPFILE = DISK

An intermediate file on private disk is generated in each case. The source and/or target file must not simultaneously be specified as a tape file via an ADD-FILE-LINK command. If the source and/or target file is defined by means of an ADD-FILE-LINK command (for private disk), the volume specified in this command must not be identical to the private disk volume identified in the DEVICE-FOR-TEMPFILE operand.

The following options exist:

– The source file is specified as a disk file *and* there is no catalog entry or ADD-FILE-LINK command for the target file, or

– the target file is specified as a disk file.

⇨ Conversion from source file on magnetic disk to target file on magnetic disk (one-step conversion via intermediate file on disk).

Format of the intermediate file on magnetic tape

The source (disk) files may be SAM, ISAM or PAM files. The ISAM access method is not defined for tapes. A standard format is therefore used for the intermediate file on tape. This is a SAM file containing the data records of the source file (ISAM: sorted by keys in ascending order). For general PAM files and load modules, a record consists of an 8-byte field with the user part of the PAM key and a 2048-byte field with the PAM block.

Additional file attributes are stored in a separate user header label (UHL).

Such an intermediate file is merely intended as a temporary file for conversion purposes. Magnetic tapes with standard labels must be used.

- Two-step conversion via intermediate file on tape

  If several files are to be converted at the same time, an ADD-FILE-LINK command with SUPPORT = TAPE(FILE-SEQUENCE =...) must be issued; otherwise the intermediate file on tape will be overwritten.

- File attributes

  An intermediate file on magnetic tape has the following attributes:

```
FCBTYPE = SAM
BLKSIZE = (STD,16)
RECFORM = VARIABLE
LABEL   = STD
BLKCTRL = PAMKEY
```

  Otherwise the default values from the FCB macro apply.

  As can be seen from the following diagram, an intermediate file on tape is always a K-SAM file, regardless of the key format or FCB type of the source and target files.

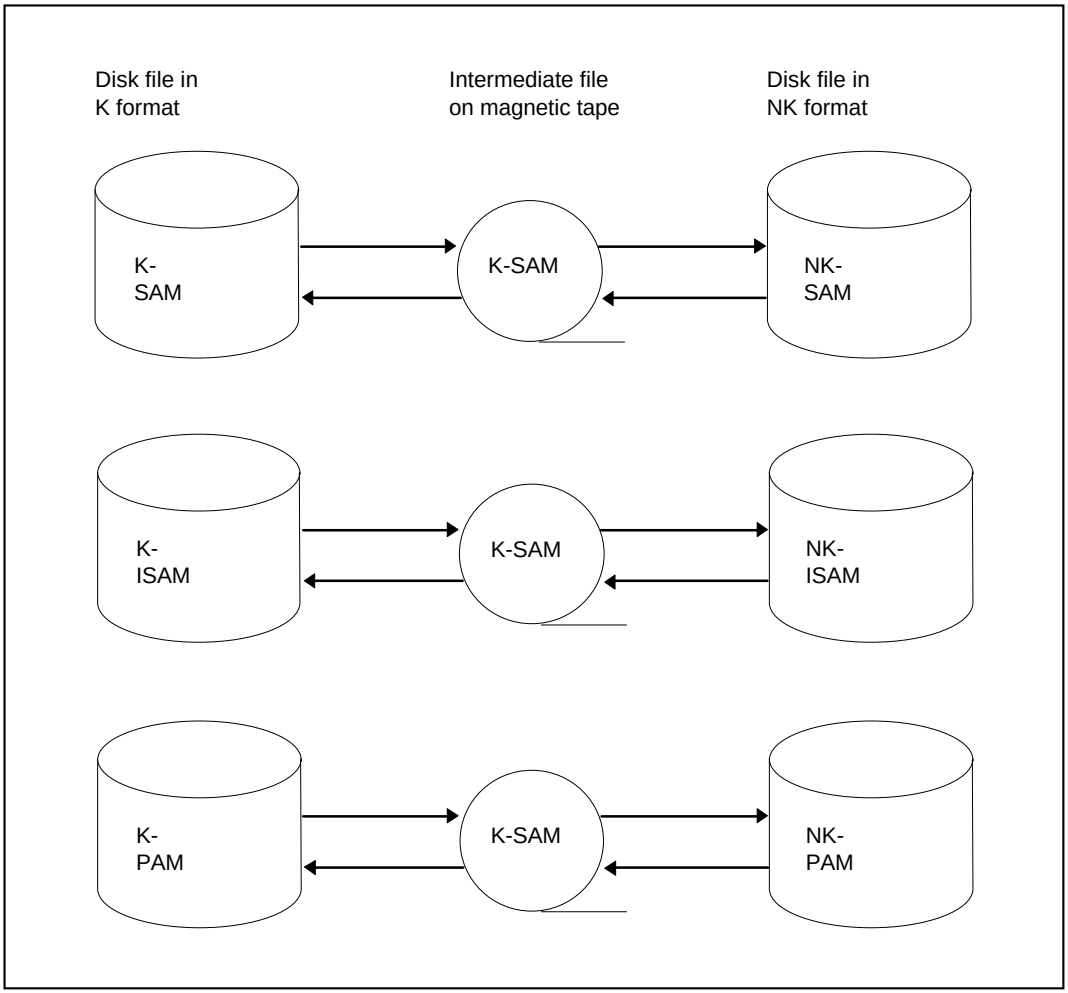


Figure 11: Intermediate file on magnetic tape

- User labels

  For the registration of the source file attributes, the user header label UHL1 with the following format is used.

Position	Contents	Meaning
1..4	UHL1	user header label identification
5..12	PAMELA-I	ID for ISAM intermediate file
5..12	PAMELA-S	ID for SAM intermediate file
5..12	PAMELA-P	ID for PAM intermediate file
13..66	CL54	Name of source file
67	AL1(b)	BLKSIZE=(STD,b)
68	X‘02‘	RECFORM=VARIABLE
68	X‘04‘	RECFORM=FIXED
69..70	AL2(r)	RECSIZE=r
71..72	AL2(p)	KEYPOS=p
73	AL1(k)	KEYLEN=k
74	X‘00‘	VALPROP=MIN
74	X‘01‘	VALPROP=MAX
75	AL1(f1)	LOGLEN=f1
76	AL1(f2)	VALLEN=f2
77	X‘80‘	DUPEKY=YES
77	X‘00‘	DUPEKY=NO
78..80	XL3‘00‘	not used

Format of the intermediate file on private disk

In contrast to an intermediate file on tape, an intermediate file on private disk has no special format. It is always a copy of the source/target file, depending on the conversion direction. The intermediate file is always generated as an NK file, which makes it independent of the private disk’s key mode.

Note

In the case of SAM files, the RECSIZE value of the copy (intermediate file) may differ from that of the NK source file. If the RECSIZE value of the NK source file is zero, the maximum possible value is used for the intermediate file.

The following diagram shows in which format the intermediate file is created on private disk and from which file the copy is made (depending on the conversion direction).

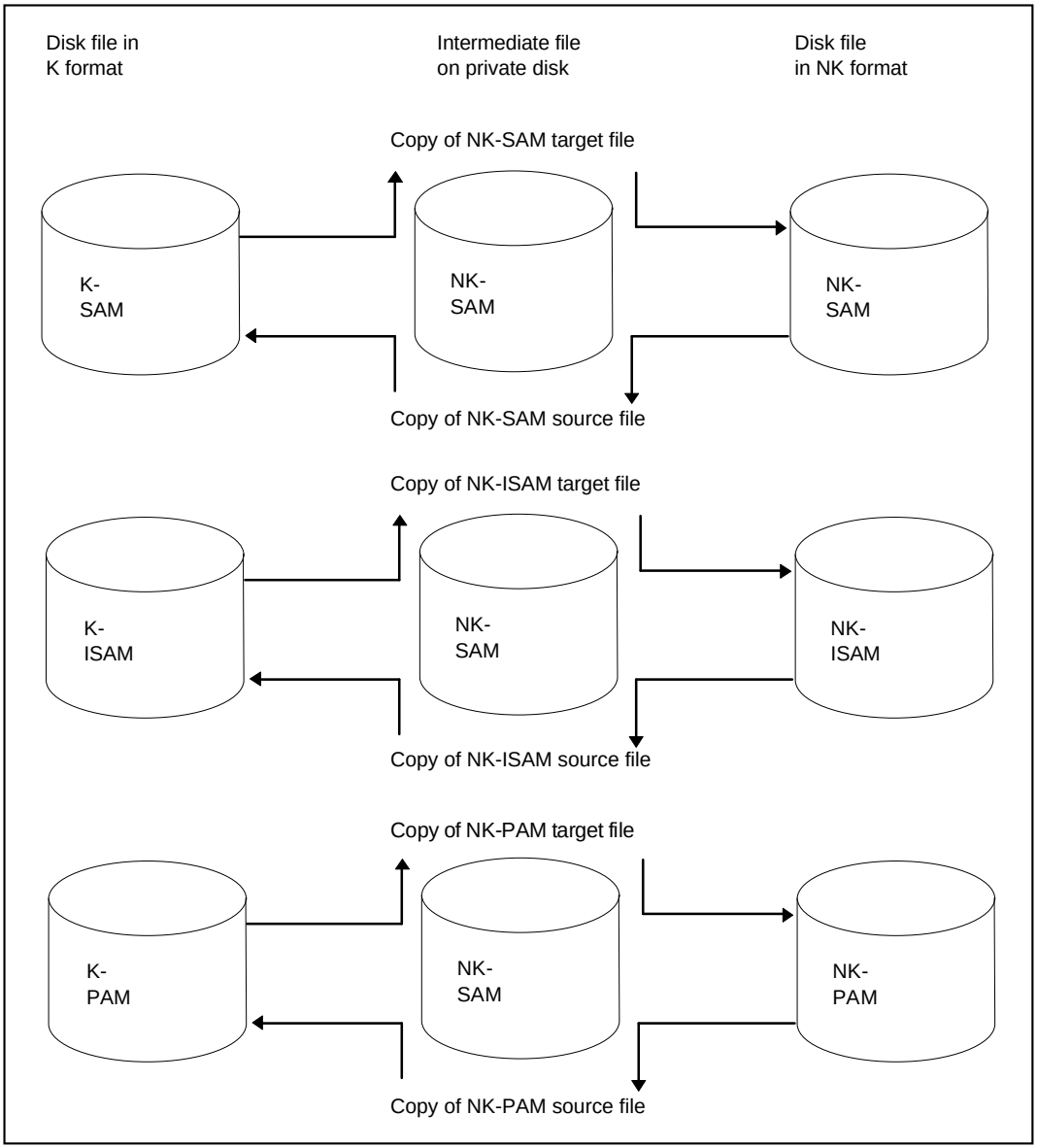


Figure 12: Intermediate file on private disk

7.2.2 System environment requirements

Tape peripherals

Magnetic tapes which are to store intermediate files must be equipped with standard labels.

Disk peripherals

If a system contains pure NK disk peripherals (without key simulation), creation of K disk files is not possible. As a consequence, "nonkey to key" conversion is impossible in such systems.

If, generally speaking, both the source and the target file are to be situated within one pubset, care must be taken to provide enough space so that both files can be accommodated during conversion. (The target file may need more space than the source file.) If this is not possible, conversion should be effected via an intermediate file.

Operating system versions

The NK-ISAM access method is used for the "to nonkey" conversion of ISAM files. Accordingly, NK-ISAM must be present in the system (optional in V9.5, standard as of V10.0).

Generation of ISAM files with BLKCTRL = DATA4K is only possible as of BS2000/OSD-BC V1.0 and PAMCONV V11.0A.

SDF

PAMCONV statements have an SDF interface.

The SDF subsystem is available in the system as of BS2000 V10.0 and therefore does not have to be loaded separately.
In BS2000 V9.5, SDF is loaded automatically from the $TSOS.SDF.OML.P1 module library. This library must therefore be available in the system.

ACS

The ACS (Alias Catalog Service) subsystem is available as of BS2000/OSD-BC V1.0. Alias names may be used when specifying the source and target files for conversion. These names must be specified in precisely the form in which they are entered in the alias catalog, otherwise it is impossible to establish the connection to the correct file name.

Diagnostic documents

If problems occur in a PAMCONV run (unexpected messages, dump, ...), the following documents are required for the diagnosis:

- SYSLST log
  The log is designed to show the factors possibly affecting the PAMCONV run (e.g. inputs made immediately before the error occurred).
- Input file in which the error occurred. The development engineers may then be able to find the cause of a problem which may not lead to an abort until considerably later.

Syntax files

A syntax file with a description of the PAMCONV syntax must be active. For BS2000/OSD-BC versions earlier than V1.0, two SDF syntax files are supplied, which system administration has to integrate into the existing syntax files. As of BS2000/OSD-BC V1.0, there is only one system syntax file, which contains the full function range of PAMCONV.

- For BS2000/OSD-BC versions earlier than V1.0:

  1. PAMCONV system syntax file

     Contains the restricted function range of PAMCONV, which is available to all users. This file must be merged with the system syntax file.

  2. PAMCONV group syntax file for TSOS

     Contains the full function range of PAMCONV, which is only available to system administration under the TSOS ID. This file must be merged with the group syntax file for TSOS.

- For BS2000/OSD-BC V1.0 and later:

  PAMCONV system syntax file

  Contains the full function range of PAMCONV. This file must be merged with the system syntax file.

7.2.3 Specifying source and target files

The source file and the corresponding target file can be specified in one of three ways:

Specifying fully qualified file names

Source and target files can be selected by specifying fully qualified file names.
A single file generation can also be specified in this way.

In this case, the file attributes of the source file are transferred internally from the catalog entry.

The file attributes of the source file likewise apply to the target file.

The following file attributes are registered (see the ADD-FILE-LINK command):

– access method
– record format
– record length
– data block length
– key length in ISAM files
– key position in ISAM files
– multiple keys in ISAM files
– length of logical markers in ISAM files
– length of value markers in ISAM files
– character set

Specifying selection criteria

The source file is specified in the form of a partial qualification with or without additional selection criteria operand (SELECT=BY-ATTRIBUTES(...) of the CONVERT-FILE statement with CREATION-DATE, LAST-ACCESS-DATE, SIZE, FILE-STRUCTURE, BLKSIZE, BLKCTRL).

This information is used to determine all files to be converted (see the SHOW-FILE-ATTRIBUTES command in the manual “Commands, Volume 3" [3]).

The names of the target files may include wildcards (see the SHOW-FILE-ATTRIBUTES command).

The remaining attributes of the target file are the same as for the source file.

Note

If LMR module libraries were included in the specification of selection criteria for PAM files, the module libraries could be destroyed.

To avoid this, a check is performed prior to conversion of a PAM file, and if an LMR module library is detected it is not converted.
LMR module libraries can be converted using LMS (see the description of LMSCONV starting on page 141).

PAM files which are load modules are recognized as such and converted to the appropriate load module file format.

Specifying a reference via a link name

The reference is established via the link name to a previous ADD-FILE-LINK command specifying the file.
The link name is specified in the *LINK(LINK-NAME=...) operand of the CONVERT-FILE statement.
This method can also be used to select a single file generation.

If PAMCONV is executed under a BS2000 version ≥ V9.5, the BLKCTRL-INDICATOR operand of the MODIFY-BLKCTRL-INDICATOR statement need not be specified.
If it is specified, it must be correct; in the case of any inconsistencies, conversion is rejected.

The link name is mandatory if the target file is to have specific attributes which cannot be taken from the catalog entry of the source file. These include in particular:

– SPACE definitions
– index/data separation (for K-ISAM files only)
– PAD factor (normally the default value is used)
– RETPD specification
– WROUT=NO (WROUT=YES is standard)
– WRCHK=YES (WRCHK=NO is standard)

7.3 Reblocking

7.3.1 General information

Only files with an even blocking factor can be stored on NK4 disks. For this reason, the "reblocking" function has been incorporated in PAMCONV V11.0. This function allows files with an uneven blocking factor to be stored on NK4 disks. It is implemented in the TO-FILE-BLKSIZE operand of the CONVERT-FILE, MODIFY-CONVERT-FILE-DEFAULTS and SHOW-CONVERT-FILE-DEFAULTS statements.

7.3.2 Explicit reblocking

The user initiates reblocking in the TO-FILE-BLKSIZE operand, which defines the logical block size of the target file. If the block size of the source file is uneven and the user explicitly specifies the block size of the target file, an error message is issued when the file is opened on an NK4 disk.

7.3.3 Implicit reblocking

If TO-FILE-BLKSIZE = STD or TO-FILE-BLKSIZE = NK4 is specified, reblocking is carried out by PAMCONV.

- **Increasing the blocking factor implicitly with TO-FILE-BLKSIZE = STD**

  The blocking factor of the target pubset applies; if necessary, PAMCONV increases the blocking factor implicitly.
  The block size remains unchanged for NK2 and NK2(8K,64K) pubsets. In the case of NK4 pubsets, the blocking factor of the target file is increased if the blocking factor of the source file is uneven.

- **Increasing the blocking factor implicitly with TO-FILE-BLKSIZE = NK4**

  If the blocking factor of the source file is uneven, the blocking factor of the target file is increased, regardless of the target pubset blocking factor.

PAMCONV does not decrease the blocking factor implicitly.

7.3.4 Reblocking PAM-DATA files without changing the file format

Another function of PAMCONV V11.0 is reblocking NK-PAM-DATA files without changing the file format, i.e. both the source and the target file have the attribute BLKCTRL = DATA. Reblocking without conversion is supported by the specification DIRECTION = TO-NONKEY and by the TO-FILE-BLKSIZE operand.

The blocking factor of a PAM-DATA file is **increased** only if the block size of the target file (TO-FILE-BLKSIZE) can be divided by the block size of the source file without leaving a remainder.

Modulo(target-BLKSIZE / source-BLKSIZE) = 0

If the remainder is not zero, reblocking is aborted with an error message.

When increasing the blocking factor of PAM-DATA files, only the block control field of the first block in each logical block is written to the target file. All other control fields in the same logical block are filled to 12 bytes with X'00'.

The blocking factor of PAM-DATA files is **decreased** only if the block size of the source file can be divided by the block size of the target file without leaving a remainder.

Modulo(source-BLKSIZE / target-BLKSIZE) = 0

If the remainder is not zero, reblocking is aborted with an error message.

When the blocking factor is decreased, a check is made as to whether the control fields contain X'00'. If not, reblocking is aborted with an error message.
The blocking factor is decreased only if the blocking factor of the file has already been increased.

If the source PAM-DATA file has the block size BLKSIZE = (STD,1), PAMCONV performs implicit reblocking. Since this changes the file structure, a warning is issued.

Restrictions

– PAMCONV cannot convert NK2-PAM-DATA files with a BLKSIZE greater than (STD,8).

– The "reblocking" function for PAM-DATA files is executed for the TO-NONKEY conversion direction only. If NONKEY-TO-KEY is specified, conversion is aborted with message PEA2212, because PAM-DATA files cannot be converted to KEY.

7.3.5 Problems when decreasing the blocking factor

Decreasing the blocking factor can cause the record length (RECSIZE) of the source file to exceed the block size of the target file.

- Fixed record length (RECFORM = F)

  PAMCONV checks the record length of the source file and compares it with the record length of the target file, calculated from the block size of the target file. If the record length of the source file is greater than that of the target file, processing is aborted with a message.

- Variable or undefined record length (RECFORM = V/U)

  PAMCONV assumes that the record length of the target file is not exceeded and starts processing. If the record length of the target file is nevertheless exceeded, DMS informs PAMCONV and processing is aborted with a message.

Increasing the blocking factor does not cause any problems relating to the record length of the target file.

7.4 Controlling conversion and reblocking

The CONVERT-FILE statement is used to control the conversion and reblocking of files. Other PAMCONV statements are used to set or query the user-defined PAMCONV environment.

The following subsections describe special points relating to the conversion of ISAM and PAM file structures and to the conversion procedure.

7.4.1 Special points relating to conversion

Special points relating to the conversion of ISAM files

The TO-FILE-BLKCTRL operand of the CONVERT-FILE or MODIFY-CONVERT-FILE-DEFAULTS statement can be used to define the file format for ISAM files if the conversion direction TO-NONKEY is selected.
This particular case is subject to certain restrictions with regard to compatibility between the logical block size (TO-FILE-BLKSIZE) and the file format (TO-FILE-BLKCTRL) of an ISAM target file.

The assignment TO-FILE-BLKCTRL = STD means that the block control information is set according to the target pubset. With NK2 and NK2(8K,64K) pubsets, the file format is DATA2K; with NK4 pubsets, it is DATA4K.
If TO-FILE-BLKCTRL = NK4 is specified, the block control information is assigned the value DATA4K.

TO-FILE-BLKSIZE = STD or NK4 or <integer 1..16> defines the logical block size of the target file.
If TO-FILE-BLKSIZE = STD is specified, the logical block size is set according to the target pubset.
If TO-FILE-BLKSIZE = NK4 is specified, the logical block size is always even.
If TO-FILE-BLKSIZE = <integer 1..16> is specified, the target file is generated with a logical block size equal to the value specified here.

The following table shows the compatibility between the logical block size TO-FILE-BLKSIZE and the file format TO-FILE-BLKCTRL of an ISAM target file.

		TO-FILE-BLKSIZE								
		STD	NK4	1	2	3	4	5	6	7
TO-FILE-BLCTRL	STD	X	X	X	X	X	X	X	X	X
	NK4		X		X		X		X	

		TO-FILE-BLKSIZE								
		8	9	10	11	12	13	14	15	16
TO-FILE-BLCTRL	STD	X	X	X	X	X	X	X	X	X
	NK4	X		X		X		X		X

Key

X means that both specifications are supported; otherwise, an error message is issued.

Special points relating to the conversion of PAM files (not load modules)

In order to ascertain whether a PAM file with a PAM key actually uses this key, the user part of the PAM key is checked.

If the user part contains 8 X’00’ bytes in each PAM block, the file is considered to be convertible.

Conversion to NK format consists in dropping the PAM key information (which is not in use); the BLKCTRL indicator field (see the section on "Defining block control information during conversion") is assigned the value NO.

If the user part does not contain X’00’, the file is assumed to use the PAM key and is therefore classified as inconvertible, i.e. conversion is aborted.

Exception:
For PAM files whose PAM key user part contains X’01’ or X’80’ in byte 1, but otherwise X’00’, the system assumes that the PAM key is not used and that byte 1 contains a deviating value due to a DMS error. Such files are therefore classified as convertible to nonkey format, and a message pointing out this exception is displayed during conversion.

If a K-PAM file contains gaps, these are replaced by "zero blocks" (2048 X’00’ bytes) in the NK-PAM file during conversion.

Note

This statement incorporates checking functions which reject any attempt to convert LMR module libraries (PAM files) by means of CONVERT-FILE in V11.0A.
Conversion of LMR module libraries (and also MLU macro libraries) into PLAM libraries can be effected via the utility routine LMSCONV (see page 141).

Conversion of file generations

The file generations making up a file generation group can be converted by issuing a separate CONVERT-FILE statement for each file generation.

Fully automatic conversion of a file generation group with all its file generations in one pass is not possible; any such attempt is rejected with an error message.

Defining block control information during conversion

When converting from K to NK format, the block control information is set for the target file, depending on the current BS2000 version and the relevant access method:

```
= V9.5 :      to BLKCTRL = DATA for ISAM files only.
as of V10.0:  to BLKCTRL = DATA for ISAM andSAM files.
              to BLKCTRL = NO for PAM files..
```

If the file name is specified by means of a link name and the BLKCTRL parameter is used, the BLKCTRL specification must be correct.

Converting an NK file into NK format

A special case is the conversion of a file into NK format when the source file is already in NK format.
The source file is copied to the target file, but the target file receives the correct block control information, provided the operating system version permits this (see the section on "Defining block control information during conversion").

This method enables also the nonprivileged user to modify block control information (which is otherwise only possible via the MODIFY-BLKCTRL-INDICATOR statement under the TSOS ID).

Transfer of file protection attributes after conversion

Following successful conversion, the file protection attributes of a source file can be transferred to the target file. To do this, the PROTECTION operand must be specified with the SAME value in the CONVERT-FILE statement. If the user does not specify this value, the target file will be created without the file protection and file security features of the source file. This response corresponds to the functionality of PAMCONV for BS2000 versions ≤ V10.0.

File protection attributes are transferred in accordance with the specification PROTECTION=SAME in the system command /COPY-FILE (see "Commands, Volumes 1 - 3" [1], [2], [3]).
In order to be able to convert a protected file, before the PAMCONV run the user must specify all access authorizations (e.g. issuing of passwords) with the corresponding system commands. The conversion algorithm itself is then executed in the usual way, without change. At the end of a successful conversion procedure, the protection attributes of the source file are transferred to the target file.

- Transfer of file protection attributes according to user ID

  The list below is based on direct conversion (without intermediate medium) from disk to disk.

  – Conversion within any user ID

    The following protection attributes are transferred:

Protection attribute	Description
ACCESS	Standard access control; specifies whether write access (implicit read access) is permitted for the file, or read access only.
ACL	Access control list (ACL); access control for the file is implemented via an ACL entry (only possible with the software product SECOS).

Protection attribute	Description
BACKUP-CLASS	Specifies the frequency of automatic file saving with the ARCHIVE or HSMS backup system.
BASIC-ACL	Basic access control list; access control for the file is implemented via a BASIC-ACL entry. The read, write and execute access rights can be distributed among various user groups.
DESTROY-BY-DELETE	Files that are no longer required are overwritten with X'00', thereby increasing data protection.
ENCRYPTION	Password encryption
GUARDS	Access kontrol via GUARDS; GUARDS is a functional unit of the SECOS software product.
LARGE	Exent of automatic file saving with the ARCHIVE or HSMS backup system.
MIGRATE	Files are migrated to another storage level if they have not been accessed for a certain length of time.
OPNBACK	Specifies whether database files can also be saved with ARCHIVE even when they are open.
RETENTION-PERIOD	Defines a protective deadline up until which only read access to the file is allowed, i.e. it must not be modified or deleted.
USER-ACCESS	Controls access to the file via other user IDs.

Passwords cannot be transferred as part of conversion within any user ID. This is only possible during conversion under the system administration ID (TSOS).

– Conversion under the system administration ID (TSOS)

  Read (RDPASS), write (WRPASS) and execute passwords (EXPASS) are transferred, as are all protection attributes described above.

– Conversion of a source file from another user ID to the user's own ID

  The file saving attributes LARGE, BACKUP, MIGRATE and OPNBACK and the file security attributes DESTROY-BY-DELETE, RETENTION-PERIOD, RDPASS, WRPASS, EXPASS and ENCRYPTION are transferred.
  ACL, BASIC-ACL or GUARDS entries in the source file are not transferred to the target file. These entries are assigned default values in the target file, as is the file protection attribute ACCESS.
  If an existing target file already has protection entries, they will be reset before the source file entries are transferred.

– Conversion of a source file from the user's own ID to another user ID

  The protection attributes of the source file are not transferred to the target file, even if the user is authorized to create the target file.

- Restrictions applying to the transfer of file protection attributes when converting via an intermediate medium

  The transfer of file protection attributes is only supported within "one-step conversion using an intermediate file".
  The file protection attributes are not transferred in the event of "two-step conversion using an intermediate file". If PROTECTION=SAME is specified in the CONVERT-FILE statement, it is ignored.

  – One-step conversion using an intermediate medium

    The protection attributes of the source file are transferred because neither the PAMCONV run nor the current conversion command CONVERT-FILE are interrupted.
    The following processing steps are executed:

    – conversion from a common disk to an intermediate medium
    – release of the source file's storage space
    – internal transfer from the intermediate medium to the common disk
    – setting of protection attributes
    – deletion of the source file

  – Two-step conversion using an intermediate medium

    With this method of conversion both the PAMCONV run and the conversion command may be interrupted. The file protection attributes cannot be transferred when the intermediate file is output on magnetic tape or disk, for the following reasons:

    – The second conversion step (from magnetic tape or private disk to the target file) could take place at an arbitrary later time, on an arbitrary system. This may result in incompatibility, because the transfer of the NK file from the intermediate medium to a common data medium does not have to be executed with PAMCONV. Other transfer routines do not receive any information about the transfer of file protection attributes.

    – The user label of the magnetic tape on which the intermediate file is to be stored must be retained, for compatibility reasons; since further file characteristics cannot be included due to lack of space, it is impossible to transfer the protection attributes.

- Procedure with the target file after conversion

  The procedure with the target file after conversion is governed by the FILE-DISPOSAL operand in the CONVERT-FILE statement. The file protection attributes are transferred as shown below when the following operand values are specified:

FILE-DISPOSAL =	**Meaning for transfer of file protection attributes**
KEEP	Default setting. Source and target files are retained. The file protection attributes are transferred without problems.
RENAME	The source file is deleted after conversion. The protection attributes are transferred before deletion. If an error occurs during transfer, the source file is not deleted and the process is aborted with an error. The file protection attributes are therefore retained.
REPLACE	The source file is deleted after conversion and the target file is recataloged in accordance with the source file. The protection attributes are transferred before deletion. If an error occurs during transfer, the source file is not deleted and the process is aborted with an error. The file protection attributes are therefore retained.
INPLACE	After conversion, the source file is overwritten with the target file. The file protection attributes are transferred.

7.4.2 Further notes on conversion

To facilitate decision-making, PAMCONV offers a function which checks files for their convertibility.

The presence of more than one file format may lead to inconsistencies between catalog entries and file formats, in particular in multiprocessor systems with different operating system versions.

PAMCONV therefore checks the file format for consistency with the catalog entry (BLKCTRL indicator).

Since catalog updates can only be performed by privileged users, the function for changing the BLKCTRL indicator is only executable under the TSOS ID.

7.5 Statements

PAMCONV reads one statement at a time and executes it immediately. If link names are specified for a conversion, the associated TFT entries are not deleted following conversion, so that the link names can be used again.

7.5.1 Overview of the PAMCONV statements

Statement	Function	Page
CHANGE-TO-SYSTEM-MODE	Switch to system mode	312
CHECK-BLKCTRL-INDICATOR	Check file format and BLKCTRL indicator for consistency, show file format	313
CLASSIFY-FILE	Classify files according to their convertibility	320
CONVERT-FILE	Convert ISAM/SAM/PAM files or load module files	327
END	Terminate the PAMCONV program	335
HALT	Same as END	335
MODIFY-BLKCTRL-INDICATOR	Set the specified BLKCTRL indicator in the catalog entry of the file. Valid under TSOS only.	336
MODIFY-CONVERT-FILE-DEFAULTS	Change the current default values for the CONVERT-FILE statement	343
MODIFY-LOGGING-OPTIONS	Change the current logging options	351
SHOW-CONVERT-FILE-DEFAULTS	List the current default values for the CONVERT-FILE statement	352
SHOW-LOGGING-OPTIONS	List the current logging options	354
STOP	Same as END	355

Further permissible statements

In addition, the SDF standard statements such as //STEP can be used. They are of general importance in connection with SDF; for a detailed description see the manual "Introductory Guide to the SDF Dialog Interface" [13].

Restriction

The nonprivileged user can only access a limited range of functions. The MODIFY-BLKCTRL-INDICATOR statement is based on system functions and thus may only be entered by privileged users under the TSOS ID.

For BS2000 ≤ V10.0:
The functions initiated by this statement are defined only in the PAMCONV group syntax file for TSOS.

For BS2000/OSD-BC > V1.0:
The MODIFY-BLKCTRL-INDICATOR statement is available only to users with the TSOS privilege.

If a nonprivileged user enters this statement nonetheless, it is rejected.

7.5.2 Descriptions of the statements

CHANGE-TO-SYSTEM-MODE
Switch to system mode

Function

The CHANGE-TO-SYSTEM-MODE statement causes a switchover to BS2000 system mode with the subsequent possibility of entering BS2000 commands. Provided PAMCONV was not unloaded (e.g. by a START-PROGRAM or LOAD-PROGRAM command), the PAMCONV run can then be continued via the RESUME-PROGRAM command.

Format

CHANGE-TO-SYSTEM-MODE

Alternatively the SYSTEM or SYS statement can be entered to switch to system mode. These statements cannot be further abbreviated.

CHECK-BLKCTRL-INDICATOR
Check file format consistency and BLKCTRL indicator

Function

This statement checks the file format entered in the BLKCTRL indicator of the catalog against the actual file format. It also informs the user as to whether the file is in K-PAM or NK-PAM format.

Note

The default values set via the MODIFY-CONVERT-FILE-DEFAULTS statement are **not** taken into account here.

Format

```
CHECK-BLKCTRL-INDICATOR

 FROM-FILE = *LINK(...) / *ALL / <partial-filename 2..79 with-wild> / <full-filename 1..54>
    *LINK(...)
       │  LINK-NAME = <full-filename 1..8 without-gen>
,SELECT = ALL / BY-ATTRIBUTES(...)
    BY-ATTRIBUTES(...)
       │  CREATION-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
       │     INTERVAL(...)
       │        │  FROM = 0000-01-01 / <date 8..10> / YESTERDAY
       │        │  ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
       │  ,LAST-ACCESS-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
       │     INTERVAL(...)
       │        │  FROM = 0000-01-01 / <date 8..10> / YESTERDAY
       │        │  ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
```

continued ➠

```
        ,SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
            INTERVAL(...)
                FROM = 0 / <integer 0..16777215>
                ,TO = 16777215 / <integer 0..16777215>
        ,FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
        ,BLKSIZE = ANY / <integer 1..16>
,BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
```

Operands

FROM-FILE = <full-filename 1..54> / *LINK(...) / <partial-filename 2..79 with-wild> / *ALL
Designates the files to be checked.

FROM-FILE = <full-filename 1..54>
Specifies the fully qualified file name. Specification of a file generation is possible.

FROM-FILE = *LINK(...)
Identifies the files via a link name.

LINK-NAME = <full-filename 1..8 without-gen>
Specifies the link name.

FROM-FILE = <partial-filename 2..79 with-wild>
Specifies the partially qualified file name with wildcard syntax.

FROM-FILE = *ALL
All files of the user ID are to be checked.

SELECT = ALL / BY-ATTRIBUTES(...)
Specifies whether the files to be checked are selected via specific selection criteria in addition to the partially qualified file name.

SELECT = ALL
No additional selection criteria are set for the source files.

SELECT = BY-ATTRIBUTES(...)
Defines the selection criteria for the files to be checked.

CREATION-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the creation date as a selection criterion.

CREATION-DATE = ANY
The creation date is not used as a selection criterion. All files are taken into account for selection.

CREATION-DATE = INTERVAL(...)
Files with a creation date within the specified interval are checked. The interval limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / YESTERDAY / <date 8..10>
Files with a creation date equal to or later than the specified limit are checked.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = YESTERDAY
The lower limit is yesterday's date. Files with a creation date ≥ yesterday's date are checked.

FROM = <date 8..10>
The lower limit is the specified date. Files with a creation date ≥ the specified value are checked.

TO = TODAY / YESTERDAY / <date 8..10>
Files with a creation date equal to or earlier than the specified limit are checked.

TO = TODAY
The upper limit is the current date. Files with a creation date ≤ the current date are checked.

TO = YESTERDAY
The upper limit is yesterday's date. Files with a creation date ≤ yesterday's date are checked.

TO = <date 8..10>
The upper limit is the specified date. Files with a creation date ≤ the specified value are checked.

LAST-ACCESS-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the date of the last file access as a selection criterion.
For the meaning of ANY, INTERVAL(...), <date 8..10>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are checked.

SIZE = INTERVAL(...)
Files with a size within the specified range are checked. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size ≥ the specified limit are checked.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size ≤ the specified limit are checked.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
Designates the access method as selection criterion.

FILE-STRUCTURE = ANY
The access method is not used as selection criterion.

FILE-STRUCTURE = SAM
Files with the SAM access method are checked.

FILE-STRUCTURE = ISAM
Files with the ISAM access method are checked.

FILE-STRUCTURE = PAM
Files with the PAM access method are checked.

BLKSIZE = ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are checked.

BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = PAMKEY
Files with the block control attribute PAMKEY are checked.

BLKCTRL = NO
Files with the block control attribute NO are checked.

BLKCTRL = DATA
Files with the block control attribute DATA are checked.

BLKCTRL = DATA2K
Files with the block control attribute DATA2K are checked.

BLKCTRL = DATA4K
Files with the block control attribute DATA4K are checked.

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement.

– **Output of the results to SYSLST:**

A maximum of 132 characters per line is output.

```
%//CHECK-BLKCTRL-INDICATOR FROM-FILE=>from-file<
% CHECK-BLKCTRL-INDICATOR >from-file<
%
% FILENAME                                          ! FOR-   !          BLKCTRL-INDICATOR        !
%                                                   ! MAT    ! IN CATALOG ! SHOULD BE ! COMPARE  !
% ------------------------------------------------------------------------------------------------
% :CATID:$USERID.>filename 1<..................... !>format<! >catalog<  ! >should<  ! >compare<!
% :CATID:$USERID.>filename 2<..................... !>format<! >catalog<  ! >should<  ! >compare<!
% :CATID:$USERID.>filename 3<..................... !>format<! >catalog<  ! >should<  ! >compare<!
%
%
% :CATID:$USERID.>filename n<..................... !>format<! >catalog<  ! >should<  ! >compare<!
% ------------------------------------------------------------------------------------------------
%            >n<         FILE(S) LISTED
```

– **Output of the results to SYSOUT:**

A maximum of 80 characters per line is output.

```
%//CHECK-BLKCTRL-INDICATOR FROM-FILE=>from-file<
% CHECK-BLKCTRL-INDICATOR >from-file<
%
% FILENAME                                                  ! FOR-   !  BLKCTRL    !
%                                                           ! MAT    !  COMPARE    !
% ------------------------------------------------------------------------------------
% :CATID:$USERID.>filename 1<.........................!>format<!  >compare<  !
% :CATID:$USERID.>filename 2<.........................!>format<!  >compare<  !
% :CATID:$USERID.>filename 3<.........................!>format<!  >compare<  !
%
%
% :CATID:$USERID.>filename n<.........................!>format<!  >compare<  !
% ------------------------------------------------------------------------------------
%           >n<         FILE(S) LISTED
```

Meanings of the output fields:

>from-file<	File names specified in the CHECK-BLKCTRL-INDICATOR statement	
>filename<	Name of the file checked	
>n<	Total number of files checked	
>format<	K ... File has format with PAM key. NK ... File has format without PAM key	
>catalog<	Value of BLKCTRL inidcator in catalog entry. Possible values:	
	*NONE	The BLKCTRL indicator from the catalog entry is not available (in BS2000 < V9.5)
	PAMKEY	The BLKCTRL indicator from the catalog entry has the value PAMKEY
	DATA	The BLKCTRL indicator from the catalog entry has the values DATA. Possible for PAM, SAM or ISAM in BS2000/OSD-BC < V1.0.
	DATA4K	The BLKCTRL indicator from the catalog entry has the value DATA4K. Possible only for ISAM and BS2000/OSD-BC ≥ V1.0.
	NO	The BLKCTRL indicator in the catalog entry has the value NO.
>should<	BLKCTRL indicator value the file ought to have in accordance with the file structure. Possible values:	
	PAMKEY	The BLKCTRL indicator in the catalog entry should have the value PAMKEY.
	DATA	The BLKCTRL indicator from the catalog entra has the value DATA. Possible for PAM, SAM and ISAM in BS2000/OSD-BC < V1.0.

>should<	DATA4K	The BLKCTRL indicator in the catalog entry should have the value DATA4K. Possible only for ISAM and BS2000/OSD-BC ≥ V1.0.
	NO	The BLKCTRL indicator in the catalog entry should have the value NO.
>compare<	Value comparsion, possible values:	
	SAME	The BLKCTRL indicator in the catalog entry corresponds to that of the actual file structure.
	DIFFERENT	The BLKCTRL indicator iin the catalog entry does not correspond to that of the actual file structure.

CLASSIFY-FILE
Classify files by convertibility

Function

On each selected file, information as to its convertibility and any incompatibilities is requested.

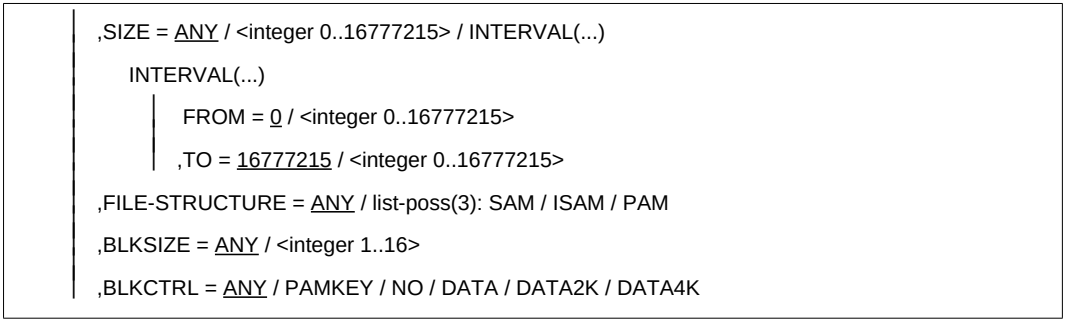
Note

The default values set via the MODIFY-CONVERT-FILE-DEFAULTS statement are **not** taken into account here. The results of the CLASSIFY-FILE statement indicate the convertibility of a source file. For similar information on the target file, see section “Reblocking” on page 301.

Format

```
CLASSIFY-FILE

 DIRECTION = TO-NONKEY / NONKEY-TO-KEY
,FROM-FILE = *LINK(...) / *ALL / <partial-filename 2..79 with-wild> / <full-filename 1..54>
    *LINK(...)
        | LINK-NAME = <full-filename 1..8 without-gen>
,SELECT = ALL / BY-ATTRIBUTES(...)
    BY-ATTRIBUTES(...)
        | CREATION-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
        |    INTERVAL(...)
        |        | FROM = 0000-01-01 / <date 8..10> / YESTERDAY
        |        | ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
        |  ,LAST-ACCESS-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
        |    INTERVAL(...)
        |        | FROM = 0000-01-01 / <date 8..10> / YESTERDAY
        |        | ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
```

continued ➠



Operands

DIRECTION = TO-NONKEY / NONKEY-TO-KEY
Designates the user-defined direction of file conversion. This must be specified here because it influences the classification type.

DIRECTION = TO-NONKEY
Files are to be converted to NK format.

DIRECTION = NONKEY-TO-KEY
Files are to be converted from NK to K format.

FROM-FILE = <full-filename 1..54> / *LINK(...) /
<partial-name 2..79 with-wild> / *ALL
Defines the files to be checked.

FROM-FILE = <full-filename 1..54>
Designates the fully qualified file name. Specification of a file generation is possible.

FROM-FILE = *LINK(...)
Identifies the files via a link name.

LINK-NAME = <full-filename 1..8 without-gen>
Specifies the link name.

FROM-FILE = <partial-filename 2..79 with-wild>
Identifies the partially qualified file name with wildcard syntax.

FROM-FILE = *ALL
All files of the user ID are to be checked.

SELECT = ALL / BY-ATTRIBUTES(...)
Specifies whether the files to be classified are selected via specific selection criteria in addition to the partially qualified file name.

SELECT = ALL
No additional selection criteria are set for the source files.

SELECT = BY-ATTRIBUTES(...)
Defines the selection criteria for the files to be classified.

CREATION-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the creation date as selection criterion.

CREATION-DATE = ANY
The creation date is not used as a selection criterion. All files are taken into account for selection.

CREATION-DATE = INTERVAL(...)
Files with a creation date within the specified interval are selected. The interval limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / YESTERDAY / <date 8..10>
Files with a creation date equal to or later than the specified limit are selected.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = YESTERDAY
The lower limit is yesterday's date. Files with a creation date ≥ yesterday's date are selected.

FROM = <date 8..10>
The lower limit is the specified date. Files with a creation date ≥ the specified value are selected.

TO = TODAY / YESTERDAY / <date 8..10>
Files with a creation date equal to or earlier than the specified limit are selected.

TO = TODAY
The upper limit is the current date. Files with a creation date ≤ the current date are selected.

TO = YESTERDAY
The upper limit is yesterday's date. Files with a creation date ≤ yesterday's date are selected.

TO = <date 8..10>
The upper limit is the specified date. Files with a creation date ≤ the specified value are selected.

LAST-ACCESS-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the date of the last file access as a selection criterion.
For the meaning of ANY, INTERVAL(...), <date>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are selected.

SIZE = INTERVAL(...)
Files with a size within the specified range are selected. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size ≥ the specified limit are selected.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size ≤ the specified limit are selected.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
Designates the access method as selection criterion.

FILE-STRUCTURE = ANY
The access method is not used as selection criterion.

FILE-STRUCTURE = SAM
Files with the SAM access method are selected.

FILE-STRUCTURE = ISAM
Files with the ISAM access method are selected.

FILE-STRUCTURE = PAM
Files with the PAM access method are selected.

BLKSIZE = ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are selected.

BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = PAMKEY
Files with the block control attribute PAMKEY are selected.

BLKCTRL = NO
Files with the block control attribute NO are selected.

BLKCTRL = DATA
Files with the block control attribute DATA are selected.

BLKCTRL = DATA2K
Files with the block control attribute DATA2K are selected.

BLKCTRL = DATA4K
Files with the block control attribute DATA4K are selected.

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement.

The results of the check are output to SYSLST in the following form: (line length up to 132 characters)

```
%//CLASSIFY-FILE   FROM-FILE=>from-file<
% CLASSIFY-FILE    >from-file<    DIRECTION = >direction<
%
% FILENAME                                          !  PAM-  !FCB- !CONVER-!  INCOMPATIBILITIES  !
%                                                   !  PAGES !TYPE !TIBLE  !                     !
% ------------------------------------------------------------------------------------------------
% :CATID:$USERID.>filename 1< ..................... ! >size< !>fcb<!>yesno<!      >reason<       !
% :CATID:$USERID.>filename 2< ..................... ! >size< !>fcb<!>yesno<!      >reason<       !
% :CATID:$USERID.>filename 3< ..................... ! >size< !>fcb<!>yesno<!      >reason<       !
%                                                   !        !     !       !                     !
%                                                   !        !     !       !                     !
% :CATID:$USERID.>filename n< ..................... ! >size< !>fcb<!>yesno<!      >reason<       !
% ------------------------------------------------------------------------------------------------
%            >n<       FILE(S) LISTED
```

The results of the check are output to SYSOUT in the following form: (line length up to 80 characters)

```
% //CLASSIFY-FILE   FROM-FILE=>from-file<
% CLASSIFY-FILE   >from-file<    DIRECTION = >direction<
%
% FILENAME                                                        ! CONVER- !
%                                                                 ! TIBLE   !
% ----------------------------------------------------------------------------
% :CATID:$USERID.>filename 1< ..............................  ! >yesno< !
% :CATID:$USERID.>filename 2< ..............................  ! >yesno< !
% :CATID:$USERID.>filename 3< ..............................  ! >yesno< !
%
%
% :CATID:$USERID.>filename n< ..............................  ! >yesno< !
% ----------------------------------------------------------------------------
%            >n<       FILE(S) LISTED
```

Meanings of the output fields:

>from-file<	File names specified in the CLASSIFY-FILE statement	
>direction<	Conversion direction specified in the CLASSIFY-FILE statement	
>filename<	Name of the file checked	
>size<	Size of the file checked	
>fcb<	FCB type of the file checked	
>n<	Total number of files checked	
>yesno<	File convertibility information. Possible values:	
	YES	File is convertible
	NO	File is not convertible
	NK2	File is only convertible on NK2 pubsets with standard blksize (PLAM libraries).
	NK4 ...	File is only convertible on NK2 pubsets (e.g. SAM files with BLKSIZE=RECSIZE and standard blksize).
>reason<	Reason for incompatibility. Possible values:	
	NONE	No incompatibilities, file is convertible
	RECSIZE EXCEEDS MAXIMUM	The record length exceeds the maximum value determined by BLKSIZE. The file can be converted by increasing the blocking factor.
	RECSIZE EXCEEDS MAX(NK2)	The record length would exceed the maximum value determined by BLKSIZE on NK2 pubsets. It is possible to convert to NK4 pubsets.

>reason<	PLAM(NK4) NO CONVERT PLAM libraries with BLKSIZE > 2 are not converted with PAMCONV.
	FILE ALREADY IN KEY-FORMAT Since the file is already in K format, it cannot be converted into K format.
	FILESIZE INCREASES The formation of overflow blocks increases the size of the target file, the file is convertible.
	KEYPOS IN OVERFLOW-BLOCK The ISAM key would have to be stored in an overflow block. The file can be converted by increasing the blocking factor.
	LMR-LIBRARY File is an LMR library, i.e. it is not convertible.
	PAMKEY IS USED The file makes use of the user part of the PAM key, file is not convertible
	PAMKEY CONTAINS SPECIAL FLAG The file contains X'01' or X'80' only in byte 1 of the user part of the PAM key, file is convertible

CONVERT-FILE
Convert files

Function

This statement serves to convert files from a format in which the PAM key is used for data representation into a format in which the PAM key is not used, or vice versa.
As of PAMCONV V11.0A, the CONVERT-FILE statement is also used for reblocking (see section “Reblocking” on page 301).

The CONVERT-FILE statement now offers three options:

– **Conversion**: changes the file format, i.e. converts a file from K format to NK format or vice versa.

– **Reblocking**: changes the logical block size without changing the file format.

– **Conversion and reblocking**

The default values for the CONVERT-FILE statement are set using the MODIFY-CONVERT-FILE-DEFAULTS statement.

The default values indicated by underscoring take effect only if no other values have been specified.

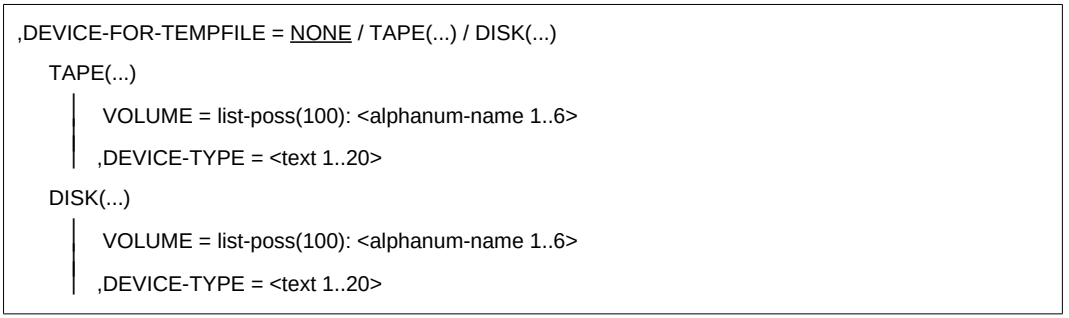
Format

```
CONVERT-FILE

 DIRECTION = TO-NONKEY / NONKEY-TO-KEY
,FROM-FILE = <full-filename 1..54> / *LINK(...) / <partial-filename 2..79 with-wild> / *ALL
    *LINK(...)
      │ LINK-NAME = <full-filename 1..8 without-gen>
```

continued ➠

```
,SELECT = ALL / BY-ATTRIBUTES(...)
    BY-ATTRIBUTES(...)
        CREATION-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
            INTERVAL(...)
                FROM = 0000-01-01 / <date 8..10> / YESTERDAY
                ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY

        ,LAST-ACCESS-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
            INTERVAL(...)
                FROM = 0000-01-01 / <date 8..10> / YESTERDAY
                ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
        ,SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
            INTERVAL(...)
                FROM = 0 / <integer 0..16777215>
                ,TO = 16777215 / <integer 0..16777215>
        ,FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
        ,BLKSIZE = ANY / <integer 1..16>
        ,BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
,TO-FILE = <full-filename 1..54> / *LINK(...) / <partial-filename 2..79 with-wild>
    *LINK(...)
        LINK-NAME = <full-filename 1..8 without-gen>
,TO-FILE-BLKSIZE = STD / NK4 / <integer 1..16>
,TO-FILE-BLKCTRL = STD / NK4
,REPLACE-OLD-FILES = NO / YES / DIALOG
,FILE-DISPOSAL = KEEP / REPLACE / INPLACE / RENAME
,PROTECTION = STD / SAME
```

continued ➠



Operands

DIRECTION = TO-NONKEY / NONKEY-TO-KEY
Designates the direction in which file conversion is to take place.

DIRECTION = TO-NONKEY
The file is to be converted into NK format. The source file may be in K format or NK format.

DIRECTION = NONKEY-TO-KEY
File conversion is to be from NK to K format.

FROM-FILE = <full-filename 1..54> / *LINK(...) /
<partial-filename 2..79 with-wild> / *ALL
Identifies the files to be converted.

FROM-FILE = <full-filename 1..54>
Designates the file to be converted. Specification of a file generation is possible.

FROM-FILE = *LINK(...)
The file to be converted was specified by a previous ADD-FILE-LINK command; the link name given there must be identical to the one specified here.

LINK-NAME = <full-filename 1..8 without-gen>
Specifies the link name.

FROM-FILE = <partial-filename 2..79 with-wild>
Means that all files corresponding to the specified wildcard syntax and to any additional selection criteria are to be converted.

FROM-FILE = *ALL
Means that all files corresponding to the specified selection criteria are to be converted.

SELECT = ALL / BY-ATTRIBUTES(...)
Defines whether the files to be converted are to be selected via specific criteria in addition to the partially qualified file name.

SELECT = ALL
No additional selection criteria are specified for the source files.

SELECT = BY-ATTRIBUTES(...)
Defines the selection criteria for the files to be converted.

CREATION-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the creation date as a selection criterion.

CREATION-DATE = ANY
The creation date is not used as a selection criterion. All files are taken into account for selection.

CREATION-DATE = INTERVAL(...)
Files with a creation date within the specified interval are selected. The interval limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / YESTERDAY / <date 8..10>
Files with a creation date equal to or later than the specified limit are selected.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = YESTERDAY
The lower limit is yesterday's date. Files with a creation date ≥ yesterday's date are selected.

FROM = <date 8..10>
The lower limit is the specified date. Files with a creation date ≥ the specified value are selected.

TO = TODAY / YESTERDAY / <date 8..10>
Files with a creation date equal to or earlier than the specified limit are selected.

TO = TODAY
The upper limit is the current date. Files with a creation date ≤ the current date are selected.

TO = YESTERDAY
The upper limit is yesterday's date. Files with a creation date ≤ yesterday's date are selected.

TO = <date 8..10>
The upper limit is the specified date. Files with a creation date ≤ the specified value are selected.

LAST-ACCESS-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the date of the last file access as a selection criterion.
For the meaning of ANY, INTERVAL(...), <date 8...10>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are selected.

SIZE = INTERVAL(...)
Files with a size within the specified range are selected. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size ≥ the specified limit are selected.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size ≤ the specified limit are selected.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
Designates the access method as a selection criterion.

FILE-STRUCTURE = ANY
The access method is not used as a selection criterion.

FILE-STRUCTURE = SAM
Files with the SAM access method are selected.

FILE-STRUCTURE = ISAM
Files with the ISAM access method are selected.

FILE-STRUCTURE = PAM
Files with the PAM access method are selected.

BLKSIZE = ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are selected.

BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = PAMKEY
Files with the block control attribute PAMKEY are selected.

BLKCTRL = NO
Files with the block control attribute NO are selected.

BLKCTRL = DATA
Files with the block control attribute DATA are selected.

BLKCTRL = DATA2K
Files with the block control attribute DATA2K are selected.

BLKCTRL = DATA4K
Files with the block control attribute DATA4K are selected.

TO-FILE = <full-filename 1..54> / *LINK(...) / <partial-filename 2..79 with-wild>
Identifies the files to be created by conversion.

TO-FILE = <full-filename 1..54>
Designates the file to be created by conversion. Specification of a file generation is possible.

TO-FILE = *LINK(...)
The file to be created was specified by a previous ADD-FILE-LINK command; the link name given there must be identical to the one specified here.

LINK-NAME = <full-filename 1..8 without-gen>
Specifies the link name.

TO-FILE = <partial-filename 2..79 with-wild>
Specifies the files to be created by conversion in the form of a partial qualification with wildcard syntax.

TO-FILE-BLKSIZE = STD / NK4 / <integer 1..16>
Specifies the logical block size of the target file.

TO-FILE-BLKSIZE = STD
The logical block size of the target file is not defined by the user. PAMCONV uses the values defined for the target pubset and, if necessary, increases the blocking factor. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = NK4
The logical block size of the target file is controlled in such a way that it is always even, i.e. the target file can be stored on an NK4 pubset. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = <integer 1..16>
The target file is generated with a block size equal to the specified value, provided that this specification is compatible with the other conditions governing reblocking (see section “Reblocking” on page 301).

TO-FILE-BLKCTRL = STD / NK4
Specifies the block control indicator of the target file. This operand is relevant only for the conversion direction TO-NONKEY and for ISAM files.

TO-FILE-BLKCTRL = STD
The block control indicator is set in accordance with the target pubset. The DATA2K data format is defined for NK2 and NK2(8K,64K) pubsets, and DATA4K for NK4 pubsets.

TO-FILE-BLKCTRL = NK4
The block control indicator is assigned the value DATA4K.

REPLACE-OLD-FILES = NO / YES / DIALOGUE
Indicates whether any files existing under this name are to be overwritten.

REPLACE-OLD-FILES = NO
Existing files must not be overwritten; file conversion is aborted in this case.

REPLACE-OLD-FILES = YES
Existing files are overwritten unless additional protection (password, ACCESS=READ) has been provided.

REPLACE-OLD-FILES = DIALOGUE
The system issues a query as to the desired procedure for existing files with the same name. Possible in interactive mode only.

FILE-DISPOSAL = KEEP / RENAME / REPLACE / INPLACE
Determines what happens to the file after conversion.

FILE-DISPOSAL = KEEP
The target files are to be created with the names specified for them in the conversion statement. The target files exist in addition to the source files.

FILE-DISPOSAL = RENAME
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted.

FILE-DISPOSAL = REPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted and the target files are to receive the names of the source files, i.e. in effect the source file is replaced by the target file.

FILE-DISPOSAL = INPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, an attempt is to be made to overwrite the source file with the target file and to assign the source file name to the target file. As a result, the target file will occupy roughly the same physical space as the source file. In effect, the source file is replaced by the target file.

PROTECTION = STD / SAME
Specifies whether the file protection attributes of the source file are to be transferred to the target file.

PROTECTION = STD
The file protection attributes are not transferred. Conversion is executed as in BS2000 versions ≤ V10.0.

PROTECTION = SAME
The file protection attributes are transferred to the target file. For more details see "Transfer of file protection attributes after conversion" on page 307.

DEVICE-FOR-TEMPFILE = NONE / TAPE(...) / DISK(...)
Designates the storage medium which is to accommodate the intermediate (temporary) file created.

DEVICE-FOR-TEMPFILE = NONE
No intermediate file is to be stored on a private volume.

DEVICE-FOR-TEMPFILE = TAPE(...)
The intermediate file is to be stored on magnetic tape.

VOLUME = list-poss(100): <alphanum-name 1..6>
Designates the VSN(s) of the tape(s) to be used as storage medium.

DEVICE-TYPE = <text 1..20>
Designates the device type to be used.

DEVICE-FOR-TEMPFILE = DISK(...)
The intermediate file is to be stored on private disk.

VOLUME = list-poss(100): <alphanum-name 1..6>
Designates the VSN(s) of the private disk(s) to be used as storage medium.

DEVICE-TYPE = <text 1..20>
Designates the device type to be used.

END
Terminate PAMCONV

Function

The END statement terminates the PAMCONV program.

Format

END

The END statement has no operands.

HALT
Terminate PAMCONV

Function

The HALT statement terminates the PAMCONV program. It is a synonym of the END statement but cannot be abbreviated and does not appear in the menu screen of available PAMCONV statements.

Format

HALT

The HALT statement has no operands.

MODIFY-BLKCTRL-INDICATOR
Change value in BLKCTRL indicator of catalog entry for file

Function

This statement is available to privileged (TSOS) users only.

The value entered by the system in the catalog entry of the specified files is replaced by the value specified by system administration for the file format in the BLKCTRL indicator field.

The actual file format is **not** checked.
System administration is responsible for preventing inconsistencies resulting from invalid specifications.

Note

The default values set via the MODIFY-CONVERT-FILE-DEFAULTS statement are **not** taken into account here.

Format

```
MODIFY-BLKCTRL-INDICATOR

 FROM-FILE = *LINK(...) / *ALL / <partial-filename 2..79 with-wild> / <full-filename 1..54>
    *LINK(...)
      | LINK-NAME = <full-filename 1..8 without-gen>
,SELECT = ALL / BY-ATTRIBUTES(...)
    BY-ATTRIBUTES(...)
      | CREATION-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
      |   INTERVAL(...)
      |     | FROM = 0000-01-01 / <date 8..10> / YESTERDAY
      |     | ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
```

continued ➠

```
        ,LAST-ACCESS-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
            INTERVAL(...)
                FROM = 0000-01-01 / <date 8..10> / YESTERDAY
                ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
        ,SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
            INTERVAL(...)
                FROM = 0 / <integer 0..16777215>
                ,TO = 16777215 / <integer 0..16777215>
        ,FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
        ,BLKSIZE = ANY / <integer 1..16>
        ,BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
,BLKCTRL-INDICATOR = PAMKEY / NO / DATA / DATA4K
```

Operands

FROM-FILE = <full-filename 1..54> / *LINK(...) /
<partial-filename 2..79 with-wild> / *ALL
Designates the files whose catalog entry is to be modified.

FROM-FILE = <full-filename 1..54>
Specifies the fully qualified file name. Specification of a file generation is possible.

FROM-FILE = *LINK(...)
Identifies the files via a link name.

LINK-NAME = <full-filename 1..8 without-gen>
Specifies the link name.

FROM-FILE = <partial-filename 2..79 with-wild>
Designates the partially qualified file name with wildcard syntax.

FROM-FILE = *ALL
The catalog entries of all files of the user ID are to be changed.

SELECT = ALL / BY-ATTRIBUTES(...)
Specifies whether the files whose catalog entry is to be changed are selected via specific selection criteria in addition to the partially qualified file name.

SELECT = ALL
No additional selection criteria are set for the source files.

SELECT = BY-ATTRIBUTES(...)
Determines the selection criteria for the files whose catalog entry is to be changed.

CREATION-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the creation date as selection criterion.

CREATION-DATE = ANY
The creation date is not used as selection criterion. All files are taken into account for selection.

CREATION-DATE = INTERVAL(...)
Files with a creation date within the specified interval are selected. The interval limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / YESTERDAY / <date 8..10>
Files with a creation date equal to or later than the specified limit are selected.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = YESTERDAY
The lower limit is yesterday's date. Files with a creation date ≥ yesterday's date are selected.

FROM = <date 8..10>
The lower limit is the specified date. Files with a creation date ≥ the specified value are selected.

TO = TODAY / YESTERDAY / <date 8..10>
Files with a creation date equal to or earlier than the specified limit are selected.

TO = TODAY
The upper limit is the current date. Files with a creation date ≤ the current date are selected.

TO = YESTERDAY
The upper limit is yesterday's date. Files with a creation date ≤ yesterday's date are selected.

TO = <date 8..10>
The upper limit is the specified date. Files with a creation date ≤ the specified value are selected.

LAST-ACCESS-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the date of the last file access as selection criterion.
For the meaning of ANY, INTERVAL(...), <date 8...10>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are selected.

SIZE = INTERVAL(...)
Files with a size within the specified range are selected. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size ≥ the specified limit are selected.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size ≤ the specified limit are selected.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
Designates the access method as a selection criterion.

FILE-STRUCTURE = ANY
The access method is not used as a selection criterion.

FILE-STRUCTURE = SAM
Files with the SAM access method are selected.

FILE-STRUCTURE = ISAM
Files with the ISAM access method are selected.

FILE-STRUCTURE = PAM
Files with the PAM access method are selected.

BLKSIZE = ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are selected.

BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = PAMKEY
Files with the block control attribute PAMKEY are selected.

BLKCTRL = NO
Files with the block control attribute NO are selected.

BLKCTRL = DATA
Files with the block control attribute DATA are selected.

BLKCTRL = DATA2K
Files with the block control attribute DATA2K are selected.

BLKCTRL = DATA4K
Files with the block control attribute DATA4K are selected.

BLKCTRL-INDICATOR = PAMKEY / DATA / DATA4K / NO
Identifies the value of the BLKCTRL indicator which is to be entered in the catalog entry.

BLKCTRL-INDICATOR = PAMKEY
The BLKCTRL indicator is to have the value PAMKEY.

BLKCTRL-INDICATOR = DATA
The BLKCTRL indicator is to have the value DATA.

BLKCTRL-INDICATOR = DATA4K
The BLKCTRL indicator is to have the value DATA4K (for ISAM files only).

BLKCTRL-INDICATOR = NO
The BLKCTRL indicator is to have the value NO.

The files whose catalog entry was modified are listed.

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement.

The results are output via

– **SYSLST: (line length up to 132 characters)**

```
% MODIFY-BLKCTRL-INDICATOR >from-file<
%
% FILENAME               ! BLKCTRL!MODIFIED !
%                        ! FROM   !  TO     !
%-----------------------------------------
% >filename 1<            ! >old<  ! >new<   !
% >filename 2<           ! >old<  ! >new<   !
%                        !        !         !
% >filename n<           ! >old<  ! >new<   !
%-----------------------------------------
% >n<     FILES LISTED
```

– **SYSOUT: (line length up to 80 characters)**

```
% MODIFY-BLKCTRL-INDICATOR >from-file<
%
% FILENAME               ! BLKCTRL!MODIFIED !
%                        ! FROM   !  TO     !
%-----------------------------------------
% >filename 1<            ! >old<  ! >new<   !
% >filename 2<           ! >old<  ! >new<   !
%                        !        !         !
% >filename n<           ! >old<  ! >new<   !
%-----------------------------------------
% >n<     FILES LISTED
```

Meanings of the output fields:

>from-file<	File names given in the MODIFY-BLKCTRL-INDICATOR statement	
>filename<	Name of the modified file	
>n<	Total number of files modified	
>old<	Value of the BLKCTRL indicator as stored in the catalog entry, possible values:	
	PAMKEY	The BLKCTRL indicator from the catalog entry has the value PAMKEY
	DATA	The BLKCTRL iindicator from the catalog has the value DATA
	DATA4K	The BLKCTRL indicator from the catalog entry has the value DATA4K (for ISAM files only)
	NO	The BLKCTRL indicator from the catalog entry has the value NO
>new<	Value of the BLKCTRL indicator as updated in the catalog entry of the file, possible values:	
	see >old<	

When changing the BLKCTRL indicator on private disks (SPD), error messages with the following return codes may occur.

Sub-return code:

04 Device allocation failed (no device available).
Notify operator.

08 Volume allocation failed (volume not available).
Notify operator.

Main return code:

333 File name does not exist.

MODIFY-CONVERT-FILE-DEFAULTS
Set default values for CONVERT-FILE statement

Function

This statement sets the default values for the CONVERT-FILE statement. The specified defaults then apply to the current program run until the next MODIFY-CONVERT-FILE-DEFAULTS statement is issued.

If the statement is entered without operands, the existing default values remain valid.

The currently applicable values may be queried using the SHOW-CONVERT-FILE-DEFAULTS statement.

Note

The default values set here apply **only** to the CONVERT-FILE statement.
The CLASSIFY-FILE, CHECK-BLKCTRL-INDICATOR and MODIFY-BLKCTRL-INDICATOR statements are not influenced by this statement.

Format

```
MODIFY-CONVERT-FILE-DEFAULTS

 DIRECTION = UNCHANGED / TO-NONKEY / NONKEY-TO-KEY
,SELECT = UNCHANGED / ALL / BY-ATTRIBUTES(...)
    BY-ATTRIBUTES(...)
    |   CREATION-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
    |   |   INTERVAL(...)
    |   |   |   FROM = 0000-01-01 / <date 8..10> / YESTERDAY
    |   |   |   ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
```

continued ➠

```
        ,LAST-ACCESS-DATE = ANY / <date 8..10> / TODAY / YESTERDAY / INTERVAL(...)
            INTERVAL(...)
                FROM = 0000-01-01 / <date 8..10> / YESTERDAY
                ,TO = TODAY / <date 8..10> / TODAY / YESTERDAY
        ,SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
            INTERVAL(...)
                FROM = 0 / <integer 0..16777215>
                ,TO = 16777215 / <integer 0..16777215>
        ,FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
        ,BLKSIZE = ANY / <integer 1..16>
        ,BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
,TO-FILE-BLKSIZE = UNCHANGED / STD / NK4 / <integer 1..16>
,TO-FILE-BLKCTRL = UNCHANGED / STD / NK4
,REPLACE-OLD-FILES = UNCHANGED / NO / YES / DIALOG
,FILE-DISPOSAL = UNCHANGED / KEEP / REPLACE / INPLACE / RENAME
,PROTECTION = UNCHANGED / STD / SAME
,DEVICE-FOR-TEMPFILE = UNCHANGED / NONE / TAPE(...) / DISK(...)
    TAPE(...)
        VOLUME = list-poss(100): <alphanum-name 1..6>
        ,DEVICE-TYPE = <text 1..20>
    DISK(...)
        VOLUME = list-poss(100): <alphanum-name 1..6>
        ,DEVICE-TYPE = <text 1..20>
```

Operands

DIRECTION = UNCHANGED / TO-NONKEY / NONKEY-TO-KEY
Designates the desired default value for the direction of file conversion.

DIRECTION = UNCHANGED
The current default value for DIRECTION is not changed.

DIRECTION = TO-NONKEY
This file is to be converted into NK format.

DIRECTION = NONKEY-TO-KEY
This file is to be converted from NK to K format.

SELECT = UNCHANGED / ALL / BY-ATTRIBUTES(...)
Specifies whether the files to be converted are selected via specific selection criteria in addition to the partially qualified file name.

SELECT = UNCHANGED
The current default value for SELECT is not changed.

SELECT = ALL
No selection criteria are set for the source files.

SELECT = BY-ATTRIBUTES(...)
Defines the selection criteria for the files to be converted and which are to apply as default values for the CONVERT-FILE statement.

CREATION-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the creation date as a selection criterion.

CREATION-DATE = ANY
The creation date is not used as a selection criterion. All files are taken into account for selection.

CREATION-DATE = INTERVAL(...)
Files with a creation date within the specified interval are selected. The interval limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / YESTERDAY / <date 8..10>
Files with a creation date equal to or later than the specified limit are selected.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = YESTERDAY
The lower limit is yesterday's date. Files with a creation date ≥ yesterday's date are selected.

FROM = <date 8..10>
The lower limit is the specified date. Files with a creation date ≥ the specified value are selected.

TO = TODAY / YESTERDAY / <date 8..10>
Files with a creation date equal to or earlier than the specified limit are selected.

TO = TODAY
The upper limit is the current date. Files with a creation date ≤ the current date are selected.

TO = YESTERDAY
The upper limit is yesterday's date. Files with a creation date ≤ yesterday's date are selected.

TO = <date 8..10>
The upper limit is the specified date. Files with a creation date ≤ the specified value are selected.

LAST-ACCESS-DATE = ANY / INTERVAL(...) / <date 8..10> / TODAY / YESTERDAY
Designates the date of the last file access as a selection criterion.
For the meaning of ANY, INTERVAL(...), <date 8...10>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = ANY / <integer 0..16777215> / INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are selected.

SIZE = INTERVAL(...)
Files with a size within the specified range are selected. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size ≥ the specified limit are selected.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size ≤ the specified limit are selected.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = ANY / list-poss(3): SAM / ISAM / PAM
Designates the access method as a selection criterion.

FILE-STRUCTURE = ANY
The access method is not used as a selection criterion.

FILE-STRUCTURE = SAM
Files with the SAM access method are selected.

FILE-STRUCTURE = ISAM
Files with the ISAM access method are selected.

FILE-STRUCTURE = PAM
Files with the PAM access method are selected.

BLKSIZE = ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are selected.

BLKCTRL = ANY / PAMKEY / NO / DATA / DATA2K / DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = PAMKEY
Files with the block control attribute PAMKEY are selected.

BLKCTRL = NO
Files with the block control attribute NO are selected.

BLKCTRL = DATA
Files with the block control attribute DATA are selected.

BLKCTRL = DATA2K
Files with the block control attribute DATA2K are selected.

BLKCTRL = DATA4K
Files with the block control attribute DATA4K are selected.

TO-FILE-BLKSIZE = UNCHANGED / STD / NK4 / <integer 1..16>
Specifies the logical block size of the target file.

TO-FILE-BLKSIZE = UNCHANGED
The default value currently valid for BLKSIZE-TO-FILE is to remain unchanged.

TO-FILE-BLKSIZE = STD
The logical block size of the target file is not defined by the user. PAMCONV uses the values defined for the target pubset and, if necessary, increases the blocking factor. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = NK4
The logical block size of the target file is controlled in such a way that it is always even, i.e. the target file can be stored on an NK4 pubset. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = <integer 1..16>
The target file is generated with a block size equal to the specified value, provided that this specification is compatible with the other conditions governing reblocking (see section “Reblocking” on page 301).

TO-FILE-BLKCTRL = UNCHANGED / STD / NK4
Specifies the block control indicator of the target file. This operand is relevant only for the conversion direction TO-NONKEY and for ISAM files.

TO-FILE-BLKCTRL = UNCHANGED
The default value currently valid for TO-FILE-BLKCTRL is to remain unchanged.

TO-FILE-BLKCTRL = STD
The block control indicator is set in accordance with the target pubset. The DATA2K data format is defined for NK2 and NK2(8K,64K) pubsets, and DATA4K for NK4 pubsets.

TO-FILE-BLKCTRL = NK4
The block control indicator is assigned the value DATA4K.

REPLACE-OLD-FILES = UNCHANGED / NO / YES / DIALOGUE
Indicates whether any files existing under this name are to be overwritten.

REPLACE-OLD-FILES = UNCHANGED
The current default value for REPLACE-OLD-FILES is not changed.

REPLACE-OLD-FILES = NO
Existing files must not be overwritten; file conversion is to be aborted in this case (default value).

REPLACE-OLD-FILES = YES
Existing files are overwritten unless additional protection (password, ACCESS=READ) has been provided.

REPLACE-OLD-FILES = DIALOGUE
The system issues a query as to the desired procedure for existing files with the same name. Possible in interactive mode only.

FILE-DISPOSAL = UNCHANGED / KEEP / RENAME / REPLACE / INPLACE
Determines what happens to the file after conversion.

FILE-DISPOSAL = UNCHANGED
The current default value for FILE-DISPOSAL is not changed.

FILE-DISPOSAL = KEEP
The target files are to be created with the names specified for them in the conversion statement. The target files exist in addition to the source files.

FILE-DISPOSAL = RENAME
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted.

FILE-DISPOSAL = REPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted and the target files are to receive the names of the source files, i.e. in effect the source file is replaced by the target file.

FILE-DISPOSAL = INPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, an attempt is to be made to overwrite the source file with the target file and to assign the source file name to the target file. As a result, the target file will occupy about the same physical space as the source file. In effect, the source file is replaced by the target file.

PROTECTION = UNCHANGED / STD / SAME
Specifies whether the file protection attributes of the source file are to be transferred to the target file.

PROTECTION = UNCHANGED
The currently valid default value for PROTECTION is to remain unchanged.

PROTECTION = STD
The file protection attributes are not transferred. Conversion is executed as in BS2000 versions ≤ V10.0.

PROTECTION = SAME
The file protection attributes are transferred to the target file. For more details see "Transfer of file protection attributes after conversion" on page 307.

DEVICE-FOR-TEMPFILE = UNCHANGED / NONE / TAPE(...) / DISK(...)
Designates the storage medium which is to accommodate the intermediate (temporary) file created.

DEVICE-FOR-TEMPFILE = UNCHANGED
The current default value for DEVICE-FOR-TEMPFILE is not changed.

DEVICE-FOR-TEMPFILE = NONE
No intermediate file is to be stored on a private volume.

DEVICE-FOR-TEMPFILE = TAPE(...)
The intermediate file is to be stored on magnetic tape.

VOLUME = list-poss(100): <alphanum-name 1..6>
Designates the VSN(s) of the tape(s) to be used as storage medium.

DEVICE-TYPE = <text 1..20>
Designates the device type to be used.

DEVICE-FOR-TEMPFILE = DISK(...)
The intermediate file is to be stored on private disk.

VOLUME = list-poss(100): <alphanum-name 1..6>
Designates the VSN(s) of the private disk(s) to be used as storage medium.

DEVICE-TYPE = <text 1..20>
Designates the device type to be used.

Note

The volume specified in the DEVICE-FOR-TEMPFILE operand is only reserved/released during execution of the CONVERT-FILE statement.

MODIFY-LOGGING-OPTIONS
Set logging values

Function

This statement defines logging values for PAMCONV which have global validity for all functions of the program. If the statement is entered without operands, the existing values remain valid. The currently applicable values can be queried using the SHOW-LOGGING-OPTIONS statement.

Format

MODIFY-LOGGING-OPTIONS
INFORMATION = UNCHANGED / MEDIUM / MINIMUM / MAXIMUM ,OUTPUT = UNCHANGED / list-poss(2): SYSOUT / SYSLST

Operands

INFORMATION = UNCHANGED / MEDIUM / MINIMUM / MAXIMUM
Controls the scope of the log created by PAMCONV.

INFORMATION = UNCHANGED
The current default value for INFORMATION is not changed.

INFORMATION = MEDIUM
Statements are logged in the case of errors only. Positive acknowledgments (message class 5) are logged in addition to MINIMUM.

INFORMATION = MINIMUM
Only error messages, completion messages and negative acknowledgments are logged (i.e. all message classes except class 5).

INFORMATION = MAXIMUM
The log created consists of statements, positive and negative acknowledgments, error messages and completion messages.

OUTPUT = UNCHANGED / list-poss(2): SYSOUT / SYSLST
Defines the output medium for the logs created by PAMCONV.

OUTPUT = UNCHANGED
The current default value for OUTPUT is not changed.

OUTPUT = list-poss(2): SYSOUT / SYSLST
Either one output medium or a list of 2 output media (SYSOUT and SYSLST) may be entered. If SYSOUT is specified, logs are output on the display terminal in interactive mode (in batch mode they are written to the SYSOUT system file). If SYSLST is specified, logs are written to the SYSLST system file. In the case of a list, the logs are output to the appropriate system files (in interactive mode also to the terminal).

Note

The following values are preset:
INFORMATION = MEDIUM , OUTPUT = SYSOUT

SHOW-CONVERT-FILE-DEFAULTS
List current default values for CONVERT-FILE statement

Function

The SHOW-CONVERT-FILE-DEFAULTS statement lists the currently valid default values for the CONVERT-FILE statement.

Format

SHOW-CONVERT-FILE-DEFAULTS

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement.

The default values are listed in the following form:

```
%//SHOW-CONVERT-FILE-DEFAULTS
% CURRENT   CONVERT-FILE   DEFAULTS
%   DIRECTION                  : <value>
%   SELECT                     : <value>
%   TO-FILE-BLKSIZE            : <value>
%   TO-FILE-BLKCTRL            : <value>
%   REPLACE-OLD-FILES          : <value>
%   FILE-DISPOSAL              : <value>
%   PROTECTION                 : <value>
%   DEVICE-FOR-TEMPFILE        : <value>
```

<value> ... see MODIFY-CONVERT-FILE-DEFAULTS for possible values.

Note

The following text segments have been defined in the message file and are taken from there:

– ’CURRENT CONVERT-FILE DEFAULTS’
– ’DIRECTION’
– ’SELECT’
– ’TO-FILE-BLKSIZE’
– ’TO-FILE-BLKCTRL’
– ’REPLACE-OLD-FILES’
– ’FILE-DISPOSAL’
– ’PROTECTION’
– ’DEVICE-FOR-TEMPFILE’

If the text is to be changed, these segments can be redefined in the message file. The SHOW-CONVERT-FILE-DEFAULTS statement can then be used to check whether the user-defined modification of statement and operand names in the SDF syntax file has been duly performed. This also facilitates work with multilingual output.

SHOW-LOGGING-OPTIONS
List specified logging options

Function

A list of the currently valid values for logging is requested.

Format

SHOW-LOGGING-OPTIONS

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement.

The specified values are listed in the following form:

```
%//SHOW-LOGGING-OPTIONS
%CURRENT LOGGING OPTIONS
%  INFORMATION : <value>
%  OUTPUT      : <value>
```

<value> ... see MODIFY-LOGGING-OPTIONS for possible values.

Note

The text segments

– 'CURRENT LOGGING OPTIONS'
– 'INFORMATION'
– 'OUTPUT'

have been defined in the message file and are taken from there. If the text is to be changed, these segments can be redefined in the message file by system administration. The SHOW-LOGGING-OPTIONS statement can then be used to check whether the user-defined modification of statement and operand names in the SDF syntax file has been duly performed. This also facilitates work with multilingual output.

STOP
Terminate PAMCONV

Function

The STOP statement terminates the PAMCONV program. It is a synonym of the END statement but cannot be abbreviated and does not appear in the menu screen of available PAMCONV statements.

Format

STOP

The STOP statement has no operands.

7.6 PAMCONV program execution

The program reads the control statements via SYSDTA.

Messages are output via SYSOUT and/or SYSLST, depending on the logging options specified (see the MODIFY-LOGGING-OPTIONS and SHOW-LOGGING-OPTIONS statements).

Example

```
/START-PAMCONV
%  BLS0500 PROGRAM 'PAMCONV', VERSION '11.0A' OF '1993-03-11' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993.
   ALL RIGHTS RESERVED
%  PEA7000 15:41:51/0.0223 PAMCONV VERSION V11.0A64 STARTED IN BS2000 V11.2
%  PEA7001 PLEASE ENTER PAMCONV STATEMENTS ————————————————————————————  (1)

%//CONVERT-FILE FROM-FILE=DAT*,SELECT=BY-ATTRIBUTES(CREATION-DATE=
   INTERVAL(,YESTERDAY),FILE-STRUCTURE=SAM),TO-FILE=NK.DAT* ———————————  (2)

%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI1'; TARGET FILE: ':N:$USER0001.NK.DATEI1'
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI2'; TARGET FILE: ':N:$USER0001.NK.DATEI2'
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI3'; TARGET FILE: ':N:$USER0001.NK.DATEI3'
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI4'; TARGET FILE: ':N:$USER0001.NK.DATEI4'  ———————  (3)

%//CONVERT-FILE FROM-FILE=DAT*,TO-FILE=NK.DAT* ———————————————————————  (4)

%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI5'; TARGET FILE: ':N:$USER0001.NK.DATEI5' ————————  (5)

%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI1'
%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI2'
%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI3'
%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI4' ——————————  (6)

%//MODIFY-CONVERT-FILE-DEFAULTS SELECT=BY-ATTRIBUTES(FILE-STRUCTURE=ISAM) (7)
```

```
%//SHOW-CONVERT-FILE-DEFAULTS ------------------------------------------ (8)

% CURRENT   CONVERT-FILE   DEFAULTS
%   DIRECTION                   : TO-NONKEY
%   SELECT                      : BY-ATTRIBUTES( )
%      CREATION-DATE            : ANY
%      LAST-ACCESS-DATE         : ANY
%      SIZE                     : ANY
%      ACCESS-METHOD            : ISAM
%      BLKSIZE                  : ANY
%      BLKCTRL                  : ANY
%   TO-FILE-BLKSIZE             : STD
%   TO-FILE-BLKCTRL             : STD
%   REPLACE-OLD-FILES           : NO
%   FILE-DISPOSAL               : KEEP
%   PROTECTION                  : STD
%   DEVICE-FOR-TEMPFILE         : NONE ---------------------- (9)

%//CONVERT-FILE DIRECTION=NONKEY-TO-KEY,FROM-FILE=NK.DAT*,TO-FILE=K.DAT* (10)

%  PEA5001 CONVERSION FROM NON-KEY TO KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.NK.DATEI5'; TARGET FILE: ':N:$USER0001.K.DATEI5' ------ (11)

%//CLASSIFY-FILE FROM-FILE=*DAT* ---------------------------------------- (12)

% CLASSIFY-FILE   *DAT*  DIRECTION = KEY-TO-NONKEY
%
% FILENAME                                                  ! CONVER- !
%                                                           ! TIBLE   !
-------------------------------------------------------------------------
% :N:$USER0001.DATEI1 ..................................    !   YES   !
% :N:$USER0001.DATEI2 ..................................    !   YES   !
% :N:$USER0001.DATEI3 ..................................    !   YES   !
% :N:$USER0001.DATEI4 ..................................    !   YES   !
% :N:$USER0001.DATEI5 ..................................    !   YES   !
% :N:$USER0001.K.DATEI5 ................................    !   YES   !
% :N:$USER0001.NK.DATEI1 ...............................    !   YES   !
% :N:$USER0001.NK.DATEI2 ...............................    !   YES   !
% :N:$USER0001.NK.DATEI3 ...............................    !   YES   !
% :N:$USER0001.NK.DATEI4 ...............................    !   YES   !
% :N:$USER0001.NK.DATEI5 ...............................    !   YES   !
-------------------------------------------------------------------------
%            11      FILE(S)LISTED ------------------------------------ (13)
```

```
%//CHECK-BLKCTRL-INDICATOR FROM-FILE=*DAT* ------------------------------ (14)
% CHECK-BLKCTRL-INDICATOR    *DAT*
%
% FILENAME                                              ! FOR- !  BLKCTRL    !
%                                                       ! MAT  !  COMPARE    !
-------------------------------------------------------------------------------
% :N:$USER0001.DATEI1 ..............................!   K  !       SAME  !
% :N:$USER0001.DATEI2 ..............................!   K  !       SAME  !
% :N:$USER0001.DATEI3 ..............................!   K  !       SAME  !
% :N:$USER0001.DATEI4 ..............................!   K  !       SAME  !
% :N:$USER0001.DATEI5 ..............................!   K  !       SAME  !
% :N:$USER0001.K.DATEI5 ............................!   K  !       SAME  !
% :N:$USER0001.NK.DATEI1 ...........................!  NK  !       SAME  !
% :N:$USER0001.NK.DATEI2 ...........................!  NK  !       SAME  !
% :N:$USER0001.NK.DATEI3 ...........................!  NK  !       SAME  !
% :N:$USER0001.NK.DATEI4 ...........................!  NK  !       SAME  !
% :N:$USER0001.NK.DATEI5 ...........................!  NK  !  DIFFERENT  !
-------------------------------------------------------------------------------
%           11        FILE(S)LISTED ------------------------------------- (15)
%//END ---------------------------------------------------------------------- (16)
%  PEA7003 15:46:32/1.5323 PAMCONV TERMINATED ABNORMALLY ------------------- (17)
```

(1) The PAMCONV program is called.

(2) The CONVERT-FILE statement is entered with partially qualified file names and selection criteria.

(3) Acknowledgment of successful file conversion.

(4) Input of the CONVERT-FILE statement with partially qualified file names without further selection criteria.

(5) Acknowledgments of successful file conversions.

(6) File exists already, conversion is rejected.

(7) Definition of selection criteria for further CONVERT-FILE statements.

(8) Input of the SHOW-CONVERT-FILE-DEFAULTS statement.

(9) Output of the values requested via the SHOW-CONVERT-FILE-DEFAULTS statement. The values shown are the defaults for subsequent CONVERT-FILE statements.

(10) Input of the CONVERT-FILE statement with partially qualified file names for NK to K conversion.

(11) Acknowledgments of successful file conversions.

(12) Input of the CLASSIFY-FILE statement with partially qualified file names.

(13) Output of the values requested via the CLASSIFY-FILE statement. Classification of the input files by convertibility.

(14) Input of the CHECK-BLKCTRL-INDICATOR statement with partially qualified file names.

(15) Output of the results requested via the CHECK-BLKCTRL-INDICATOR statement (check of the internal file format and comparison with the BLKCTRL value from the catalog entry).

(16) Input of the END statement.

(17) The PAMCONV program is terminated abnormally, since an error has occurred in the PAMCONV run.

7.7 Error handling

7.7.1 Control statement errors in interactive mode

– Syntax errors
  Errored control statements are rejected with appropriate error messages.
– Semantic errors
  An error dialog is conducted with the user.

7.7.2 Control statement errors in batch mode

Syntax and/or semantic errors.
As soon as an errored statement is detected, all statements up to the next STEP or END statement are skipped.

7.7.3 Errors during conversion

If errors occur during processing of a CONVERT-FILE statement before the target file has been successfully generated and closed, the incomplete target file (or the intermediate file) is deleted and the source file is retained regardless of the explicitly or implicitly (default value) specified FILE-DISPOSAL operand of the MODIFY-CONVERT-FILE-DEFAULTS statement. Successful closure of the target file is acknowledged with a corresponding message.

The following error situations are possible after successful closure of the target file (or intermediate file):

– The source file cannot be closed properly. Conversion has been successfully concluded, but the activities specified explicitly or implicitly (default value) in the FILE-DISPOSAL operand are not performed. Successful closure of the source file is acknowledged with a corresponding message.

– Both the source and the target file have been successfully closed, and in accordance with the conversion specification of the FILE-DISPOSAL operand the source file is to be replaced with the target file (REPLACE or RENAME), but the source file cannot be deleted. The same procedure as above applies.

– The source file is to be replaced with the target file; the source file has been deleted, but the renaming operation in accordance with the conversion value REPLACE in the FILE-DISPOSAL operand cannot be performed. In this case the end result is the same as if RENAME had been specified in the FILE-DISPOSAL operand.

Subsequently the next statement is executed.

7.7.4 DMS error messages, error code

DMS error messages are output for the user (missing passwords, ACCESS=READ,...). The DMS error code reported is output with additional PAMCONV-specific information in certain cases.

7.7.5 Inconsistent files

If the source file is an ISAM/SAM file or a load module, it is subjected to a strict check for formal consistency during conversion. Inconsistencies may be subdivided into two groups:

– The inconsistency is such that loss of data is to be expected during further conversion (e.g. in the case of defective index entries). Conversion is aborted with an appropriate message.

– There is a formal inconsistency, but loss of data can be excluded (e.g. reserved fields or free blocks do not contain binary zero). In this case a warning is output and conversion continues.

Subsequently the next statement is executed.

7.7.6 Incompatible files

Operation of NK formats is not fully compatible. Consequently, various incompatibilities result from conversion into NK format. See the section on PAM key elimination in the manual "Introductory Guide to DMS" [3].

As soon as incompatibilities are sensed which prevent proper conversion, the conversion process is aborted with an appropriate message (hard incompatibility).

If incompatibilities are detected which reduce performance but do not essentially prevent conversion, a warning is issued and conversion continues (soft incompatibility).

Subsequently the next statement is executed.

7.7.7 Messages

The messages of the PAMCONV program have a 7-character code consisting of the string PEA followed by a 4-digit hexadecimal number. These messages can be found in volume 2 of the "System Messages" manual [17].

8 PASSWORD
Password encryption routine

Version: **PASSWORD V11.2A**

This chapter describes the password encryption routine PASSWORD.

The PASSWORD routine encrypts passwords in a system without automatic password encryption, thus enabling files to be read that were created in a system with password encryption.

PASSWORD is a service routine for use when exporting files from one system to another (e.g. with the ARCHIVE routine). PASSWORD is needed only if files that have been created and protected in a system with password encryption are to be processed in a system without password encryption. PASSWORD encrypts the specified passwords and enters them in the password table of the job.

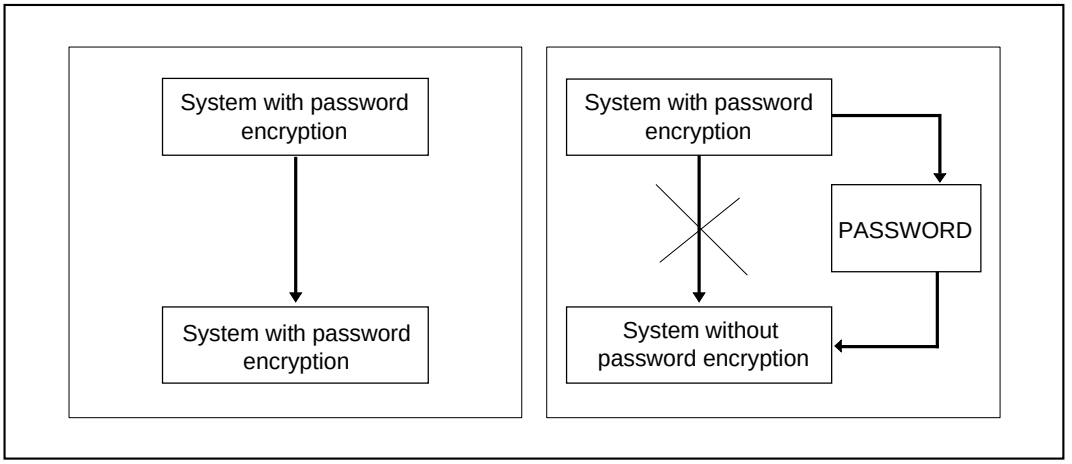


Figure 13: Password processing in systems with and without password encryption

8.1 Operation and execution

A file or LOGON password specified by a user is written into the catalog entry of the file or the user catalog, respectively. In systems operating with password encryption, the password entered by the user is encrypted according to an internal code and transferred in this form to the catalog entry or user catalog. The user must enter the password in its unencrypted form when calling the file or entering the SET-LOGON-PARAMETERS command, since the encryption is performed internally by the system.
The purpose of this procedure is to prevent unauthorized persons from discovering passwords if parts of the catalog or user catalog happen to be included in the output when a dump is taken.

When files with encrypted passwords are imported into a system operating without password encryption, a user who specifies the unencrypted password will not be given access to the file because the password is contained in its encrypted form in the catalog. The function of PASSWORD is to convert the unencrypted passwords into encrypted passwords in order to allow the file to be accessed.

8.1.1 Passwords

PASSWORD processes file passwords and LOGON passwords.

File passwords protect files against

– unauthorized writing (write password, WRITE-PASSWORD operand of the MODIFY-FILE-ATTRIBUTES command)
– unauthorized reading (read password, READ-PASSWORD operand of the MODIFY-FILE-ATTRIBUTES command)
– unauthorized execution in the case of program and procedure files (execute password, EXEC-PASSWORD operand of the MODIFY-FILE-ATTRIBUTES command).

The power of these file passwords is weighted with respect to access. Write passwords are the most powerful, read passwords the next most powerful, then execute passwords. A less powerful password does not have to be specified explicitly if a more powerful one has already been specified.

File passwords are defined for one or more files. A check is made for the presence of the correct passwords prior to execution of the system commands START-PROGRAM, LOAD-PROGRAM, MODIFY-FILE-ATTRIBUTES, DELETE-FILE, CREATE-FILE, ADD-FILE-LINK, COPY-FILE and SHOW-FILE, and also at file opening.

To avoid repeated specification of file passwords for system commands, the password can be written to the password table during task execution. This table is then searched for the appropriate password whenever a password-protected file is to be processed.

The **LOGON password** is a password that must be entered with the SET-LOGON-PARAMETERS command in order to ensure that only authorized users may work under a particular user ID.

LOGON passwords contained in ENTER procedures are evaluated only if the ENTER job was called from an operator terminal. Otherwise all LOGON operands specified in the ENTER procedure, including the LOGON password, are ignored.

Format conventions for passwords

File passwords and job variable passwords must not exceed 4 bytes in length. They are represented in the following format:

C'x' where x stands for 1 to 4 alphanumeric or special characters

X'n' where n stands for 1 to 8 hexadecimal digits

d where d stands for a decimal number of up to 8 digits (with or without a positive or negative sign) whose value is converted to a binary value

LOGON passwords may be between 1 and 8 characters long. For security reasons, the maximum length should always be used. The passwords are represented in the following format:

C'x' where x stands for 1 to 8 alphanumeric or special characters

X'n' where n stands for 1 to 16 hexadecimal digits

8.1.2 Program execution

The PASSWORD routine is loaded and started with the system command:

```
/START-PROGRAM FROM-FILE=$PASSWORD
```

This produces the message:

```
BLS0500 PROGRAM 'PASSWORD', VERSION '11.2A00' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
ENTER COMMAND NOW :
*
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in the "Commands, Volume 3" manual [3].

When the program has been called, you are prompted to input statements.

Each statement issued to PASSWORD is executed immediately. Any output is displayed on the terminal (SYSOUT).

Once a statement has been completely executed, a new PASSWORD statement may be entered or the routine may be terminated with the END statement. Normal termination is indicated by the message:

```
PASSWORD : NORMAL END
```

8.2 Statements

Statements for the PASSWORD routine are entered via SYSDTA.

8.2.1 Overview of the PASSWORD statements

Control statements for PASSWORD are entered via SYSDTA.

Operation	Operands	Brief description
CONVERT	filename [,WRPASS=fpassword] [,RDPASS=fpassword] [,EXPASS=fpassword]	The most powerful password for the specified file is entered in the password table in its encrypted form. The MODIFY-FILE-ATTRIBUTES command containing the passwords is issued for the file.
JVCONV	jvname [,WRPASS=jpasswordt] [,RDPASS=jpassword]	The most powerful password for the specified job variable is entered in the password table in its encrypted form. The MODIFY-JV-ATTRIBUTES command containing the passwords is then issued for the job variable.
{ ENCPASS | EP }	dpassword	Encrypts the specified password and enters the result in the password table.
{ ENCRYPTD | ED }	dpassword	Encrypts the specified file password and writes the result to SYSOUT.
{ ENCRYPTJ | EJ }	lpassword	Encrypts the specified LOGON password and writes the result to SYSOUT.
END		Terminates the PASSWORD run.
{ HELP | H }	statement	Outputs a brief description of all statements or of the specified statement.

Operation	Operands	Brief description
{ MODE / M }	mode	Determines the encryption algorithm.
PASSWORD	dpassword	Encrypts the specified file password and enters both the password and its encrypted form in the password table.

Note

fpassword stands for "file password".

jpassword stands for "job variable password".

lpassword stands for "LOGON password".

8.2.2 Description of the individual statements

CONVERT statement

The CONVERT statement encrypts the most powerful password (see also the ADD-PASSWORD command in the manual "Commands, Volume 1" [1] and writes it to the password table by issuing the ADD-PASSWORD command. Then a MODIFY-FILE-ATTRIBUTES command is issued containing the specified passwords.

Operation	Operands
CONVERT	fileiname, { WRPASS=fpassword | RDPASS=fpassword | EXPASS=fpassword }

file	Fully qualified name of the file whose passwords are to be encrypted.
WRPASS=fpassword	Specifies the write password to be encrypted.
RDPASS=fpassword	Specifies the read password to be encrypted.
EXPASS=fpassword	Specifies the execute password to be encrypted.

Note

fpassword is a file password consisting of 1 to 4 bytes (see "Format conventions for passwords" on page 365). At least one of the three passwords must be specified.

Example

```
BLS0500 PROGRAM 'PASSWORD', VERSION '11.2A00' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
ENTER COMMAND NOW :
*convert anne.3,rdpass=c'susi'
PASSWORD X'9A41632A'
CAT     ANNE.3,RDPASS=C'SUSI',STATE=U
*end
PASSWORD : NORMAL  END
/
```

JVCONV statement

The JVCONV statement is used to encrypt the password(s) for a specified job variable. A password must be specified in addition to the job variable name. The most powerful password is encrypted and entered in the password table by means of the ADD-PASSWORD command. Then a MODIFY-JV-ATTRIBUTES command is issued.

Operation	Operands
JVCONV	jvname { ,WRPASS=jpassword / ,RDPASS=jpassword }

jvname	Name of the job variable whose passwords are to be encrypted.
WRPASS=jpassword	Specifies the write password to be encrypted.
RDPASS=jpassword	Specifies the read password to be encrypted.

Note

jpassword is a job variable password consisting of 1 to 4 bytes (see "Format conventions for passwords" on page 365). At least one of the two passwords must be specified.

END statement

The END statement terminates the PASSWORD run.

Operation	Operands
END	

Normal termination of the routine produces the message:

```
PASSWORD : NORMAL END
```

ENCPASS statement

This statement encrypts the specified file password and enters the result in the password table of the task. The encrypted password is also output to SYSOUT.

Operation	Operands
{ ENCPASS | EP }	fpassword

fpassword File password consisting of 1 to 4 bytes (see "Format conventions for passwords" on page 365).

Example

```
BLS0500 PROGRAM 'PASSWORD', VERSION '11.2A00' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
ENTER COMMAND NOW :
*encpass c'susi'
PASSWORD X'9A41632A'
*end
PASSWORD : NORMAL   END
/
```

ENCRYPTD statement

This statement encrypts the specified file password and outputs the result to SYSOUT.

Function

A protected file from a system operating with password encryption is transferred (e.g. by ARCHIVE) to a system not using password encryption. The catalog entry of the file contains an encrypted password, i.e. access to the file is possible only by specifying the encrypted password.

The ENCRYPTD statement encrypts the original password. By entering the encrypted password the user regains access to the file.

Operation	Operands
{ ENCRYPTD ED }	fpassword

fpassword — File password consisting of 1 to 4 bytes (see "Format conventions for passwords" on page 365).

Example

```
BLS0500 PROGRAM 'PASSWORD', VERSION '11.2A00' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
ENTER COMMAND NOW :
*encryptd c'susi'
PASSWORD ENCRYPTED IS = X'9A41632A'
*end
PASSWORD : NORMAL   END
/
```

ENCRYPTJ statement

This statement encrypts the specified LOGON password and outputs the result to SYSOUT.

Operation	Operands
{ ENCRYPTJ | EJ }	lpassword

lpassword LOGON password consisting of 1 to 8 bytes (see "Format conventions for passwords" on page 365).

Note

As of BS2000/OSD-BC V2.0, LOGON passwords of up to 32 characters can be declared. To be able to edit them with PASSWORD, they must be shortened to 8 characters at system level. The charegable product SDF-P V2.0 provides the predefined function HASH-STRING() which enables this conversion.

```
&(TO-X-LIT(HASH-STRING('<long password>',8)))
```

Example 1

Encryption of a password 4 characters long.

```
BLS0500 PROGRAM 'PASSWORD', VERSION '11.2A00' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
ENTER COMMAND NOW :
*encryptj c'susi'
PASSWORD ENCRYPTED IS = X'6B0537211705D615'
*end
PASSWORD : NORMAL   END
/
```

Example 2

Encryption of a password 13 characters long.

```
/A=&(TO-X-LIT(HASH-STRING('long password',8)))
/show-var a,value=*x-literal
A = X'523E146036CED784'

/start-prog $password
%  BLS0500 PROGRAM 'PASSWORD', VERSION '11.2A00' OF '1994-11-17' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
ENTER COMMAND NOW :
*encryptj x'523E146036CED784'
PASSWORD ENCRYPTED IS = X'73FE57A922EA780D'
*end
PASSWORD : NORMAL   END
/
```

HELP statement

This statement lists either all control statements for PASSWORD with their permitted operands, or all additional operands of a specified statement.

Operation	Operands
{ HELP | H }	statement

statement — Full name of a PASSWORD statement, i.e.:

CONVERT
JVCONV
ENCRYPTD
ENCRYPTJ
ENCPASS
MODE
PASSWORD
END

Note, however, that the short form of each statement, as shown in the statement formats, is equally valid as input.

MODE statement

MODE selects the encryption routine to be used in BS2000. The choice is between the "old" and the "SCA85" encryption routine.

Operation	Operands
{ MODE / M }	mode

mode — The following may be specified for "mode":

{ OLD / O } — This means that the predefined default routine is to be chosen.

SCA — This means that the SCA85 encryption routine is to be used.

PASSWORD statement

This statement encrypts the specified file password. Both the encrypted and the unencrypted password are entered in the password table of the job. Both forms of the password are also output to SYSOUT.

This means that although the file password is contained in its encrypted form in the catalog, the user can specify the unencrypted password when accessing the file, without the access being rejected as unauthorized.

Operation	Operands
PASSWORD	fpassword

fpassword File password consisting of 1 to 4 bytes (see "Format conventions for passwords" on page 365).

Example

```
BLS0500 PROGRAM 'PASSWORD', VERSION '11.2A00' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
ENTER COMMAND NOW :
*password c'susi'
PASSWORD X'9A41632A'
PASSWORD C'SUSI'
*end
PASSWORD : NORMAL   END
/
```

9 PDPOOLS
Creating and managing private disk pools

Version: **PDPOOLS V11.2A**

The PDPOOLS utility routine (Private Disk POOL Support) is used to create and manage private disk pools. PDPOOLS can run only under the user ID TSOS and is available in different versions according to the version of BS2000. A set of up to 32 private disks can be declared as a pool. Pool monitoring prevents a file that extends over several volumes (multi-volume file) from beginning on a volume of one pool and continuing on a volume of another pool. In extreme cases even a single disk can be declared a "pool", thereby preventing multivolume files from being set up at all.

PDPOOLS supports reconstruction of a pool where a private disk has been physically destroyed.
A defective volume requiring re-initialization with VOLIN is removed from the pool by a special function.
To prevent subsequent duplicate allocations, all multivolume files that were continued on the defective disk are deleted.

The SVL (standard volume label) of each volume of a pool contains a directory of all volume serial numbers belonging to the pool. When new pools are formed, reference is made to this directory so as to prevent overlaps.

Note

For compatibility reasons, private disks not assigned to a pool by the PDPOOLS routine are handled as though they belong to every pool. This may produce files which cannot be opened or extended. For this reason it makes sense to combine all the private disks of a computer center into pools, if possible.

Calling PDPOOLS

The PDPOOLS utility routine can be started in two ways:

```
/START-PDPOOLS
```

You receive the followIng messages:

```
%  BLS0523 ELEMENT 'PDPOOLLM', VERSION 'V11.2R10' OF LIBRARY
           ':1SBZ:$TSOS.SYSLNK.PDPOOLS.112' IN PROGRESS
%  BLS0524 LLM 'PDPOOLS', VERSION 'V11.2' OF '1995-03-29:19:50:49' LOADED
%  BLS0551 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993. ALL
           RIGHTS RESERVED
P D P O O L S  VERS. 11.2A40
ENTER „HELP“ OR „HELP <CMD>“ TO GET MORE INFORMATION
*
```

In addition, the VERSION, MONJV and CPU-LIMIT operands are available for calling the routine.

or

```
/START-PROGRAM $PDPOOLS
```

You receive the followIng messages:

```
%  BLS0500 PROGRAM 'PDPOOLS', VERSION '11.2A' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993. ALL
RIGHTS RESERVED
P D P O O L S  VERS. 11.2A20
ENTER „HELP“ OR „HELP <CMD>“ TO GET MORE INFORMATION
*
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in the "Commands, Volume 3" manual [3].

You are prompted to enter statements.

9.1 Statements

9.1.1 Overview of the PDPOOLS statements

Statement	Meaning
BKPT	Interrupt PDPOOLS in interactive mode
CHECK	Check the pool for consistency
CREATE	Create or extend pools
EDT	Call the EDT file editor
END	Terminate PDPOOLS
EXPORT	Remove catalog entries from the system catalog
HELP	Output explanations
LIST	List directory or multivolume files
MODIFY	Modify default parameters
PATCH	Output information on corrections made in patch form
PURGE	Release dead space, remove a pool directory
REMOVE	Remove VSNs from a pool

9.1.2 Description of the individual statements

PDPOOLS is controlled by statements entered in interactive mode at the terminal. In batch mode input is initially read from SYSDTA; when end-of-file is detected for SYSDTA, a switch is made to input from the console. Here output is also to console but may be redirected to SYSLST by the user. Apart from a few exceptions the statements can be formatted freely. In interactive mode a slash (/) in position 1 is ignored.

BKPT
Interrupt PDPOOLS in interactive mode

In interactive mode the BKPT (BreaK PoinT) statement is used to interrupt PDPOOLS. In batch mode a PASS loop is formed. Control can be returned to PDPOOLS by means of the BS2000 command SEND-MSG.

Operation	Operands
{ / BKPT }	

"/" is recognized in place of "BKPT" only in position 1.

I/O control

The following inputs, which are recognized only in position 1, are used for input/output control:

Operation	Operands
{ = - # * }	

= The entire ensuing dialog with PDPOOLS is additionally logged to SYSLST. Not logged, however, are input/ output operations during execution of EDT and while in system mode (after a BKPT statement), as well as BS2000 command output.

– This activates SYSLST logging and at the same time suppresses output of all subsequent PDPOOLS messages to SYSOUT or to the console. If SYSLST logging was already activated by means of "=", the only effect is that SYSOUT/console output is deactivated. It can be reactivated with "=" (SYSLST logging is continued) or with "#" (SYSLST logging is terminated).

SYSLST logging activated by "=" or "-" is deactivated.

* When PDPOOLS detects SYSDTA EOF, it requests further input via the WRTRD macro with the message:

```
PDPOOLS INPUT REQUEST
```

The input request can be switched back to RDATA by entering an asterisk (*).

CHECK
Check pool for consistency

The CHECK statement can be used to check a pool for consistency. Among other checks, the VTOC of each volume is read to ascertain whether there are multivolume files that are incompatible with the directory in the SVL of the volume.

Operation	Operands
{ CH | CHECK }	POOL={ vsn | (vsn,dc) | (vsn,[dc],vsn,[dc],...) }

POOL

=vsn
=(vsn,dc)
=(vsn,[dc],vsn,[dc],...)

Specifies the pool involved. The directory entries are always used in order to determine all volumes of the pool whenever possible. The pool is always complete when no overlapping occurs.
For an explanation of "dc", see the CREATE statement.

CREATE
Create or extend pools

The CREATE statement is used to set up or extend pools. The SVL of each volume contains a directory of all volume serial numbers on which multivolume files beginning on that volume may be continued. Normally all directories of a pool contain the same volume serial numbers, so that specifying a single volume is sufficient to define the pool. If, however, overlapping is permitted, the SVL entries of two volumes of the same pool can be of different lengths. This is the only indication of pool overlapping. An explicit entry of the number of pools to which a volume belongs is not made. Overlapping pools can be created only when expressly requested by system administration (MODE=MULTIPLE). However, such pools should generally be avoided since the danger is always present that not all relevant volumes will be recorded if reconstruction is necessary.

Operation	Operands
{ CR / CREATE }	[{ vsn / (vsn,dc) } ,] POOL= { vsn / (vsn,dc) / (vsn,[dc],vsn,[dc],...) } [,DIRECTORY= { Y[ES] / N[O] }] [,MODE= { S[INGLE] / M[ULTIPLE] }]

vsn — The volume serial number specified is to be added to the pool defined by POOL=.... If DIRECTORY=NO is specified, the result is the same if "vsn" is specified under POOL=... instead of additionally.

(vsn,dc) — This specification, where "dc" = device type, is required if volume "vsn" is not available online.

POOL

=vsn

=(vsn,dc) Specifies which volume serial number is to be used to generate the pool. The result depends upon the specification for DIRECTORY. For "dc", see above.

=(vsn,[dc],vsn,[dc],...)
Specifies a number of volume serial numbers to be used to generate the pool. For "dc", see above.

DIRECTORY

=YES Specifies that the directories of the volume serial numbers specified under POOL=... are to be taken into account when the pool is generated (default value).

=NO Specifies that the pool is to be generated without taking the directory entries into account. This specification is mandatory if none of the volumes listed under POOL= has a directory.

MODE

=SINGLE When a pool is generated, overlapping with a second pool is not permitted and will be rejected with an error message (default value).

=MULTIPLE Pool overlapping is permitted.

The volumes required for execution of the CREATE statement are reserved exclusively. If this is not possible because, for example, a volume may be accessed by several jobs, the statement is rejected. However, if the disk is mounted on an SPD device, the volume is locked against any other access while the CREATE statement is being executed.

Example of pools without overlap

Using the volumes A and B and also C and D, none of which has a directory, 2 separate pools are to be created. The following statements are required to do this:

```
CR  POOL=(A,,B),DIRECTORY=NO
CR  POOL=(C,,D),DIRECTORY=NO
```

The following directories are generated on the four volumes:

```
A:  A, B
B:  B, A
C:  C, D
D:  D, C
```

If a single pool is now to be formed from the four volumes, this can be done with the following statement:

```
CR POOL=(A,,C)
```

The following directories are then written to the SVLs:

```
A:  A, B, C, D
B:  B, C, D, A
C:  C, D, A, B
D:  D, A, B, C
```

Example of pools with overlap

The two pools (A, B) and (C, D) of the previous example are used once again. If a new pool is to be created from volumes A and C alone, this can be achieved by means of one of the two statements:

```
CR  POOL=(A,,C),DIRECTORY=NO,MODE=MULTIPLE
CR  C,POOL=A,DIRECTORY=NO,MODE=MULTIPLE
```

The directories of the four volumes are then as follows:

```
A:  A, B, C
B:  B, A
C:  C, D, A
D:  D, C
```

If a new pool is to be created from volumes A, B and C of the two pools (A, B) and (C, D), the following statement must be issued:

```
CR C,POOL=A,MODE=MULTIPLE
```

The volume serial number C is thus added to pool (A,B). The four volumes contain the directories:

```
A:  A, B, C
B:  B, A, C
C:  C, D, A, B
D:  D, C
```

EDT
Call EDT file editor

The EDT statement calls the EDT file editor. Control is returned to PDPOOLS by means of the EDT statement @RETURN or @HALT. This does not cause destruction of the data in the virtual memory of EDT.

Operation	Operands
{ @ / EDT }	

The entry "@" instead of "EDT" is recognized only in position 1.

END
Terminate PDPOOLS

The END statement terminates PDPOOLS.

Operation	Operands
{ E / END }	

If //E ..... is entered, the BS2000 command ENTER-JOB is executed.

EXPORT
Remove catalog entries

The EXPORT statement can be used to remove all file catalog entries referring to a particular VSN from the system catalog. The volume itself is not affected.

Operation	Operands
{ EXP / EXPORT }	vsn[,{ ALL / VTOC }] [,OUTPUT={ NO / ERROR / ALL / (ERROR [,{ SYSOUT / SYSLST }]) / (ALL [,{ SYSOUT / SYSLST }]) }] [,CATID=catid]

vsn — Specifies the volume for which the EXPORT function is to be performed.

ALL — Specifies that all catalog entries with a pointer to "vsn" are to be removed from the system catalog (default value).

VTOC — Specifies that only those catalog entries also contained in the VTOC of "vsn" are to be removed from the system catalog.

OUTPUT

=NO — No check list of the exported files is output.

=ERROR

=(ERROR,SYSOUT)
A check list of all the errors that occurred during export is output to SYSOUT.
ERROR is the default value.

=ALL

=(ALL,SYSOUT) — A check list of all exported files is output to SYSOUT.

OUTPUT=(...,SYSLST)

The check list is output to SYSLST.

CATID=catid — Specifies the catalog from which the catalog entries are to be removed. If this entry is omitted, the user's default catalog ID will be assumed.

HELP
Output explanations

The HELP statement is used to output explanations for all statements.

Operation	Operands
{ H / HELP }	[stmt]

stmt — An explanation of the specified statement is provided.

If no statement is specified, a summary of all statements is given.

LIST
List directory or multivolume files

The LIST statement is used to list the directories or the multivolume files in the VTOC of a volume.

Operation	Operands
{ L / LIST }	{ vsn / (vsn,dc) } [, { CATL / (CATL,SYSOUT) / (CATL,SYSLST) / EXTS / POOL / MVFS / (MVFS,SYSOUT) / (MVFS,SYSLST) }][,CATID=catid]

vsn

(vsn,dc) Specifies the volume for which the LIST function is to be performed. For an explanation of "dc", see the CREATE statement.

CATL Specifies that a list of the system catalog entries for "vsn"

(CATL,SYSOUT) is to be output, or, if "vsn" is omitted, a list of the entries for all private disks. The list contains the number of users sharing the volume, the number of first entries (also contained in the VTOC of the volume) and the number of second entries (not contained in the VTOC of the volume). The list thus also provides an overview of the volumes for which multivolume files are cataloged.

(CATL,SYSLST) Specifies that the summary of catalog entries for private disks is to be output to SYSLST, sorted according to volume serial numbers.

EXTS Specifies that all extents with free, occupied or dead space on the volume designated by "vsn" are to be output to SYSLST, sorted by extents. In particular, information is given on any duplicate allocations (overlays).
This operand can be used for pool volumes only. All volumes entered in the directory of the disk are requested. Execution of the statement is aborted if conflicting directory entries (missing backward references) occur.

POOL Specifies that the directory of the volume "vsn" is to be output to SYSOUT.

CATID=catid	Specifies the catalog to be searched in connection with the CATL option. If the operand is omitted, the user's default catalog ID is assumed.
MVFS	Specifies that all multivolume files in the VTOC of the
(MVFS,SYSOUT)	volume "vsn" (i.e. all files beginning on "vsn") are to be output to SYSOUT.
(MVFS,SYSLST)	Specifies that all multivolume files in the VTOC of the volume "vsn" are to be output to SYSLST.

MODIFY
Modify default parameters

The MODIFY statement can be used to modify various default operands. These include:

– routing code for output via the console (default value: X)
– lines per page on SYSLST logs (default value: 60)
– mode of reservation for disk devices (default value: "exclusive").

If no operands are specified, MODIFY functions as an information statement, displaying information on the current values of the variable parameters.

Operation	Operands
{ M / MODIFY }	[{ RC = r / LINES=nn / EXCL={ Y[ES] / N[O] } }][,HOLD]

RC=r	Sets the routing code to the value "r". Any character may be used for "r". No validity check is made.
LINES=nn	Specifies the number of lines per page for SYSLST logs. "nn" is a decimal number between 5 and 99.
EXCL	
=Y[ES]	Sets the mode of reservation for disk devices to "exclusive", i.e. with CREATE and REMOVE statements attempts to access private volumes of the system are rejected unless the volumes are mounted on shareable private disks (SPD).

=N[O]	Sets the mode of reservation for disk devices to "shareable". Private volumes of the system are locked against access by other jobs during execution of the CREATE and REMOVE statements. Since this modifies running conditions for the other jobs to such an extent as to provoke abnormal task termination, this mode should be selected only when unavoidable.
HOLD	Specifies that the required modification is to be permanent and is to remain valid for all further calls. This operand cannot be used with multiple loading of PDPOOLS. In such a case the program must first be unloaded by all jobs involved.

PATCH
Output information on corrections made in patch form

The PATCH statement is used to output information on corrections made in patch form to the current version of PDPOOLS. The number of existing patches is indicated by "nn" in the following message after the program is loaded:

```
PDPOOLS VERS. vvvvv (xxx / nn)
```

where "vvvvv" is the PDPOOLS version and "xxx" the source version.

Operation	Operands
{ + / PATCH }	

The entry "+" instead of "PATCH" is recognized only in position 1.

PURGE
Release dead space/remove pool directory

The PURGE statement has two distinct functions:

1. It releases dead space (SPACE NOT ASSIGNED) that has appeared on a pool volume as a result of REMOVE.
2. It completely removes the pool directory from a volume.

Both functions may be used only if the disk in question is a pool volume. The directory can be deleted only when it consists of a single entry (SINGLE POOL VOLUME).

Operation	Operands
{ P / PURGE }	{ vsn / (vsn,dc) } [,POOL]

vsn	Specifies the volume for which the PURGE function is to
(vsn,dc)	be performed. For an explanation of "dc", see the CREATE statement.
POOL	Specifies that the directory of the volume "vsn" is to be removed. If this operand is omitted, the dead space resulting from the REMOVE statement is removed from the disk. This involves an implicit request for all volumes entered in the directory (cf. the EXTS operand of the LIST statement).

Note

The PURGE statement in the form PURGE vsn is invalid for the 3410 External High-Speed Storage Unit; however, PURGE vsn,POOL is acceptable.

REMOVE
Remove volume serial numbers from pool

The REMOVE statement can be used to remove individual volume serial numbers from a pool. A search is made of the remaining volumes of the pool to find any multivolume files that continue on the volumes to be removed; if there are any such files, they are deleted. The catalog entries of all files with extents on volumes having these volume serial numbers are normally removed from the system catalog. However, this option can also be suppressed. If a volume contains no directory entries, a dummy directory is generated from the F1 label. In this way it is possible to "clean up" even those pools that were not set up using CREATE. It remains the responsibility of system administration to specify all volume serial numbers with multivolume files that continue on the volume to be removed.

Operation	Operands
{ R | REMOVE }	{ vsn | (vsn,vsn,...) } , POOL={ vsn | (vsn,dc) | (vsn,[dc],vsn,[dc],...) } [,EXPORT={ Y[ES] | N[O] }] [OUTPUT={ NO | ERROR | ALL | (ERROR [,{ SYSOUT | SYSLST }]) | (ALL [,{ SYSOUT | SYSLST }]) }]

vsn (vsn,...)	All volume serial numbers specified here under "vsn" are to be removed from the pool defined under POOL=... .

POOL
=vsn
=(vsn,dc)
=(vsn,[dc],vsn,[dc],...)

This specifies the pool involved. The directory entries are always used in order to determine all volumes of the pool whenever possible. The pool is always complete when no overlapping occurs. For an explanation of "dc", see the CREATE statement.

EXPORT

=YES — All system catalog entries for the volumes to be removed from the pool are deleted (default value).

=NO — The system catalog entries for the volumes to be removed from the pool are not deleted.

OUTPUT

=NO — No check list of the deleted files and catalog entries is output.

=ERROR

=(ERROR,SYSOUT)
A check list of all the errors that occurred during deletion is output to SYSOUT.

=ALL

=(ALL,SYSOUT) — A check list of all files deleted is output to SYSOUT. ALL is the default value.

OUTPUT=(...,SYSLST)

The check list is output to SYSLST.

As with the CREATE statement, the volumes required for execution of the REMOVE statement are reserved exclusively. The procedure is the same as for the CREATE statement.

9.2 PDPOOLS Messages

`EXPORT ERROR dddd`

Meaning
DMS error dddd has occurred whilst performing EXPORT.

`NO DIRECTORY AVAILABLE. USE DIRECTORY=NO`

Meaning
None of the volumes designated by POOL= has a directory.

Response
The CREATE statement must be repeated with the option DIRECTORY=NO.

`OVERFLOW POOL SAVE AREA`

Meaning
The pool designated by POOL= contains more than 32 volumes when a REMOVE or CHECK statement is issued.

Response
The statement must be repeated with a different pool definition.

`POOL TO BE CREATED HAS MORE THAN 32 VOLUMES`

Meaning
Self-explanatory.

Response
The CREATE statement must be repeated with a different pool definition.

`REQUEST MEMORY ERROR xx (CLASS 6)`

Meaning
A request for class 6 memory could not be processed successfully. Error xx has occurred.

`VOLUME vvvvvv CANNOT BE ACQUIRED EXCLUSIVELY`

Meaning
Volume vvvvvv may be accessed by several jobs but is not mounted on a shareable private disk.

Response
The rejected statement (CREATE or REMOVE) may be repeated after changing the reservation mode for the disk drive from "shareable" to "exclusive" with the MODIFY statement.

`VOLUME vvvvvv: CLOSE ERROR dddd`

Meaning
DMS error dddd has occurred during CLOSE.

```
VOLUME vvvvvv: I/O ERROR dddd
```

Meaning
DMS error dddd has occurred during reading/writing.

```
VOLUME vvvvvv NOT IN CVT. USE DEVICE OPTION.
```

Meaning
Volume vvvvvv is not online.

Response
The statement must be repeated with specification of the device type.

```
VOLUME vvvvvv NOT IN POOL. USE DIRECTORY=NO.
```

Meaning
A volume which does not belong to the pool defined has been found in the second level via the directory entries. The pool to be created contains too many individual pools.

Response
The CREATE statement should be repeated with a different pool definition or with the option DIRECTORY=NO.

```
VOLUME vvvvvv NOT IN POOL. USE MODE=MULTIPLE.
```

Meaning
The pool to be created overlaps with another pool.

Response
The pool is set up by repeating the CREATE statement with the option MODE=MULTIPLE.

```
VOLUME vvvvvv: OPEN ERROR dddd
```

Meaning
DMS error dddd has occurred during OPEN.

```
VTOC ENTRY WITH VOLUME vvvvvv EXCEEDS POOL
```

Meaning
A volume of the pool to be created contains no directory entries; however, a multivolume file is cataloged in the VTOC with a pointer to the volume vvvvvv, which is located outside the pool.

Response
The statement should be reformulated using a pool definition which includes volume vvvvvv.

```
LIST ERROR dddd
```

Meaning
A DMS error with the error code dddd has occurred during execution of the LIST statement.

`MODIFY ERROR dddd`

Meaning
A DMS error with the error code dddd has occurred during execution of the LIST statement.

`PURGE ERROR dddd`

Meaning
A DMS error with the error code dddd has occurred during execution of the PURGE statement.

`PURGE REQUEST PARTIALLY SATISFIED`

Meaning
The dead space (SPACE NOT ASSIGNED) which is to be released consists of more than 255 extents. The space belonging to the first 255 extents is removed.

Response
A further PURGE statement must be issued in order to release the remaining space.

`SORT ERROR`

Meaning
An error has occurred in SORT output during execution of the statement LIST, REMOVE or EXPORT, in which the SYSLIST operand was specified.

`VOLUME vvvvvv: INCONSISTENT DIRECTORY ENTRY wwwwww`

Meaning
After entry of the CREATE statement, a directory entry for volume wwwwww was found on volume vvvvvv. However, the check reference to volume vvvvvv is missing on volume wwwwww. Volume vvvvvv cannot be included in the pool since multiple space assignment on volume wwwwww would then be possible.

Note

This message may appear more than once with different wwwwww volume specifications for the same vvvvvv volume. In this case, all wwwwww volumes should be removed from volume vvvvvv by means of a single REMOVE statement: If this is not done, unforeseen results may occur.

Response
Volume vvvvvv should be cleaned up by means of the statement REMOVE wwwwww,POOL=vvvvvv,EXPORT=NO prior to repeating the CREATE statement.

`VOLUME vvvvvv: NO DIRECTORY`

Meaning
A PURGE or LIST statement containing the operand EXTS was issued for a volume without directory entries.

```
VOLUME vvvvvv: NO SINGLE POOL VOLUME
```

Meaning
A PURGE statement containing the POOL operand was issued for volume vvvvvv. However, this volume contains more than one directory entry.

```
DEVICE gggggggg INVALID
```

Meaning
NKGTYPE does not recognize the specified device gggggggg.

```
VOLUME vvvvvv: NO PRIVATE DISK
```

Meaning
The disk with the volume serial number vvvvvv is not a private disk. PDPOOLS operates exclusively with private disks.

9.3 Error codes

Certain DMS error codes are generated by PDPOOLS. Some have a special meaning in this context and are listed in the following.

0222: invalid page request when reading/writing a volume

0228: invalid extent list address when reading/writing a volume

0300: DKCMSE entry not found in EOLDTAB

0301: invalid function for READ-VTOC routine (program error)

0302: SORT error during execution of PURGE statement

0331: save area problem or release memory error when reading VTOCs

0332: the file to be removed from the system catalog is currently being used.

0334: request memory error when reading VTOCs

033D: invalid VTOC size in F5 label

0400: DRM0100 or DRN1110 entry not found in EOLDTAB

0500: EJWDWGNX entry not found in EOLDTAB

05B6: $REQM error during EXPORT initialization

05C5: the EXPORT function should have been performed for a tape volume

05C7: $RELM error during initialization or termination of EXPORT function

05FC: error in reading TSOSJOIN during EXPORT initialization

0900: DQNFILTA entry not found in EOLDTAB

0922: invalid PAM page found when executing MODIFY statement

0932: program is or was multiply loaded, MODIFY statement with HOLD operand is not possible.

099F: request memory error when executing MODIFY statement

0D9F: $REQM error when requesting memory for extent list

Normally, no overlapping pools should be used. However, if they are permitted it should be noted that an unsuitable choice of operands in the CREATE-FILE and MODIFY-FILE-ATTRIBUTES commands may result in the creation of files that cannot be extended beyond a certain point. For this to happen the volume serial numbers specified in the commands must be selected incorrectly, i.e. such that they belong to different pools.

9.4 Examples

Example 1

Assume the following pools exist:

```
Pool1 = vsna, vsnb
Pool2 = vsnc, vsnb
```

If the command:

```
/CREATE FILE  FILE-NAME=filename,
              SUPPORT=*PRIVATE-DISK(VOLUME=vsnb,vsna,vsnc,
              DEVICE-TYPE=devicetype,SPACE=*RELATIVE(PRIMARY-ALLOCATION=3))
```

is given and if it is assumed that at least 3 blocks are free on "vsnb", the file will be created despite the incorrect operand ("vsnc", a volume serial number belonging to the other pool, namely pool 2) having been given in the command. This is because a check is made only as to whether the disks "vsna", "vsnb" and "vsnc" are entered in the SVL of b. When OPEN is issued, the file is accepted since there are extents on only one volume. If, however, the file is to be extended onto volume "vsna", a check is made as to whether "vsna", "vsnb" and vsnc are in the SVL of vsna. This is not the case and the request will be rejected.

Example 2

Assume the following pools exist:

```
Pool1 = vsna, vsnb
Pool2 = vsnc, vsnd
```

If the command:

```
/CREATE-FILE  FILE-NAME=filename,SUPPORT=*PRIVATE-DISK(VOLUME=vsna,
              DEVICE-TYPE=devicetype,SPACE=*RELATIVE(PRIMARY-ALLOCATION=3))
```

is given, followed by the command

```
/MODIFY-FILE-ATTRIBUTES  FILE-NAME=dateiname,
                         SUPPORT=*PRIVATE-DISK(VOLUME=vsnc,
                         SPACE=*RELATIVE(PRIMARY-ALLOCATION=3))
```

the MODIFY-FILE-ATTRIBUTES command is rejected as vsnc belongs to pool 2. This command is intended to extend the file on volume vsnc which, however, was created with the CREATE-FILE command (in this case pool 1).

When files are created or extended with the CREATE-FILE and MODIFY-FILE-ATTRIBUTES command, the following messages may occur:

```
DMS0511E  PRIVATE DISK POOL INCONSISTENCY OR PARAMETER ERROR DETECTED.
          ALLOCATION REJECTED.
```

or

```
DMS051E  PRIVATE DISC POOL INCONSISTENCY OR PARAMETER ERROR DETECTED.
         REQUEST PARTIALLY SATISFIED.
```

In the latter case it is advisable to delete the file.

If an inconsistency is recognized during secondary allocation, no message is issued; instead only the return code

```
441  NO SPACE AVAILABLE
or
44a  REQUEST PARTIALLY SATISFIED
```

is returned to the calling access method. This then activates the NOSPACE exit or abnormal program termination.

10 PVSREN Pubset conversion

Version: **PVSREN V1.1A**

In order to extend the number of pubsets and give a greater choice of pubset names, Version 10.0A introduces a new format called POINT notation in addition to the existing form of VSN addressing (PUB notation) for public volumes. The number of volumes per pubset depends on the value of ALLOCATION-UNIT (AU):

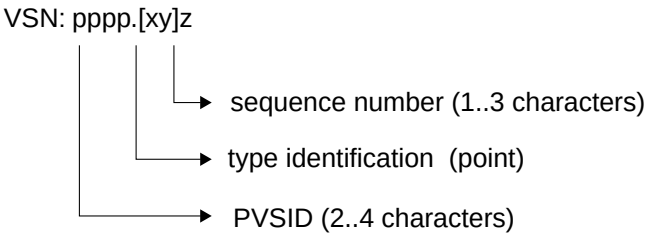
AU = 8 Kbytes or 64 Kbytes and 2- or 3-digit sequence number: max. 255 volumes per pubset;
AU = 8 Kbytes or 64 Kbytes and a 1-digit sequence number: max. 36 volumes per pubset;
AU = 6 Kbytes: max. 32 volumes per pubset.

The PVSREN utility routine (Public Volume Set RENname) enables a pubset to be converted without re-initializing.

PVSREN V1.1A under BS2000/OSD-BC V1.0 enables system administration under TSOS to rename NK2, NK2(8K,64K), NK4 and key pubsets under TSOS.

10.1 Comparison of the two formats of VSN addressing

The POINT notation (’.’ notation) is the format used for extended VSN addressing:



The following features distinguish the new format from the old format:

– The PUB prefix (allowed by the old format) is not permitted. (Example: PUBA.0 and PUB.01 are not permitted by the new format.)

– In contrast to the format for VSN addressing used to date (1-character PVSID), in the new format of POINT notation the PVSID length is variable and can be from 2 to 4 characters.

– The PVSID and the sequence number are separated by a period (.). The period (point) distinguishes the volume from a private disk and is also the type identification (in the old format: PUB).

– The length of the sequence number is variable and can be from 1 to 3 characters (in the old format the sequence number was always 2 characters long). The following values are valid sequence numbers (depending on ALLOCATION-UNIT):
ALLOCATION-UNIT = 8 Kbytes or 64 Kbytes: x, y, z = 0...9,A...Z.
ALLOCATION-UNIT = 6 Kbytes: x, y = 0; z = 0...9, A...V.

– The SYSID is always a value other than PVSID. In the ’.’ notation, SYSID can be a numerical value between 65 and 192; in the old format SYSID was a value between A...Z and 0..9. The system default SYSID is 250.
In the new format, the SYSID can be assigned on initialization and can also be modified by the SET-PUBSET-ATTRIBUTES command. It is also possible for the SYSID not to be present in the SVL. In the old format, however, the SYSID is the same as the PVSID and cannot be specified at generation time, nor modified.

– In shared-pubset mode, note that the SYSIDs must be unique within in the network - particular care is required when the system default is used.
The value can be assigned on initialization and can be modified by means of the SET-PUBSET-ATTRIBUTES command.

Furthermore, the format of the file names on a disk with the new VSN addressing is different from the old format. With both file name formats, the maximum length (54 characters) is the same, but in the new format the catalog ID is 4 to 6 characters long (2 to 4 characters between two colons). This shortens the length of the user ID and of the file name proper. The same is true of file generation groups and job variable entries.

Conversion is supported in the following directions:

– PUB notation to PUB notation (irrespective of ALLOCATION-UNIT 6 Kbytes or 8 Kbytes / 64 Kbytes)

– PUB notation to POINT notation (irrespective of ALLOCATION-UNIT 6 Kbytes or 8 Kbytes / 64 Kbytes)

– POINT notation to PUB notation: renaming is not possible if the number of disks belonging to the pubset is in excess of 32, the maximum permissible number for PUB notation. If the number is ≤ 32, renaming is implemented.

– POINT notation to POINT notation: a pubset may be renamed only if the number of disks in the pubset is not in excess of the value that would be yielded by the PVSID length of the pubset after renaming.

For more details on the new format for VSN addressing, see the manual "Commands, Volumes 1 - 3" [1], [2], [3].

10.2 Prerequisites for running PVSREN

For PVSREN to execute, a syntax file containing a description of the PVSREN syntax must be activated.

BS2000/OSD-BC versions ≥ V1.0:
The SDF syntax file called $TSOS.SYSSDF.PVSREN.011.SYSTEM, which was supplied with the system and which contains the entire functionality of PVSREN, must be incorporated in the existing syntax files by system administration by means of SDF-I. This entails merging the file with the system syntax file.

BS2000/OSD-BC versions < V1.0: use the syntax file $TSOS.SYSSDF.PVSREN.011.TSOS. Merge this file with the group syntax file for the user ID TSOS.

10.3 Operating instructions

Before conversion, a catalog entry must be made for each private file that may be imported either immediately or in the future. This ensures that all files on these private disks have an entry in the system catalog (TSOSCAT), and means that the length of the file names can also be checked.

It must also be ensured that no MRSCAT entry already exists with the catalog ID that the pubset is to be assigned, otherwise no change is made to the JOIN file.

A further point to note is that print jobs for files residing on the pubset to be converted cannot be performed following the conversion. The jobs remain in the queue because they are waiting for the corresponding pubset with the catalog ID specified in the jobs to be mounted. However, this catalog ID has already been changed in the current system catalog and there is no longer any means of accessing the old catalog.

The pubset should be exported by means of the EXPORT-PUBSET command and set to "inaccessible" to exclude any possibility of unwanted access to the catalog during the conversion.

If the conversion is being run from another system, all the disks belonging to the pubset must be switched to the system executing the conversion.

A home pubset is always required when a pubset is being converted. If a home pubset is being renamed, a second home pubset, e.g. a duplicate of the one to be renamed, must be available.

Before calling PVSREN, check that there are no ENTER jobs in the job pool, because jobs left in the pool may not be executable after renaming.

```
The PVSREN utility is called under the TSOS user ID with the following
command:

 START-PROGRAM FROM-FILE=PVSREN
```

The following messages are output:

```
BLS0500 PROGRAM 'PVSREN', VERSION '01.1A' OF '1992-10-07' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1992. ALL
RIGHTS RESERVED
%  PVR0000 *****   P V S R E N    VERSION '01.1A80'   *****
//
```

The output medium and scope of the log must be selected.

The `CPU-LIMIT`, `TEST-OPTIONS`, `MONJV`, `RESIDENT-PAGES` and `VIRTUAL-PAGES` operands of the `START-PROGRAM` command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the `START-PROGRAM` command in the manual "Commands, Volume 3" [3].

If the conversion will increase the length of the catalog ID, it is advisable first to run the CHECK-FILENAME-LENGTH statement. This will list all file and job variable names whose length exceeds the maximum. This list must then be used to modify the file names that are too long.

Once these steps have been performed, the CONVERT-PUBSET statement can be entered. First of all, the program checks the names for length even if the CHECK-FILENAME-LENGTH function was executed beforehand. This is because there is a possibility of other users creating problems while PVSREN is renaming the object. The implicit check of object name length also provides an opportunity of checking the pubset for migrated files. If any such files are found, conversion is not implemented and the user is informed to this effect by a negative acknowledgment.
Backup files created with ARCHIVE are treated in the same way.

After successful conversion of the file catalog TSOSCAT and the VOL1 labels, it is also possible on request to:

– modify the default catalog ID (catid-old → catid-new) in the TSOSJOIN of the home pubset and/or
– in the TSOSJOIN of the converted pubset,
– import the pubset into the system.

If the JOIN file cannot be modified because the multi-positional catalog ID means that the pubset cannot be imported into this system (prior to V10) it is possible to use a system ≥ V10 to convert the default catalog ID in the JOIN file subsequently (MODIFY-JOINFILE statement).

If the modified pubset is one that will later be used as a home pubset, the following points must be borne in mind:

In many computer centers, batch tasks in which this catalog ID is defined are held in an ENTER file and started automatically at startup (these may include, for example, ENTER to load SPEEDCAT). This catalog ID is now no longer valid and must be replaced by the new one. A fast startup is executed using a parameter file in which references to the VSN must also be changed.

Note

As of BS2000/OSD-BC V1.0, PVSREN V1.1A supports both the old NK2 format and the new NK4 format of TSOSCAT.

The PVSREN routine is controlled by statements, which are described below.

10.4 Statements

10.4.1 Overview of the PVSREN statements

Statement	Meaning
CHECK-FILENAME-LENGTH	Check the length of file and job variable names in a pubset catalog
CONVERT-PUBSET	Convert (rename) a pubset
END	Terminate the program
HOLD-PROGRAM	Interrupt the program for command input
MODIFY-JOINFILE	Modify the default catalog ID in the pubset JOIN entry
MODIFY-LOGGING-OPTIONS	Modify the current default logging option values for all program functions
SHOW-LOGGING-OPTIONS	List the current logging option values

Other permissible statements are:

```
//MODIFY-SDF-OPTIONS,
//SHOW-SDF-OPTIONS
//EXECUTE-SYSTEM-CMD
```

These are mentioned here only briefly, as they are used in conjunction with SDF and are described in detail in the manual "Introductory Guide to the SDF Dialog Interface" [13].

10.4.2 Description of the statements

CHECK-FILENAME-LENGTH
Check length of file and job variable names

The CHECK-FILENAME-LENGTH statement checks that the length of all file and job variable names in the pubset catalog does not exceed the maximum.

This function is only useful in performing a conversion in which the catalog ID length increases. The statement is only executed for a pubset that is set to "inaccessible" for an EXPORT-PUBSET command.

If the program finds names whose length exceeds the maximum length, these names are listed and PVSREN moves to the next //STEP statement. These file or job variable names must then be changed to the correct length by means of a MODIFY-FILE-ATTRIBUTES or MODIFY-JV-ATTRIBUTES command.

The maximum length of a path name (with catalog ID and user ID) is 54 characters for files and 47 characters for file generation groups.
The maximum length of the file name as such (without catalog ID and user ID) is calculated as follows:

for files:
54 minus 4 minus length of catalog ID minus length of user ID

for file generation groups:
47 minus 4 minus length of catalog ID minus length of user ID

The length check, where required, should always be performed before the conversion.

The program also checks whether the SYSID is initialized in the SVL of the PUBRES. If not, an appropriate message is output.

CHECK-FILENAME-LENGTH
CATID = <alphanum-name 1..3> ,NEW-CATID = <alphanum-name 2..4>

CATID = <alphanum-name 1..3>
The current catalog ID.

NEW-CATID = <alphanum-name 2..4>
The required catalog ID.

Example

- Input

  CATID = A
  NEW-CATID = ABC

  The new catalog ID has 3 characters, giving a maximum length as follows:

  for file names:
  39 characters
  (CATID: 3 characters + 2 colons = 5 characters,
  $userid = max. 10 characters,
  file name: 54 characters minus 15 characters = 39 characters)

  for file generation groups:
  32 characters
  (CATID: 3 characters + 2 colons = 5 characters,
  $userid = max. 10 characters,
  *index = 7 characters,
  file name: 54 characters minus 22 characters = 32 characters)

- Output:

  When names are found that exceed the maximum length, the following message is output:

Following object names exceed maximum length of 39 or 32 characters:

```
 :<catid>:<userid>.<jvname>    (JV) ———————————————————————————————— (1)
*:<catid>:<userid>.<filename> (FGG) ———————————————————————————————— (2)
 :<catid>:<userid>.<filename> (FGG) ———————————————————————————————— (3)
*:<catid>:<userid>.<filename> ———————————————————————————————————— (4)
 :<catid>:<userid>.<filename> ———————————————————————————————————— (5)
```

(1) Is a job variable

(2) Is a private file generation group

(3) Is a public file generation group

(4) Is a private file

(5) Is a public file

CONVERT-PUBSET
Convert pubset

The CONVERT-PUBSET function is executed only for pubsets that are set to "inaccessible" (not imported).

This statement provides the following functions:

1. Checking the length of file names and names of file generation groups; checking for HSMS files (migration files) and backup files (created with ARCHIVE).
2. Changing the catalog ID of the pubset
   a) Replacing the old catalog ID by a new one for file and file generation group names in the TSOSCAT catalog.
   b) Converting the VOL1 labels of all public volumes belonging to the pubset.
   c) Modifying on request the JOIN files
      – of the home pubset
      – of the modified pubset

      using the new catalog ID as the default value for all user IDs.
3. Updating the names of the paging areas.

The following conversion directions are supported:

– from PUB notation to PUB notation
– from PUB notation to POINT notation
– from POINT notation to PUB notation
– from POINT notation to POINT notation.

Checking the length of object names (files, file generation groups)

There are two methods of checking the lengths of object names.

The first method is to use the CHECK-FILENAME-LENGTH statement (see page 407).

The second method is to use the CONVERT-PUBSET statement. This check is always run before the actual conversion takes place, in case the conversion increases the length of CATID. The check is carried out implicitly even if a CHECK-FILENAME-LENGTH statement was previously used to perform a length check.

This is done for security reasons, because further problems may occur when the CHECK-FILENAME-LENGTH statement finds names that exceed the maximum length, and modifies them. If the CONVERT-PUBSET statement finds names that exceed the maximum length, the conversion cannot be performed.

The check for HSMS files and ARCHIVE backup files is performed before conversion, namely during the implicit length check of the object names in the CONVERT-PUBSET statement.
If the pubset includes catalog entries for migrated files, conversion does not take place. If this happens, a message is output to inform the user in order to avoid inconsistencies. The same procedure is applied to backup files created with ARCHIVE.

Changing the catalog ID of the pubset

The pubset catalog ID is modified in three stages:

1. The old catalog IDs are replaced by the new ones in the names of files and file generation groups in the TSOSCAT catalog.
   The VSNs are modified in the catalog entries of public files and file generation groups in the volume table.

   Note

   As of BS2000/OSD-BC V1.0, PVSREN V1.1A supports both the old NK2 and the new NK4 format of TSOSCAT.
   If a PROFIL entry (possible as of BS2000/OSD-BC V1.0) exists, the catalog ID in this entry, too, is renamed.

2. Conversion of VOL1 labels (SVL).
   The VSN is changed in the SVL of all volumes belonging to the pubset. The SYSID and the OLC in the SVL of the PUBRES are also changed.

   Before the conversion, the first entry (VSN entry for the PUBRES) in the OLC (=online-catalog, list of disks belonging to the pubset) is set to spaces and replaced by the current VSN only after the conversion. This means that after a system crash, for example (INCONSISTENCY), the pubset cannot be imported.

3. Changing the JOIN files on request.
   The system administrator receives a message asking whether the new catalog ID should be used instead of the old one as the default catalog ID in the TSOSJOIN of the home pubset and the modified pubset. The pubset is imported from the program for this purpose. Sometimes, execution of the implicit IMPORT-PUBSET command can last a considerable time. If the pubset has not been imported after 10 minutes, PVSREN aborts. If the pubset is imported before the 10 minutes expire, a message appears asking whether the imported pubset is to be "accessible" for this session, i.e. is to remain imported. If this is not required and the pubset had an entry in the home pubset's MRSCAT, this entry is deleted and replaced by the entry with the new catalog ID.

For BS2000 versions < V10.0 and the conversion directions

– PUB notation → POINT notation
– POINT notation → POINT notation
– POINT notation → PUB notation

the JOIN file cannot be changed, because these versions do not accept a multicharacter catalog ID.

Note

As of BS2000 V10.0, user information is stored in the file $TSOS.SYSSRPM and not only in $TSOS.TSOSJOIN. The user file is processed only logically, not physically, by JOINFOAI or by the JOIN command.

Warning

Any existing MRSCAT entry with the modified catalog ID prevents the implicit importation of the pubset.

```
CONVERT-PUBSET

 PUBSET = *PUB-NOTATION(...) / *POINT-NOTATION(...)
    *PUB-NOTATION(...)
       │ CATID = <alphanum-name 1..1>
    *POINT-NOTATION(...)
       │ CATID = <alphanum-name 2..4>
,NEW-NAME = *PUB-NOTATION(...) / *POINT-NOTATION(...)
    *PUB-NOTATION(...)
       │ CATID = <alphanum-name 1..1>
    *POINT-NOTATION(...)
       │ CATID = <alphanum-name 2..4>
       │ ,SYSID = <integer 65..192> / *SAME
```

PUBSET =
Gives the VSN format (notation) and catalog ID of the pubset to be converted.

PUBSET = *PUB-NOTATION(...)
The pubset to be modified has the old VSN format (PUB notation). The direction of the conversion is PUB notation to PUB notation or to POINT notation.

CATID =
Gives the catalog ID of the pubset to be converted.

CATID = <alphanum-name 1>
The catalog ID of the pubset to be converted consists of a single alphanumeric character.

PUBSET = *POINT-NOTATION(...)
The pubset to be converted has the new VSN format (POINT notation). The direction of the conversion is POINT notation to PUB notation or to POINT notation.

CATID =
Gives the catalog ID of the pubset to be converted.

CATID = <alphanum-name 2..4>
The catalog ID of the pubset to be converted consists of 2 to 4 alphanumeric characters.

NEW-NAME =
Specifies the VSN format (target notation) and catalog ID to which the pubset is to be converted.

NEW-NAME = *PUB-NOTATION(...)
The pubset is to be converted from the VSN format specified in the PUBSET operand to PUB notation format.

CATID =
Gives the new catalog ID the pubset is to receive on conversion.

CATID = <alphanum-name 1>
The new catalog ID consists of 1 alphanumeric character.

NEW-NAME = *POINT-NOTATION(...)
The pubset is to be converted from the VSN format specified in the PUBSET operand to POINT notation format.

CATID =
Specifies the new catalog ID the pubset is to receive on conversion.

CATID = <alphanum-name 2..4>
The new catalog ID consists of 2 to 4 alphanumeric characters.

SYSID =
The SYSID to be entered in the pubset's SVL.

SYSID = <integer 65..192>
The values permitted for SYSID are in the range 65 to 192.

SYSID = *SAME
If the SYSID is to remain unchanged, *SAME must be specified.
If there is no SYSID in the SVL, an appropriate message is output. In this case system administration must allocate a SYSID retroactively. If the pubset exists in PUB notation, however, *SAME is rejected.

Note

If semantic errors are found in the statement, they can be corrected interactively.
On completion, the successful/unsuccessful conversion is listed in accordance with the MODIFY-LOGGING-OPTIONS statement, to include all associated VSNs, file names and job variable names.

END
Terminate program

The END statement causes PVSREN to be terminated.

END

HOLD-PROGRAM
Switch from program mode to system mode

The HOLD-PROGRAM statement interrupts program mode and switches to system mode. In system mode, the user can enter commands between the statements. The RESUME command is used to return to program mode.

HOLD-PROGRAM

MODIFY-JOINFILE
Modify default catalog ID in JOIN entry

The MODIFY-JOINFILE statement can be used to change the default catalog ID for all users in a pubset's user catalog. There is one exception: under the TSOS user ID, a prerequisite for conversion is that the values for the PUBSET and NEW-DEFAULT-CATID operands are the same.
In order to execute this statement, the pubset must be imported.

Where a pubset has been converted to POINT notation in a system <V10.0, the JOIN file cannot be converted because multicharacter catalog IDs are not permitted in such systems. This statement is important in cases of this type, as it allows the default catalog ID to be modified after execution of a CONVERT-PUBSET statement.

```
MODIFY-JOINFILE

 PUBSET = <alphanum-name 1..4>
,DEFAULT-CATID = <alphanum-name 1..4>
,NEW-DEFAULT-CATID = <alphanum-name 1..4>
```

PUBSET = <alphanum-name 1..4>
Specifies the catalog ID of the pubset in which the JOIN file in question resides.

DEFAULT-CATID = <alphanum-name 1..4>
Specifies the default catalog ID in the JOIN file to be modified.

NEW-DEFAULT-CATID = <alphanum-name 1..4>
Specifies the catalog ID that is to replace the old one and become the new default catalog ID.

MODIFY-LOGGING-OPTIONS
Modify default logging option values

The MODIFY-LOGGING-OPTIONS statement enables default logging values for PVSREN to be modified globally for all program functions.

If the statement is entered without operands, the default values remain unchanged. The user can request the current values via the SHOW-LOGGING-OPTIONS statement.

MODIFY-LOGGING-OPTIONS
INFORMATION = UNCHANGED / *MEDIUM / *MINIMUM / *MAXIMUM ,OUTPUT = UNCHANGED / list-poss(2): *SYSOUT / *SYSLST

INFORMATION =
Gives the scope of the log generated by PVSREN.

INFORMATION = UNCHANGED
The default logging option value remains unchanged. The default value is INFORMATION = *MEDIUM.

INFORMATION = *MEDIUM
Positive acknowledgments of special significance to the user are to be logged in addition to error messages.

INFORMATION = *MINIMUM
Only error messages are to be logged.

INFORMATION = *MAXIMUM
All PVSREN messages are to be logged.

OUTPUT =
Defines the output medium for the PVSREN log.

OUTPUT = UNCHANGED
The default value for the output medium is to remain unchanged. The default value is OUTPUT = *SYSOUT.

OUTPUT = *SYSOUT
In interactive mode the output is to go to the terminal and in batch mode to the system file SYSOUT.

OUTPUT = *SYSLST
The output medium for logging is the system file SYSLST.
If SYSOUT and SYSLST are specified as lists, the logs are output to the system files SYSLST and SYSOUT (or to the terminal in interactive mode).

No message is output when the OUTPUT = function is executed.

SHOW-LOGGING-OPTIONS
List current logging option values

The SHOW-LOGGING-OPTIONS statement requests a list of the current logging option values.
The output medium is specified by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement (see previous statement).

SHOW-LOGGING-OPTIONS

The values that have been set are listed as follows:

```
%CURRENT LOGGING OPTIONS:
%  INFORMATION:<value>
%  OUTPUT:<value>
```

10.5 Messages

Like all system messages, the PVSREN utility routine messages are 7 characters long; they start with the string PVR.

The output destination is defined by means of the MODIFY-LOGGING-OPTIONS statement. They are output to the terminal by default. The individual messages are described in detail in the manual "System Messages, Volume 2" [17].

11 RFUPD System corrections management

Version: **RFUPD V11.0A**

BS2000/OSD-BC V2.0 is the last version to support RFUPD; in the next version, it will be replaced by RMS (REP mounting system).

RFUPD (REP file update) is a class 2 program that processes REP files in BS2000, generating as output REP files (loaders) on tape and/or disk. The input to RFUPD may consist of up to 16 loaders from disk, 1 loader from tape and 1 loader from SYSDTA.

The loaders are read in the following sequence:

1. Loader from SYSDTA or REP records input from the terminal
2. Loaders from disk (in the sequence LDR01-LDR16; the REP records can be 1-256 bytes long, excluding the record length field)
3. Loader from tape (tapes with and without labels and automatic tape rewind are supported).

They are then merged to form one output loader on disk and/or tape. (For the structure of a loader, see "Structure of a REP file" below.)

The output loader is checked in its entirety for format and logical errors. Format checking covers the syntax of all valid input records (REPs, modules, comments). Logical checking is conducted on the basis of the assigned object module library.

RFUPD can be run both as a batch job and in interactive mode.

From statements, REP records and earlier REP files (from disk or tape) the RFUPD routine creates or updates a file for system corrections (REP file or loader), and writes it to disk or tape.

The log of the RFUPD run is output on the printer.

At system startup the Control System is updated for the duration of the session by means of a correction file.

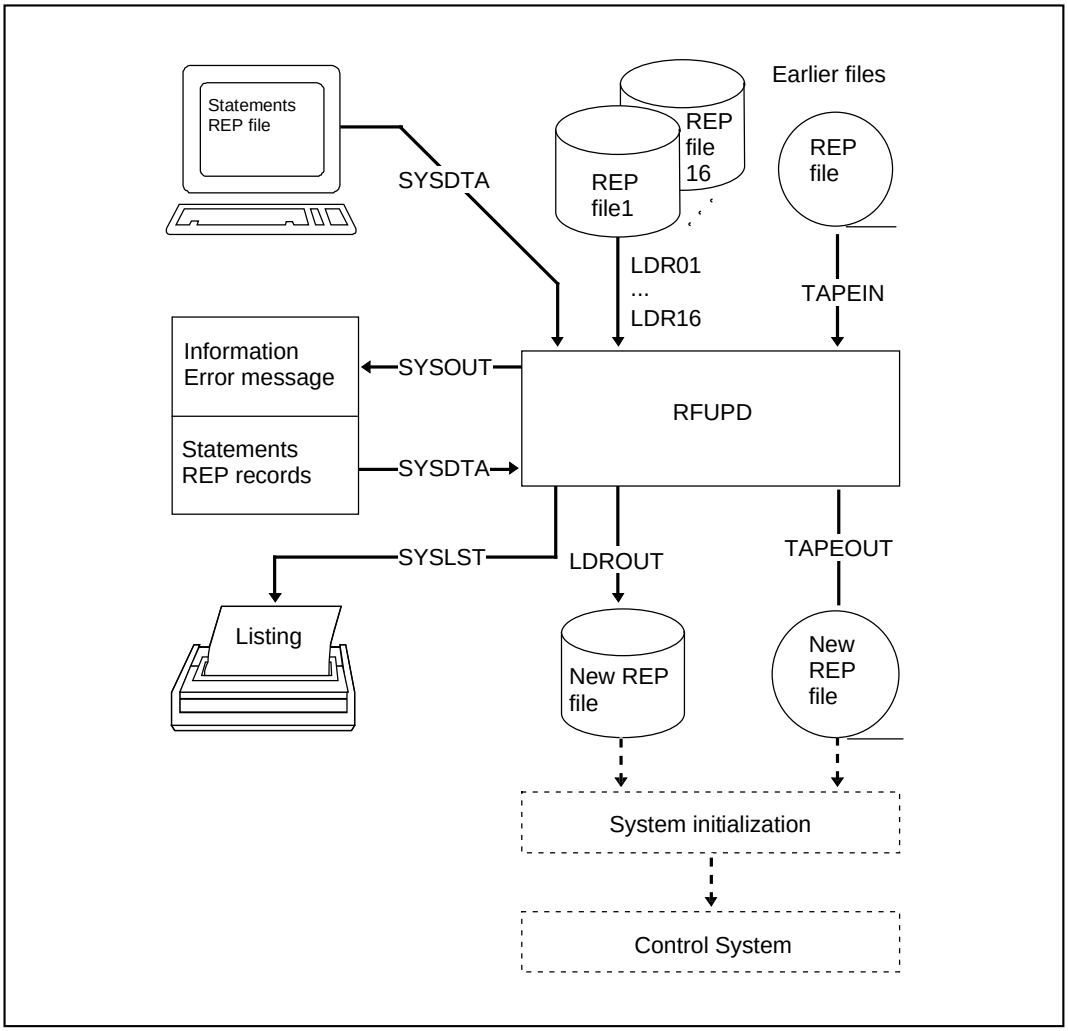


Figure 14: Creating or updating a REP file

Note

RFUPD operates with three work files:

File 1: SORTWORK.Dyyxxx.TS####

yy = Year

xxx = Current day of the year

= Task sequence number (TSN) for this job

SORTWORK is set up by the SORT routine.

File 2: RF.xxxhhmmss.SORTOUT

xxx = Current day of the year

hhmmss = Time of generation

hh = Hour

mm = Minute

ss = Second

The file is set up by RFUPD. It contains all the REP records.

File 3: RF.xxxhhmmss.MODULEFILE

xxx = Current day of the year

hhmmss = Time of generation

hh = Hout

mm = Minute

ss = Second

The file is set up by RFUPD.
It contains all the module records in the order in which they were input.

The files are erased by RFUPD at the end of the run, unless a program error occurs, in which case they must be erased by the user.

11.1 Starting the program run

The RFUPD routine is started with the following command:

```
/START-PROGRAM RFUPD
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in the "Commands, Volume 3" manual [3].

11.2 Input

1. Statements from SYSDTA.
2. Loader or individual REP records from SYSDTA.
3. Up to 16 loaders from disk (REP files), which are assigned by means of the ADD-FILE-LINK command and link names (LDR01-LDR16):

   ```
   /ADD-FILE-LINK LINK-NAME=LDR01,FILE-NAME=filename1,ACCESS-METHOD=SAM
   /ADD-FILE-LINK LINK-NAME=LDR02,FILE-NAME=filename2,ACCESS-METHOD=SAM
   ```

   The REP files must be SAM files with variable record format, but a uniform record length of 80 bytes. Records that are too short can be padded to 80 bytes by means of automatic record correction. Records that are too long are truncated.
   The REP files can be created, for example, by means of /DATA.../END.
4. A loader from a magnetic tape.

   The loader must be assigned by means of an ADD-FILE-LINK command (LINK-NAME=TAPEIN):

   ```
   /IMPORT-FILE    SUPPORT=*TAPE(DEVICE-TYPE=xxxxx)
   /ADD-FILE-LINK  LINK-NAME=TAPEIN,FILE-NAME=filename
                   ACCESS-METHOD=*BTAM,RECORD-FORMAT=*FIXED,(RECORD-SIZE=80),
                   BUFFER-LENGTH=80
   ```

Certain records are ignored in the files serving as input, namely the first record of a loader ("BS2000 LOADER"), trace records, END records for class 1 and class 2 REPs, and the terminating slash character. Consequently, there is no need for these records to be present in the input files. Except for trace records, all these records are reconstructed by RFUPD during generation.
An exception is made only if a completeness check was requested in the HDR statement. In this case a check is made to see whether all loader records are present and are in the correct place.

Note

At the end of its run the RFUPD utility implicitly releases (via a REL macro) the devices assigned for TAPEIN/TAPEOUT. If desired, this can be prevented by means of a LOCK-FILE-LINK command.
The LOCK-FILE-LINK command should be issued after the ADD-FILE-LINK command and before RFUPD is called. In this way the device assignments are retained even though RFUPD has terminated. Subsequently, the devices have to be released explicitly by means of the UNLOCK-FILE-LINK command once they are no longer required.

A loader generated by RFUPD contains 80-byte records only. Any shorter input records are automatically padded with blanks to 80 bytes and any records that are too long are truncated.
Consequently, "FORMAT ERROR" can occur on system loading only if the loader was modified, for example using EDT.

Example

```
/ADD-FILE-LINK LINK-NAME=TAPEIN,FILE-NAME=filename
/LOCK-FILE-LINK LINK-NAME=TAPEIN
/START-PROGRAM FROM-FILE=RFUPD
   .
   .
 *HALT
/UNLOCK-FILE-LINK LINK-NAME=TAPEIN
```

11.3 Output

11.3.1 Files

1. Output loader on disk

   In order to generate an output loader on disk and optionally print it out subsequently, an ADD-FILE-LINK command with the link name LDROUT must first be issued:

   ```
   /ADD-FILE-LINK LINK-NAME=LDROUT,FILE-NAME=filename,ACCESS-METHOD=*SAM
   ```

   (File name of the disk loader to be generated.)

2. Output loader on tape

   To generate an output loader on tape requires a CREATE-FILE command followed by an ADD-FILE-LINK command with the link name TAPEOUT:

   ```
   /CREATE-FILE FILE-NAME=filename,SUPPORT=*TAPE(VOLUME=vsn,DEVICE-TYPE=xxxx)
   /ADD-FILE-LINK  LINK-NAME=TAPEOUT,FILE-NAME=filename
                   ACCESS-METHOD=*BTAM,RECORD-FORMAT=*FIXED,RECORD-SIZE=80,
                   BUFFER-LENGTH=80
   ```

A loader can be created on disk and/or tape, as desired.

If neither of the two above-mentioned link names was specified correctly, no output loader is generated but all the runtime listings are produced.

The REP records and modules are included in the output loader in the order of their input, sorted according to class 1 REPs, modules and class 2 REPs.

Comments are likewise included in the output loader. To ensure that the comments are correctly sorted with the corresponding REP records and modules, they should have a character in column 72 identifying them by class of REP record (1 or 2) or module (M).

REP records and modules which are already flagged in the loader listing as existing in duplicate, or which are to be deleted, are not included in the output loader.

The text of the comment should not go beyond column 71 so as to avoid the possibility of one of the above characters wrongly occurring in column 72.

The REP records "BS2000 LOADER", "END" (for class 1 and class 2 REPs) and "/" are automatically generated for the output loader unless the statement HDR GEN=N was specified. The date and time of creation, the loader code and version are entered in the REP record "BS2000 LOADER".

The loader code is generated as a function of all the REP records and serves to identify the loader. The loader code is computed from the sum of the REP addresses and check numbers of all REP records in the output loader.

The REP records contain the loader version (a letter) in column 71. RFUPD finds the highest loader version and enters the corresponding letter in the REP record "BS2000 LOADER".

11.3.2 Listings

1. Option list

   All statements and commands input from SYSDTA are logged. Default values are also logged, if present. (See the example in "Example of RFUPD output" below.)

2. Loader listing

   This is a sorted listing of all the input files.

   Only REP records and modules are included in the listing. The files are sorted according to module name, module class and address. Errors and the input file are output with each REP record or module.
   (See the example in "Example of RFUPD output" below.)

3. Statistics listing

   A listing containing statistical information on the generated loader is printed.

   The number and type of errors that occurred are output. The number of class 1 and class 2 REP records and the number of modules are also output.

   The computed loader code is printed at the end of this listing (see the example in "Example of RFUPD output" below.)

4. Listing of the generated loader on disk

   (See the example in "Example of RFUPD output" below.)

5. Compare listing

   A compare listing is output if the COMPARE statement was entered (see statements). The listing includes all the differences between the input loaders to be compared. (See the example in "Example of RFUPD output below".)

11.3.3 Check functions

All REP records are checked for format errors and logical errors. Errored REP records are flagged by error codes in the output listing (loader listing). These appear on the left alongside the REP records or modules. If more than 10 error codes are set in a single REP record, only the first 10 errors are indicated, followed by an asterisk (*). Error statistics are also output along with the loader listing.

Standard check functions

The syntax of all REP records is checked.

1. Format check
   a) Record length: 80 bytes
   b) Format of the address field
   c) Date: Is the date present? If so, a check is made that 001 ≤ date ≤ 366
   d) Format of the REP information
   e) Version number: 000 ≤ ver ≤ 999
2. Overlay check
   a) Parity byte
      If the parity byte is missing, one is generated. If it is present, it is checked for errors.
   b) Module class
      Class 1 or class 2.
   c) Module name
      Does the module name conform to conventions?
   d) Are the module records ESD, ISD, RLD, TXT or END records?
   e) Is the END record present in the module or not?
   f) Module with or without ETPND?
   g) Do REP records overlap each other in the loader?
   h) Do REP records overlap a module in the loader?
      If so, a logical check is performed on the REP record vis-a-vis this module (see the following section, item 3),
   i) Are there duplicate REP records or modules in the loader?
   j) Sequence check for module records.
   k) Generation of the sequence number, if not present.

11.4 Statements

11.4.1 Overview of the RFUPD statements

Statement	Meaning
DELETE	Delete records in the output file
DISPLAY	Specify REP and comment records
COMPARE	Compare input files
HDR	Control generation of loader records
HALT	Terminate input from SYSDTA
HELP	Output format of a REP record to SYSOUT

11.4.2 Descriptions of the statements

All statements are entered via SYSDTA. With the exception of the DISPLAY statement, all statements may be used in both interactive and batch mode. In addition to the statements, REP records may also be entered via SYSDTA, in which case, however, the statements must always precede all REP records. The only exceptions to this rule are the HELP and HALT statements, since these support and terminate the REP input, respectively.

DELETE
Delete records in output file

This statement enables one or more records in the output file to be deleted. Deletion is performed if the conditions contained in the DEL statement are satisfied.

Operation	Operands
{ DELETE / DEL }	[type],[name],[add1],[add2],[field],[source]

type
- R for REP
- M for module
- C for comment ("*" in the first byte);
- A for ALL (i.e. R, M and C);

Default value: A.

name
Module name.
Default value: all names (i.e. no names check).

addr1
Address at which deletion of the REP records of a module is to start.

Specification: hexadecimal, up to 5 bytes.
Leading zeros may be omitted,
e.g.: X'00123'=X'123'.
Default value: X'0'

addr2
Address at which deletion is to end. Specification: as for "addr1";
addr2 ≥ addr1
Default value: X'FFFFF'.

field
A string of up to 8 bytes may be specified, with a column number ("col,string").

(col,string)

col $1 \leq col \leq 80$

If the specified character string starts at this column, the REP record is deleted.

string String of up to 8 characters, e.g.: C'ABC'.

Default value: all character strings (i.e. no check on character strings).

source — 1 ≤ source ≤ 17
Specifies the input file from which deletions are to be made. The input file itself is not modified.

1 - 16 = LDR01 - LDR16
17 = tape loader

Default value: all input files.

Notes

– All operands are optional.
– The default values apply to all records.
– Unnecessary operands at the end may be omitted. Missing operands in the middle must be replaced by commas.
– For modules, only the operands "type", "name", "field" and "source" may be specified.
– For comments, only the operands "type", "field" and "source" may be specified.
– If a column number and character string are specified under "field", they relate to the module record generated for this module (for format see "Format of module records").
– Syntactically correct DEL statements whose conditions (operands) are not satisfied are marked in the option listing with an asterisk (*). If all conditions are satisfied, the record is deleted (for an example, see "Example of RFUPD output" below). An asterisk also appears when the record involved has already been deleted by means of a previous DELETE statement.
– A maximum of 500 DELETE statements can be processed. After this, all further DELETE statements are ignored.

Examples

```
DELETE R,BOTS61#X,X'1024',X'2400',(54,C'9004'),4
DEL R,DJPGER,,,,3
DEL R,,X'01251',X'02222',(20,C'10')
DEL M,EVSTART
DEL A,,,,(19,C'11111'),1
DEL C,,,,(2,C'0')
```

DISPLAY
Specify REP and comment records

The DISPLAY statement can be used only in interactive mode. It enables the user to specify certain REP records and comment records. When HALT is entered, these records are displayed individually on the screen in the order in which they were input.
The user can then decide whether the record is to be modified, deleted or left unchanged. Subsequently, REP records or comments may be inserted.

Operation	Operands
{ DISPLAY DIS }	[type],[name],[add1],[add2],[field],[source]

type

R	for REP;
C	for comment ("*" in the first byte);
A	for ALL (i.e. R and C);

Default value: A.

name

Module name.
Default value: all names (i.e. no names check).

addr1

Address at which display of the REP records of a module is to start.
Specification: hexadecimal, up to 5 bytes. Leading zeros may be omitted, e.g.: X'00123'=X'123'.
Default value: X'0'

addr2

Address at which display of the REP records is to end.
Specification: as for "addr1"; addr2 ≥ addr1.
Default value: X'FFFF'.

field

A string of up to 8 bytes may be specified, with a column number ("col,string").

(col,string)

col $1 \leq col \leq 80$

If the specified character string starts in this column, the REP record is displayed on the screen.

string String of up to 8 characters, e.g. C'ABC'.

Default value: all character strings (i.e. no check on character strings).

source — 1 ≤ source ≤ 17
Specifies the input file from which the REP records are to be displayed, where:

1 - 16 = LDR01 - LDR16
17 = tape loader

Notes

– All operands are optional. The default values apply to all REP records.

– Unnecessary operands at the end may be omitted; missing operands in the middle must be replaced by commas.

– For comments, only the operands "type", "field" and "source" may be specified.

– Syntactically correct DIS statements whose conditions (operands) are not satisfied are marked in the option listing with an asterisk (*). If all conditions are satisfied, after a HALT statement the first record specified with DISPLAY is displayed at the terminal. An asterisk also appears when the record involved has already been displayed by means of an earlier DIS statement.

– A maximum of 500 DISPLAY statements are processed. After this, all further DISPLAY statements are ignored.

Examples

```
DISPLAY R,BOTS61#X,X'1204',X'2400',(54,C'9004'),4
DIS R,DJPGER,,,3
DIS R,,X'01251',X'02222',(20,C'10')
DIS A,,,,(19,C'11111'),1
DIS C,,,,(2,C'0')
```

After a REP or comment record specified by DISPLAY has been displayed on the screen, RFUPD expects an immediate response from the user. No explicit request to this effect is made.

The following inputs are possible:

REP...	80-byte REP record. This replaces the REP record previously displayed.
*...	A comment record with * in the first byte. This record overwrites the displayed record.
DEL	The record displayed is deleted.
CONT	The record displayed remains unchanged.
HELP	The format of a REP record is displayed.

NEXT The insert mode is terminated.
The next record specified by DISPLAY will appear on the screen.

HALT The entire DISPLAY function is deactivated.

After any of these inputs (except for NEXT or HALT), further REP or comment records may be inserted. This possibility is indicated to the user by an asterisk (*).

The inserted REP records are first subjected to a format check. In the event of errors the appropriate error messages are displayed on the screen. The corrected REP records can then be reentered.

COMPARE
Compare input files

This statement enables two input files to be compared.

Operation	Operands
{COMPARE / COMP}	x,y[,SIGN]

$1 \leq x \leq 17$ These operands specify which input files are to be compared. The 80-byte records of the files are compared, with the exception of the contents of column 72.

1 - 16 for LDR01 - LDR16

17 for tape loader

SIGN Only certain fields of the REP records are to be compared (REP address, correction data, data check bytes, version, class, module name).
If this operand is omitted, the 80-byte records are compared over their full length, with the exception of column 72.

Note

If several COMPARE statements are entered, only the last one is executed.

HDR
Control generation of loader records

This statement is used to control the generation of loader records. A completeness check may also be requested.

Operation	Operands
HDR	[CHECK={N / Y},][GEN={Y / N}]

CHECK

=N — No completeness check is to be performed (default value).

=Y — A completeness check is to be performed for all input loaders.

GEN

=Y — Requests generation of the REP records "BS2000 LOADER" and "END", as well as the slash (/) at the end of the loader (default value).

=N — Generation of the REP records "BS2000 LOADER" and "END" and the slash at the end of the loader is suppressed.

HALT
Terminate input from SYSDTA

The HALT statement terminates input from SYSDTA and starts the processing of the statements and REP records that have been specified. This statement is necessary only in interactive mode.

Operation	Operands
HALT	[LIST=$\left\{\begin{matrix}\underline{Y}\\N\end{matrix}\right\}$,][PRINT=$\left\{\begin{matrix}\underline{Y}\\N\end{matrix}\right\}$]

LIST

=<u>Y</u>	The sorted loader listing (REP file listing) is output (default value).
=N	Output of the sorted loader listing is suppressed.

PRINT

=<u>Y</u>	The printer listing of the generated output loader is output (default value).
=N	The printer listing is not output.

HELP
Output format of REP record to SYSOUT

The HELP statement causes the REP record format to be output to SYSOUT:

```
REP ADDRX DAY X'------- PATCH INFORMATION --------' DCHK PBUGREPRTVER CX
MODULNAM
```

This provides support for REP input via the terminal. The statement may be repeated as often as required.

Operation	Operands
HELP	

Examples of RFUPD output

Contents of the files INLDR1, INLDR2 and INLDR3

```
REP 0184B 280 X'B0'                                  70   C9005A592015 1  DJCTRL
REP 01852 111 X'4550B410'                            922E 19005A592015 1  DJCTRL
REP 0188E 280 X'04000700'                            9101 D9005A592015 1  DJCTRL
REP 01893 180 X'D0'                                  E0   29005A592015 1  DJCTRL
REP 01893 280 X'D0'                                  E0   29005A592015 1  DJCTRL
REP 018A4 280 X'4550B420'                            9604 19005A592015 1  DJCTRL
REP 018A4 280 X'17444340C00F4144000C4240C00F07F5' 0000 19005A592015 1  DJCTRL
```

Terminal log

```
/ADD-FILE-LINK LINK=LDR01,F-NAME=INLDR1,ACCESS-METHOD=*SAM
/ADD-FILE-LINK LINK=LDR02,F-NAME=INLDR2,ACCESS-METHOD=*SAM
/ADD-FILE-LINK LINK=LDR03,F-NAME=INLDR3,ACCESS-METHOD=*SAM
/ADD-FILE-LINK LINK=LDROUT,F-NAME=RFOUT,ACCESS-METHOD=*SAM
/START-PROGRAM RFUPD
%  BLS0500 PROGRAM 'RFUPD', VERSION 'V11.0A02' OF '1994-09-28' LOADED
% RFU0843 ENTER RFUPD CONTROLS
*DEL R,DJCTRL#,,,(12,C'280'),1
*DEL R,ECTLOF#L,X'7B6',X'AD4',,2
*DEL R,DAMGT1,X'3BC',X'638',,3
*COMP 1,3
*HALT
   FLAGS IN THE GENERATED LOADER
    8    A: RECORD TO BE DELETED
    1    D: REP INFORMATION OR DATACHECK FORM ERROR
    2    H: REP PARITY ERROR
    1    M: REPS OVERLAYING EACH OTHER
    1    P: DUPLICATE REP
    9    R: NO DATA CHECK PERFORMED
IN THE LOADER ARE:
    8  CLASS 1 REPS
   16  CLASS 2 REPS
    0  MODULES
 LOADER - CODE:
       0000B4A1
%  SPS0201 PRINT RFOUT ACCEPTED: TSN=5341
%  RFU0803 RFUPD PROCESSED
/
```

RFUPD log

1. Option list

```
RFUPD OPTION LIST:
                  HDRGEN
                NOCHECK
                  LDRLIST
                NOPRINTLIST
                NOTAPEGEN
                  COMP         LDR01       LDR03
                  LDROUT     = RFOUT
                  LDR01      = INLDR1
                  LDR02      = INLDR2
                  LDR03      = INLDR3
                  DEL R,DJCTRL# ,    0,FFFFF,(12,'280')     ,LDR01
                  DEL R,ECTLOF#L,  7B6,  AD4,  0,           ,LDR02
                * DEL R,DAMGT1  ,  3BC,  638,  0,           ,LDR03

                NODIS
```

2. Loader listing

```
ERRORS          BS2000   REPFILE  UPDATE  EXECUTED  06/28/84180       SOURCE  NR      PAGE  1

AR    REP 0184B 280 X'B0'                                 70   C9005A592015 1  DJCTRL  LDR01  1
      REP 01852 111 X'4550B410'                           922E 19005A592015 1  DJCTRL  LDR01  2
AR    REP 0188E 280 X'04000700'                           9101 D9005A592015 1  DJCTRL  LDR01  3
      REP 01893 180 X'D0'                                 E0   29005A592015 1  DJCTRL  LDR01  4
AR    REP 01893 280 X'D0'                                 E0   29005A592015 1  DJCTRL  LDR01  5
AR    REP 018A4 280 X'4550B420'                           9604 19005A592015 1  DJCTRL  LDR01  6
AR    REP 02C40 280 X'17444340C00F4144000C4240C00F07F5' 0000 19005A592015 1  DJCTRL  LDR01  7
```

3. Statistics listing

```
FLAGS IN THE GENERATED LOADER
 8  A:  RECORD TO BE DELETED
 1  D:  REP INFORMATION OR DATACHECK FORM ERROR
 2  H:  REP PARITY ERROR
 1  M:  REPS OVERLAYING EACH OTHER
 1  P:  DUPLICATE REP OR MODULE
 9  R:  NO DATA CHECK PERFORMED
IN THE LOADER ARE:
 8  CLASS 1 REP
16  CLASS 2 REP
 0  MODULES
LOADER-CODE:
   0000B4A1
```

Error code table

```
REPFILE UPDATE ERROR TABLE
A  RECORD TO BE DELETED
B  REP ADDRESS FORMAT ERROR
C  REP DATE PLAUSIBILITY ERROR
D  REP INFORMATION OR DATACHECK FORM ERROR
E  REP VERSION NUMBER ERROR
F  REP CLASS INVALID
G  MODULE NAME INVALID
H  REP PARITY ERROR
I  REP PARITY GENERATED
K  MODULE END CARD MISSING
L  MODULE ETPND MISSING
M  REPS OVERLAYING EACH OTHER
N  REPS OVERLAYING LDR MOD
O  EXCESS OF MODULE BOUND
P  DUPLICATE REP OR MODULE
Q  RECORD TRUNCATED
R  NO DATA CHECK PERFORMED
S  DATACHECK NEGATIVE
T  MODULE NOT IN OML
U  DIFFERENT VERSION IN OML OR LOADER
V  DUPLICATE MODULENAME
W  SAME OR LATER VER IN OML OR LOADER
X  MOD CARD SEQ NUMBER MISSING
Y  POSSIBLY MISSING CARD IN MOD
Z  MODIFICATION OF RECORD WITH DISPLAY
```

Compare listing (optional)

```
LOADER1  LOADER2                          COMPARE-LISTING OF TWO
LOADERS                          PAGE 0001

LDR01            REP 003BC 005 X'12664780E3D812AA4780E3D8188A'     12A  0 710084 023 2I DAMGT1
LDR01            REP 003CA 006 X'47F0EF3C'                         1987 0 710084 023 2I DAMGT1
LDR01            REP 00638 009 X'47F0EF2E'                         9200 B 700306 023 2I DAMGT1
         LDR03   REP 00F24 009 X'1BAA9500700E47F0E5EE'             C6D9 D 700306 023 2I DAMGT1
LDR01            REP 00F2E 008 X'12AA4770E63C9200700E47F0E63C'     C5C5 6 700306 023 2I DAMGT1
LDR01            REP 0184B 280 X'B0'                               70   C9005A592015 1  DJCTRL
LDR01            REP 01852 111 X'4550B410'                         922E 19005A592015 1  DJCTRL
LDR01            REP 0188E 280 X'04000700'                         9101 D9005A592015 1  DJCTRL
LDR01            REP 01893 180 X'D0'                               E0   29005A592015 1  DJCTRL
LDR01            REP 01893 280 X'D0'                               E0   29005A592015 1  DJCTRL
LDR01            REP 018A4 280 X'4550B420'                         9604 19005A592015 1  DJCTRL
LDR01            REP 02C40 280 X'17444340C00F4144000C4240C00F0F5'  0000 19005A592015 1  DJCTRL
         LDR03   REP 006C8 198 X'D20441B2'                         D204 A90130200061 2A ECTLOF
         LDR03   REP 006CE 198 X'D20C41A5'                         D20C 290130200061 2A ECTLOF
         LDR03   REP 006D8 198 X'D20341C7'                         D203 F90130200061 2A ECTLOF
         LDR03   REP 0070A 198 X'411061A0'                         4110 690130200061 2A ECTLOF

COMPARE-LISTING FINISHED          NUMBER OF DIFFERENT CARDS:    17
```

Printer listing of output file (optional)

```
BS2000 LOADER CODE=0000B4A1             VER:J    DATE: 06/28/84180   TIME: 084459
 REP 01852 111 X'4550B410'                          922E 19005A592015 1  DJCTRL
 REP 01893 180 X'D0'                                E0   29005A592015 1  DJCTRL
 END 11111                                                             11 11111111
 REP 003BC 005 X'12664780E3D812AA4780E3D8188A'      12A  0 710084 023 2I DAMGT1
 REP 003CA 006 X'47F0EF3C'                          1987 0 710084 023 2I DAMGT1
 REP 00638 009 X'47F0EF2E'                          9200 B 700306 023 2I DAMGT1
 REP 00F2E 008 X'12AA4770E63C92070E47F0E63C'        C5C5 6 700306 023 2I DAMGT1
 REP 00F2E 008 X'12AA4770E63C92007C0E47F0E63C'      C5C5 6 700306 023 2J DAMGT1
 REP 0074A 198 X'41104164'                          4110 890130200061 2A ECTLOF
 REP 00D32 198 X'0224'                              0214 590130200061 2A ECTLOF
 REP 006C8 198 X'D20441B2'                          D204 A90130200061 2A ECTLOF
 REP 006CE 198 X'D20C41A5'                          D20C 290130200061 2A ECTLOF
 REP 006D8 198 X'D20341C7'                          D203 F90130200061 2A ECTLOF
 REP 0070A 198 X'411061A0'                          4110 690130200061 2A ECTLOF
 REP 00F24 009 X'1BAA95007 0E47F0E5EE'              C6D9 D 700306 023 2I DAMGT1
 END 22222                                                             22 22222222
/    99999                                                             99 99999999
```

Structure of a REP file

BS2000_LOADER_comment	1st record (mandatory)
Class 1 REP records	REP records for selected modules of the resident portion of the Control System and system startup (optional)
_END[_comment]	END statement for class 1 REP records (mandatory)
Class 2 REP records	REP records for the entire Control System (optional)
_END[_comment]	END statement for class 2 REP records
/[_comment]	EOF indicator
. . . .	. . . .
/[_comment]	EOF indicator
*%comment	The comment record is output on the console.
*%%comment	The last comment record of this type is output on the terminal (applies also in the case of a quick startup).

Either a second END statement or an EOF indicator must be included as the end criterion for class 2 REP records and the REP file.

Format of the REP records

a: alphanumeric character (0...9, A...Z, $, #, @, &, %, −)

c: alphanumeric character (0...9, A...Z)

e: hexadecimal character (0...9, A...F)

n: digit (0...9)

z: letter (A...Z)

Column	Contents	Meaning
1	_	blank
2-4	REP	
5	_	blank
6-10	xxxxx	REP address, relative to start of module
11	_	blank
12-14	3-digit integer	Number of current day in the year (DAY)
15	_	blank
16	X	
17-50 (max.)	' xxx...x' or ' xx...x' + name	Up to 32 characters of correction data enclosed in apostrophes; or up to 22 characters of correction data enclosed in apostrophes and followed by a plus sign and the name of a Control System module. The start address of this module is added to the last 6 digits of the correction data during system initialization. This name must be specified as an 8-character name corresponding to the name given in columns 73-80. Shorter names must be padded with blanks to produce 8 characters (patch information).
51	_	blank
52-55	xx or xxxx	2 or 4 characters of check data. The first byte or the first two bytes that are to be overwritten are to be specified (DCHK).
56	_	blank
57	x	Check number (parity byte) for REP address, correction data and check data.
58-65	aaaaaaaa	Error number
66-68	aaa	Module version
69	a	REP identifier
70	1 or 2	1 for class 1 REP records, 2 for class 2 REP records

Column	Contents	Meaning
71	a or _	Loader version (A-Z) or blank
72	a	reserved for loader management
73-80	aaaaaaaa	Module name The start address for this module is added to the REP address.

a: stands for an alphanumeric character (0-9, A-Z)

x: stands for a hexadecimal character (0-9, A-F)

Explanation of the check number (column 57)

The check number in column 57 is determined as follows:

x = (A + B + C + D) mod (16)

where A = sum of the digits in the REP address
B = sum of the digits in the REP address
C = number of correction data digits
D = sum of the check data digits

mod (16) means:

The sum of A+B+C+D is to be divided by 16. The remainder is the check number being sought (l), where $0 \leq l \leq 15$.
Example: 20 mod (16)=4.

The following REP identifiers are permitted:

– D = diagnosis/intercept REP
– O = optional REP
– S = selectable unit
– T = trace (activate)
– U = selectable unit, optional
– _ = normal REP

Format of the module records

Column	Contents	Meaning
1	_	blank
2-4	MOD	
5	_	blank
6-9	VER=	
10-19	*NO ETPND* or nnnxjjmmtt	No ETPND in this module. ETPND information for this module: nnn = version x = macro library yymmdd = date from ETPND, yy = year mm = month dd = day
20	_	blank
21-24	ESD=	
25-27	nnn	Number of ESD entries
28	_	blank
29-32	RLD=	
33-35	nnn	Number of RLD records
36	_	blank
37-40	TXT=	
41-44	nnnn	Number of TXT records
45	_	blank
46-49	SIZ=	
50-54	eeee	Size of module
55	_	blank
56-59	DAY=	
60-62	*** or nnn	Day when the module was compiled
63	_	blank
64-68	XTRN=	
69-71	nnn	Number of EXTERNs
72	_	blank
73-80	aaaaaaaa or 8X' 40'	Module name

Notes on use

– For technical reasons, only a limited amount of contiguous storage space can be requested in SPL. Consequently, modules longer than 29K cannot be edited. The REP check is not performed for such modules.

– When correcting addresses, it is not generally possible to specify a data check byte, as the addresses are not inserted/relocated until later by the linkage editor (depending on module and system base).

  In this case it is necessary to switch to the preceding bytes in the REP record, generally to the first byte of a 4-byte address. The REP address must then reference this byte, and the correction data must also contain this byte; this is effective in 24-bit addressing mode only.

  In fact, in an addressing mode > 24-bit the leftmost byte of a 4-byte address cannot be used as a data check byte because it forms part of the addressing. In such an addressing mode there is no longer any data check byte present. Instead, RFUPD flags the corresponding REP record with the error code "R" in the loader listing to signal that no data check has been carried out. In this case the error code should be interpreted only as an indicator.

– Relative REP records (i.e. records with "+" following the correction data) may occur more than once for the same address of a module, e.g. if the EXTRN address is a # module. In such cases RFUPD always sets the "M" flag.

12 SODA Evaluation of SLED and SNAP dumps

Version: **SODA V11.2A**

BS2000/OSD-BC V2.0A is the last version to support SODA; in the future, it will be replaced by DAMP (see the "Diagnostics Handbook" [6]).

SODA (Selective Organizing Diagnosing Analyzer) is a dump analyzer for dump files (SLED or SNAP dumps) produced by the dump generator routines SLED and SNAP.

SODA can edit the contents of these dump files and output them to SYSLST or SYSOUT. The system files present in the errored system during the dump can be extracted from the dump by SODA and generated as a file in the diagnostic system.

SODA is capable of evaluating SLED dumps located on NK4 disks, but cannot generate system files on NK4 disks since it does not support the conversion of BLKSIZE.

Support for multiprocessor operation

SODA also supports dumps generated during multiprocessor operation. For each CPU the program context, special registers and logout area are dumped.

Restricted processing mode

If the system to be analyzed is not BS2000, or if destroyed system tables prevent it from being recognized as BS2000, SODA performs a neutral analysis of the dump.
In this case the restricted processing mode is indicated by the message `COMMAND PROCESSING RESTRICTED`. If the message `COMMAND EXECUTION SUPPRESSED` appears in this mode, this means that SODA is not executing the statement that was entered. The message `COMMAND EXECUTION RESTRICTED` indicates that the statement entered can only be executed with certain restrictions.

The BASE and MEMORY statements are processed with restrictions. If the operands HW=Y, SYS=Y, MINI=Y, NOREF=Y and ALL=Y are specified in the BASE statement, the hardware areas and SLEDLOG are output. In the case of the MEMORY statement, the REAL and HSA operands (if present) are interpreted in full, while the VIRTUAL operand is not interpreted in full unless the translation tables are available.
The CALL, CAT and TASK statements are not processed at all in the case of restricted processing.

The statements END, FILE, FSTAT, GEN, HELP, PARAM, SYSLST, SYSOUT and TEXT are processed without restrictions even in restricted processing mode.

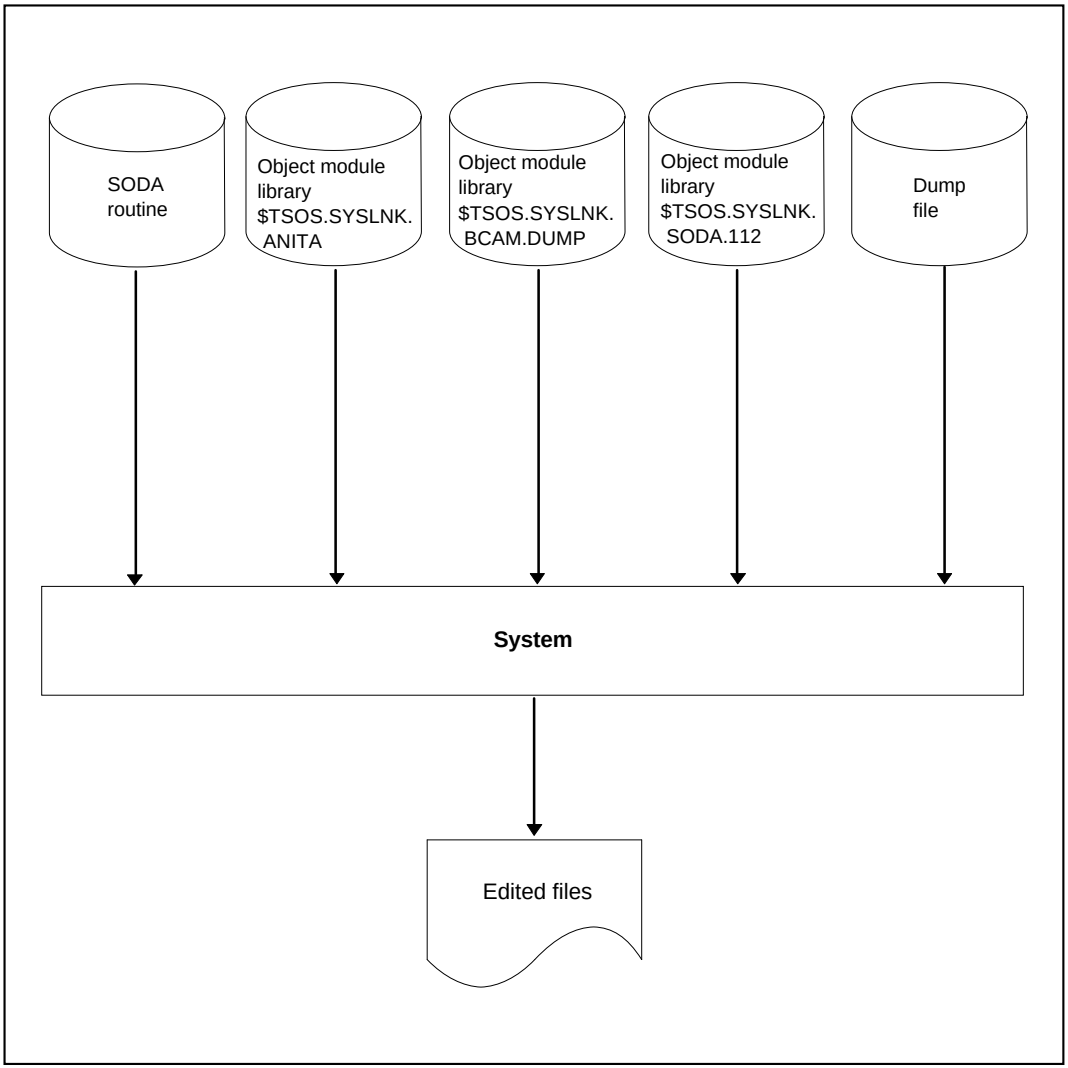


Figure 15: Data flow for the SODA routine

12.1 Task analysis

SODA performs task analysis in order to obtain information relevant to a given error. Among other things, this procedure picks out those tasks which may have caused the error.
All tasks from the following three categories are selected:

– every task which has led to a system error (error task)

– the task responsible for system abortion (crash task)

– each task in control (TIC) of a CPU.

There is usually one controlling task for each CPU; there may be one crash task, and no, one or more than one error task. Any given task may belong to more than one of these categories. The system modules being used by each of these tasks at the time of the system crash are also output.

12.1.1 Error task

Since all tasks are potential error tasks, SODA checks every task for this possibility. SODA searches the task location table (TLT) to find out which tasks are in the system. At the same time it attempts to identify the error PCB.

A task is regarded as a potential error task if at least one of the following conditions is fulfilled:

– The task has more than 100 PCBs.

– "Task pended indefinitely". Here the uppermost TPR-PCB is regarded as the error PCB.

– There is a PCB that was set up in response to an $EXER or CDUMP SVC (in the second case the system also checks for the presence of a TERMD SVC). This PCB is then the desired error PCB.

– All tasks for which a system dump is being generated are additionally output as error tasks, without error PCB identification. The program counter for the error task concerned determines the system module in use at the time of the system crash.

In the case of ESA^{TM} systems, the PCB is extended to include the ACCESS registers.

12.1.2 Crash task

The SODA routine determines

– whether a system error terminated the session. Whenever this is the case, the $CRASH call will have been used to activate the software error handler in the SIH, saving the SIH registers within the processor-specific portion of the XVT table

– whether any task can be found responsible, and whether the address lies within the system module where the error occurred.

12.1.3 Task in control (controlling task)

The XVT entry EXVTTIC contains the address of the TCB of the task in control.

If a controlling task is also an error task or the crash task, the system module is found by the error task method or the crash task method. If not, the program counter of the current PCB is used to search the EOLDTAB. If the address lies within the boundaries of a given module, that module is used.

12.2 Operation

12.2.1 SLED dump on tape

SODA can process only disk files. If the SLED dump due for processing resides on tape, it must be copied to disk before it can be analyzed. However, at the same time it must be converted using the software product PERCON as the block format of the SLED tapes is not the same as the disk format.

Tape files are transcribed to disk using the following commands:

```
/IMPORT-FILE     SUPPORT=*TAPE(VOLUME=tape vsn/vsn list,
                 DEVICE-TYP=type of tape device,
                 FILE-NAME=SLEDFILE/freely selectable file name)


/ADD-FILE-LINK   LINK-NAME=PCIN,
                 FILE-NAME=SLEDFILE/freely selectable file name,
                 ACCESS-METHOD=*SAM,RECORD-FORMAT=*BY-CATALOG,
                 BUFFER-LENGTH=*BY-CATALOG


/CREATE-FILE     FILE-NAME=freely selectable name of disk SLED file,
                 SUPPORT=PUBLIC-DISK(
                         SPACE=*RELATIVE(PRIMARY-ALLOCATION=size,
                         SECONDARY-ALLOCATION=extent size))


/ADD-FILE-LINK   LINK-NAME=PCOUT,
                 FILE-NAME=freely selectable name of disk SLED file,
                 ACCESS-METHOD=*UPAM


/START-PROGRAM   FROM-FILE=PERCON


END
```

If a freely selectable file name is used for the tape file instead of SLEDFILE, PERCON requests an acknowledgment for the copy operation.

These statements can be used to transcribe any SLED tapes to disk, regardless of the block format and of the BS2000 version used.

12.2.2 SLED dump in a private volume

If the dump file is on a private volume, an IMPORT-FILE command must first be issued.

Example

The dump file resides on a private disk and has not yet been cataloged in the system catalog (TSOSCAT). The file name is DUMP1, the volume serial number of the disk is C0135A, and the device type code is 3480.

The following commands must be issued before starting SODA:

```
/IMPORT-FILE SUPPORT=*DISK(VOLUME=C0135A,DEVICE-TYP=D3480,FILE-NAME=DUMP1)
/ADD-FILE-LINK FILE-NAME=DUMP1,LINK-NAME=SODA01
```

12.2.3 Starting the program

First of all the file to be analyzed must be assigned the file link name SODA01, otherwise SODA cannot access it. This link name is not released after analysis, with the result that the same file can be analyzed several times in succession according to different criteria.

Example

```
/ADD-FILE-LINK FILE-NAME=DUMP2,LINK-NAME=SODA01
```

SODA is started with the following command

```
/START-PROGRAM  [$]SODA
```

The `CPU-LIMIT`, `TEST-OPTIONS`, `MONJV`, `RESIDENT-PAGES` and `VIRTUAL-PAGES` operands of the `START-PROGRAM` command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the `START-PROGRAM` command in the "Commands, Volume 3" manual [3].

The SODA routine can run as a batch task, a remote batch task or an interactive task, using the same control statements in each case.

12.2.4 Interrupting and resuming output

Hitting the K2 key switches to command mode and interrupts output. Subsequently, the /RESUME-PROGRAM command can be used to revert to program mode and continue output. In order to terminate output to SYSOUT and initiate another SODA action, the command `/SEND-MSG TO=*PROGRAM` must be issued in command mode. This likewise returns the user to program mode, but SODA thereafter expects a new statement.

12.2.5 Working with SODA

For a first analysis using SODA, it is advisable to start SODA and then issue the statement BASE MINI=Y (see also B NOREF=Y) followed by the END statement. This is usually enough to generate sufficient data for an error analysis. Should additional task-specific data be required, it can be fetched explicitly by means of the TASK statement.

If required, the user can output system-specific or real address space with the aid of the MEMORY statement. If system files are to be analyzed in the course of further analysis (CONSLOG, EQUISAMQ, SERSLOG, SJMSFILE, TSOSCAT, SJOBPOOL, HERSFILE, VM2LOG and SLEDLOG), these system files can be accessed via the FILE statement. The CAT statement can be used for explicit output of the catalog file TSOSCAT.

12.3 Output format

12.3.1 Output file

Unless otherwise specified, all data to be printed out is written to the system file SYSLST.

However, in interactive mode the user may specify via the SYSOUT statement that all output data is to be written to the SYSOUT system file. If data output is to revert to SYSLST in the course of subsequent processing, the SYSLST statement must be entered.
The logging format of the output data may vary for internal reasons.

12.3.2 Page layout

The output for every individual statement and new section within a statement begins on a new page. Each page starts with a header line whose contents do not vary except for the page number. The header line contains the following items:

– name of the program producing the listing
– version number of the program generating the listing
– identification of the system to be analyzed
– version number of the system to be analyzed
– generation date of the system to be analyzed
– session number of the system to be analyzed
– date on which the system to be analyzed was loaded
– date and time of the SLED run
– page number.

The header line is followed by the title line, which contains the following items:

– description of the data being listed
– text from the TEXT statement or, if no TEXT statement was entered, contents of the last statement entered.

12.3.3 Output format of memory blocks

When memory blocks are output, 32 bytes are listed per line, in hexadecimal and in character format. In hexadecimal format, words are separated by spaces. The first one or two fields in each line are used for line identification, and may have the following meanings:

– absolute, virtual or physical (byte) address
– relative (byte) address to start of block
– relative (byte) address to any arbitrary base
– line number relative to start of block.

When memory blocks are printed, lines are omitted if:

– each word in the line is the same as the last word in the previous line; this type of omission is indicated by an asterisk (*) at the beginning of the first line following the omission
– the contents of the line are the same as those of the previous line; this type of omission is indicated by two asterisks (**) at the beginning of the first line following the omission.

12.3.4 Table of contents

If the program has been terminated normally, a table of contents is output to the system file SYSLST. Each section and subsection of the output data is listed in this table of contents, along with the number of the page on which the section starts.

Data output via SYSOUT to a terminal does not appear in the table of contents.

12.4 Statements

12.4.1 Format of the statements

The following points should be borne in mind when entering statements:

– the statements do not start with a slash
– the maximum statement length is 116 characters
– continuation lines are not permitted.

The statements are represented as follows:

Operation	Operands

The statements can be used to access all the data in the dump file.

12.4.2 Overview of the SODA statements

The following table lists all the SODA statements in alphabetical order:

Statement	**Function**
BASE	Extract basic information for diagnosis
CALL	Call up routines for special analysis
CAT	Print out information from the catalog file
CSECT	Output system modules in module-relative form
DSPACE	Output contents of a data space
END	Terminate the routine
FILE	List the contents of a system file
FSTAT	List the catalog entry for a system file
GEN	Generate a system file
HELP	List the SODA statements
MEMORY	List a memory area
PARAM	Control the number of lines for print output
SYSLST	Output print data to the SYSLST system file
SYSOUT	Write output data to the SYSOUT system file
TASK	List tasks in the same manner as the BASE statement
TEXT	Insert text into the title line

Table 5: Overview of the SODA statements

12.4.3 Descriptions of the statements

BASE
Extract basic information for diagnosis

The BASE statement extracts the basic information required for diagnosis.

The following table shows which information is output by specifying which operands. A detailed description of the information is given with the individual operands.

	Operands								
Output information	**SYS**	**TRACE**	**KEYS**	**MAP**	**ALLT**	**HW**	**MINI**	**NOREF**	**ALL**
System information	X						X	X	X
Complete trace table		X					X	X	X
Memory protect keys			X						X
Directory (map) of the Control System modules				X			X	X	X
Page selection for all tasks					X		X		X
Hardware areas/ SLEDLOG						X	X	X	X

Table 6: BASE statement: output information for the operands

Format and operand descriptions

Operation	Operands
{ BASE B }	[SYS=Y] [,TRACE=Y] [,KEYS=Y] [,MAP=Y] [,ALLT=Y] [,HW=Y] [,MINI=Y] [,NOREF=Y] [,ALL=Y]

If an operand is not specified in the BASE statement, its associated function is not performed.

If BASE is specified without operands, there is no output.

SYS=Y System information

The following information is output:

1. System synopsis
   - task sequence number, internal task number (now TID) and TCB address of the task responsible for CPU control
   - SIH, save area of the software error handler (if this is full)
   - name and relative address of the Executive module responsible, via $CRASH, for the system shutdown (if such a module can be identified)
   - name and relative address of the Executive module responsible for $CRASH being called (if such a module can be identified)
   - TSN, TID and TCB address of tasks recognized as containing errors
   - saturation indicator and saturation levels.

2. Task table for all current tasks.
   The following information is output for each current task:
   - internal task number (TID, task identifier)
   - address space number (ASN)
   - TCB address
   - task sequence number
   - task priority (PR)
   - queue number
   - pend code
   - data from the current TPR PCB (SVC, PC, module with relative address)
   - command
   - program name
   - an asterisk (*) before the TID of the task means that there is at least 1 PCB in exit mode
   - an asterisk (*) before P-COUNT identifies program counters originating from bourse calls.

3. Logging file (CONSLOG)
   The logging file is output in abbreviated form, i.e. the first three data blocks and the last three data blocks.
   If output of the entire logging file is desired, this can be requested using the FILE statement.
   The log of the SLED run (SLEDLOG) is output immediately after output of the abbreviated logging file.
4. Module CLASS20P
5. I/O trace table
   The SDITT (Start Device Interrupt Trace Table) is output, arranged in chronological order from most recent back to oldest entry. The entries are displayed in hexadecimal and in printable form.
6. CCBs and CCWs of the current I/O operations
7. System trace dump list

8. Task-specific information (for all tasks selected by SODA):
   - trace table entries for the task concerned
   - TU and TPR audit
   - linkage AUDIT (if activated)
   - task control block (TCB)
   - PCBs
   - job control block (JCB)
   - job-to-be-processed block (JTBP)
   - job-to-be-processed block extension (JTBPX)
   - PU-CSECT LIST; list of named sections from the program unit
   - class 5 or class 6 memory pages addressed.

9. Data space management table
   The management table is output in edited format if it is contained in a SLED file. The entries in the management table which start with an asterisk refer to privileged data spaces. Privileged data spaces are contained in SLED files created with MODE=STD, provided that SLED was able to find the data structures required to locate these data spaces.

10. Selected pages from the system address space. These include, for example, important system tables such as XVT, TLT, SVMT etc.
    Selection criteria include program counter, general-purpose and bourse registers of the PCBs, plus the general-purpose registers of the SPL stacks and the save areas. If an address here indicates an Executive module, the entire module is output.

TRACE=Y

- Directory of trace areas in the system; SYSTEM TRACE DUMP LIST.
- Complete system trace table.
  The entries are numbered consecutively, the chronologically last entry receiving the number 1, the penultimate the number 2, etc.

KEYS=Y

Memory protect keys
The page number (in parentheses) is followed by one or two memory protect keys for this page, depending on whether 4-Kb or 2-Kb pages are used. Only the first 5 bits are significant.
Adjacent pages with identical memory protect keys are enclosed in parentheses.

MAP=Y

Map of the Control System modules
The name, virtual load address, size (hexadecimal) and version number are output for each module. This list is output once sorted by start addresses and once sorted by module names.

ALLT=Y

All tasks are taken into account for the output of pages from the system address space. Output takes place only if SYS=Y is also specified. If ALLT=Y is not specified, only the tasks selected by SODA are taken into account.

HW=Y

Causes output of the SLEDLOG and (depending on the type of system) the following hardware areas:

- program context; registers whose contents cannot be accessed are overwritten with question marks (?)
- special registers, separated according to CPU; registers whose contents cannot be accessed are overwritten with question marks (?)
- CPU logout areas and save areas
- local processor audit
- SAVED STATUS
- PSA (prefixed storage area)
- PSA in edited form, i.e.:
  general registers
  PSW save areas
  control registers
  interrupt code save areas
  floating-point registers and other software information.

If a SLED dump is taken by SLED and HW=Y is specified, the IPL-EXEC and SLED codes will also be output.

MINI=Y	Standard output; includes the operands SYS, TRACE, ALLT, MAP and HW.
NOREF=Y	Standard output as for MINI=Y except that no referenced pages are printed.
ALL=Y	Complete output; includes all operands.

CALL
Call routine for special analysis

This statement is used to call one of a number of special analysis routines.

Format and operand description:

Operation	Operands
CALL	name [,libname]

name — Name of the function to be called (also the name of the entry point into the routine to be called).
A search for "name" is made in the object module library $TSOS.SYSLNK.SODA.xxx; the module associated with this name is then linked to SODA and activated.

The following names (functions) are permitted:

DAB:	Output of specific DAB data
DCM:	Output of specific DCM data
BOURSE:	Output of specific bourse management data
RECONF:	Output of specific reconfiguration data
VM#DUMPS:	Output of VM2000-specific data

libname — is the name of the library in which the search for the name of the function to be called is to take place.

When the external routine DCM is called, it is loaded from the library '$TSOS.SYSLNK.BCAM.DUMP' unless a different library name is specified. All other external routines are loaded as before from the library '$TSOS.SYSLNK.SODA.xxx' unless a different library is specified in the CALL statement.

i Called routines are not unloaded after execution. This means that if the CALL statement is executed again with the same "name" operand, the same routine is activated even if different library names are specified.

CAT
Print information from catalog file

This statement causes the specified section of the catalog file to be printed out.

Format and operand description:

Operation	Operands
CAT	[{USER=[TSOS] / TID={tid / ALL} / TSN=tsn / BLOCKS=[{n / ([a] [,b])}]} [,USED={Y / N}]]

USER=TSOS — The data blocks of the catalog file belonging to the specified user ID $TSOS are listed. If no operand value is specified, the data blocks of all user IDs are listed.

TID=

=tid — The data blocks of the catalog file belonging to the user ID associated with the specified internal task number are listed. If the TID of a permanent system task is specified, the catalog under user ID TSOS is listed. The task number must be entered as an 8-digit hexadecimal value, e.g. TID=X'0001012A'. Leading zeros may, however, be omitted, e.g. TID=X'1012A'.

=ALL — The data blocks of the catalog file belonging to the specified user IDs are listed for all active tasks; the listing is sorted by internal task numbers.

TSN=tsn — The data blocks belonging to the user ID associated with the specified task sequence number are listed. This value can be entered as a 1- to 4-digit decimal number.

For permanent system tasks the alphabetical task code may be specified, e.g. UCO for UCON tasks.

BLOCKS

=n — Block number "n" is listed, if used. If the block is not used, it is listed only if the operand USED=N is also specified.

=(a,b) Data block numbers "a" to "b" of the catalog file are listed. If "a" is omitted, the listing begins with the first data block in the catalog file. If "b" is omitted, the listing is carried through to the last data block (see also the USED operand, below).

If no operand value is specified, all data blocks from the first to the last one are listed (see also the USED operand, below).

USED

=Y Only used data blocks are listed (default value).

=N Unused data blocks of the catalog file are also listed in the output, as well as the used blocks.

The USED operand is evaluated only if the BLOCKS operand has been specified.

If the CAT statement is specified without operands, the entire catalog is listed, sorted by user IDs (equivalent to the specification "CAT USER=", default value).

Example

```
CAT BLOCKS=(3),USED=N
```

All data blocks from the third to the last block of the catalog file are listed.

```
CAT BLOCKS=(1,5),USED=Y or CAT BLOCKS=(,5)
```

Only used data blocks from the first to the fifth block of the catalog file are listed.

CSECT
Output system modules in module-relative form

This statement is used to output system modules. The module names must be present in the EOLDTAB.

Format and operand description:

Operation	Operands
CSECT	{ moduelname (modulename1,modulename2,...) }

modulename — Name of the module to be output.

Up to 14 module names can be specified. If the same module name is specified several times, it is still only output once.

DSPACE
Output contents of data space

DSPACE outputs the contents of the specified data space.

Format

Operation	Operands
{ DSPACE DS }	SPID=X‘aaaaaaaabbbbbbbb‘ ,V={ (X‘aaaaaaaa‘[,{ X‘bbbbbbbb‘ X‘0‘ }]) ALL }

SPID — Space identifier of the data space to be output (doubleword).

V=(X’aaaaaaaa’ — Start address of the area to be output. This address is rounded down to the nearest word (4-byte boundary) The address "aaaaaaaa" must be smaller than the address "bbbbbbbb". If the end address is not specified or if it is specified as X’0’, the start address is rounded down to the start of the page (X’aaaaaOOO’).

,X'bbbbbbbb')	End address of the area to be output. This address is rounded down to a word boundary minus 1 (4-byte boundary). If the end address is not specified or if it is specified as X'0', the end address is set to X'aaaaaFFF'. This makes it possible to output the whole of the relevant virtual page by specifying only the start address.
=ALL	The entire contents of the data space are to be output.

END
Terminate program

This statement brings execution of the SODA routine to a controlled conclusion.

Format

Operation	Operands
{END E}	

FILE
List contents of system file

This statement causes the contents of the specified system file to be printed out.

Format and operand description:

Operation	Operands
FILE	filename

filename — Name of the system file whose contents are to be listed. Possible file names are:

CONSLOG	System logging file (the line length is restricted to 132 characters). SODA also outputs the CONSLOG I/O buffer.
EQUISAMQ	System queue file for spoolout jobs.
HERSFILE	Statistics file of the hardware error logging system.
SERSLOG	Software diagnostic file; SODA also outputs the SERSLOG I/0 buffer.
SJMSFILE	File for stream and job class definitions.
SJOBPOOL	Batch job queue file. The file name is SYSTEM.JOBPOOL if the SLED dump was taken during normal system operation, and SYSTEM.JOBPOOL.COPY if the SLED dump was taken during startup.
SLEDLOG	Log of the SLED run.
TSOSCAT	System catalog; however, only the used data blocks of the system catalog are listed. To list all the data blocks of TSOSCAT, the CAT statement must be used.

FSTAT
Output catalog entry for system file

This statement tells SODA to output the catalog entry for a specified system file. Any of the FILE and GEN statement operands are permitted as system file names.

Format and operand description:

Operation	Operands
FSTAT	filename

filename — Name of the system file whose catalog entry is to be output. Possible names are:

CONSLOG	System logging file.
EQUISAMQ	System queue file for spoolout jobs.
HERSFILE	Statistics file of the hardware error logging system.
SERSLOG	Software diagnostic file.
SJMSFILE	File for stream and job class definitions.
SJOBPOOL	Batch job queue file. The file name is SYSTEM.JOBPOOL if the SLED dump was taken during normal system operation, and SYSTEM.JOBPOOL.COPY if it was taken during startup.
SLEDLOG	Log of the SLED run.
TSOSCAT	System catalog.

GEN
Generate system file

This statement generates a specific system file from the dump file as a separate file. When generating system files, the LMS library $TSOS.SYSLNK.SODA.112 must be available and must contain the PAMINT conversion module.
For the system files SERSLOG and CONSLOG, the I/O buffer is also written as the last block, assuming SODA is able to access it.
The effect of the GEN statement depends on the format of the system file saved by SLED (K or NK format) and on the version of BS2000 under which SODA is running.
If SODA is running under a version ≥ V10, it generates all files in NK2 format, i.e. SAM files with BLKCTRL=DATA, ISAM files with BLKCTRL=DATA or DATA2K and PAM files with BLKCTRL=DATA or NO. With versions ≤ V9.5, SODA generates all files in K format, i.e. BLKCTRL=PAMKEY.

If required, SODA converts the format of the system file. To convert from K to NK format or vice versa, PAMINT is called. PAMINT is loaded from the library $TSOS.SYSLNK.SODA.112 and corresponds to the conversion routine PAMCONV (see page 277).

Default name of the generated file

The generated file has the default name

```
SODA.nnn.yymmdd.hhmmss.filename
```

where

nnn	is the session number of the system to be analyzed.
yymmdd	is the date of the SLED run in the form: year, month, day.
hhmmss	is the time of day of the SLED run, in the form: hour, minute, second.
filename	is the operand selected in the GEN statement.

The CREATE-FILE and ADD-FILE-LINK commands may be issued prior to the SODA run to specify a file name and output medium different from the default value. The "filename" operand given in the GEN statement must then be used as the file link name (LINK-NAME operand).

Output to disk: it saves time if generous primary and secondary space allocations are defined in the CREATE-FILE command:

```
SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE
(PRIMARY-ALLOCATION=3000,SECONDARY-ALLOCATION=3000))
```

Output to tape: only standard labels are supported. Multifile tapes (multifiles) and multivolume tapes (multivolumes) are not supported.

Format and operand description:

Operation	Operand
GEN	filename

filename

Name of a system file to be generated as an independent file. Possible file names are:

CONSLOG	System logging file.
EQUISAMQ	System queue file.
HERSFILE	Statistics file of hardware error logging system.
SERSLOG	Software diagnostic file.
SJMSFILE	File for stream and job class definitions.
SJOBPOOL	Batch job queue file. The file name is SYSTEM.JOBPOOL if the SLED dump was taken during normal system operation, and SYSTEM.JOBPOOL.COPY if it was taken during startup.
SLEDLOG	Log of the SLED run.
TSOSCAT	System catalog.
HYP	VM2000 hypervisor, controls execution of guest systems on the virtual machines.
VM1...15	Virtual machines 1-15. SODA generates them as a separate SLED file, which can then be analyzed by a SODA compatible with the BS2000 version of the generated system.

HELP
List SODA statements

This statement produces a list of all available SODA statements.

Format:

Operation	Operands
HELP	

MEMORY
Output memory areas

This statement is used to output the specified area of main memory and/or of the virtual system address space.

Format and operand description:

Operation	Operands
{ MEMORY | M }	[{ ABSOLUTE | A | REAL | R | HSA | H } = { (X‘aaaaaaaa‘[,[{ X‘bbbbbbbb‘ | X‘0‘ }]]) | Y }] [,{ VIRTUAL | VIRT | V } =(X‘aaaaaaaa‘[,[{ X‘bbbbbbbb‘ | X‘0‘ }]])] [,CL1=Y] [,CL2=Y] [,CL3=Y] [,CL4={ Y | VAT | FP | NP | PP }]

R[EAL]
A[BSOLUTE]
H[SA]

The main memory area between the specified addresses is output.

REAL	refers to the start of the BS2000 system or, for VM2000, to the start of the monitor machine.
ABSOLUTE	refers to the start of main memory, i.e. for VM2000, MEMORY ABSOLUTE starting from address 0000 outputs the start of the hypervisor.
HSA	the specified section of the hardware system area is output.

=X’aaaaaaaa’ Main memory is to be output starting from the specified hexadecimal address. The address is rounded down to 32 bytes. If the specified address lies outside real memory, the start address of the last real page is assumed instead. The address "aaaaaaaa" must be less than "bbbbbbbb". If the end address has been specified as X’0’, or has been omitted altogether, the start address "aaaaaaaa" is rounded down to the beginning of the page (X’aaaaa000’).

=X’bbbbbbbb’ Main memory is to be output up to the specified hexadecimal address. The address is rounded up to 32 bytes. If the specified address lies outside real memory, the end address of the last page is assumed instead. If the specification is omitted or X’0’ is specified, X’aaaaaFFF’ is assumed as the end address. In this way, specification of the start address alone will effect output of the whole associated real page.

=Y Outputs all of main memory. This operand provides a means of obtaining a full memory dump when the system has been destroyed to such an extent that analysis by way of the BASE statement has no point. If HSA=Y is specified, the complete hardware system area (HSA) is output.

VIRTUAL=
VIRT=
V=

The virtual system area between the specified addresses is output.

=X'aaaaaaaa'	Start address of the system address area to be output. This address is rounded down to the nearest word (4-byte boundary). Address "aaaaaaaa" must be smaller than "bbbbbbbb". If the end address is omitted or if X'0' is specified, "aaaaaaaa" is rounded down to the beginning of the page (X'aaaaa000').
=X'bbbbbbbb'	End address of the system address area to be output. This address is rounded up to the next word boundary minus 1 (4-byte boundary). If the end address is omitted or if X'0' is specified, X'aaaaaFFF' is assumed as the end address. It is thus possible to output the complete virtual page by specifying solely the start address. If the end address is less than the start address of virtual system address space, the statement is rejected.
CL1=Y	Outputs the entire class 1 system address space.
CL2=Y	Lists the entire contents of the class 2 address space.
CL3=Y	Outputs the entire class 3 system address space.
CL4	
=Y	The entire contents of the class 4 address space are output using the VAT (virtual attribute table). The VAT is a table defining the attributes of pages of the system address space.
=VAT	The attributes of class 4 pages stored in the VAT are dumped.
=FP	Causes dumping of all fully allocated privileged class 4 pages on the basis of the VAT.
=NP	Causes dumping of all fully allocated nonprivileged class 4 pages on the basis of the VAT.
=PP	Causes dumping of all class 4 pages of which parts are allocated.

PARAM
Control number of lines in printed output

This statement controls the number of lines in the printed output.

Format and operand description:

Operation	Operands
{ PARAM | P }	LINES=linecount

LINES=linecount — Defines the page size of the SODA printer listing. This number does not have to correspond to the actual size of the paper. The specified number includes the number of lines required internally for title, etc., and for the data to be output.
The value must be specified as a two-digit decimal. The default value is 72 (6 lines/inch and 12 inches/page, or 8 lines/inch and 9 inches/page). The minimum value is 42.

SYSLST
Write output data to SYSLST system file

This statement causes all printout data to be written to the SYSLST system file. Since this will happen by default in any case, the SYSLST statement need be issued only if the user wishes to return to output via SYSLST after temporary output to the SYSOUT system file (see the SYSOUT statement).

Format:

Operation	Operands
{ SYSLST | LST }	

SYSOUT
Write output data to SYSOUT system file

This statement causes all output data to be written to the SYSOUT system file, assuming that SODA is running as an interactive task.

Format:

Operation	Operands
{ SYSOUT / OUT }	

TASK
List tasks in same way as BASE statement

This statement is used to specify one or more tasks that are to be listed in the same way as with the statement BASE SYS=Y. TASK does not select any system modules for normal tasks.

Format and operand description:

Operation	Operands
{ TASK / T }	{ TID={ tid / ALL } / TSN=tsn } [,UVM{ REF / CL5 / CL6 / ALL / (X'aaaaaaaa'[,[{ X'bbbbbbbb' / X'0' }]]) }] [,STACKS=ALL]

TID

=tid — Internal task number ("tid"= task identifier). This operand selects a single task by means of its internal task number. The number must be entered as an 8-digit hexadecimal value, e.g. TID=X'0001012A'. Leading zeros may optionally be omitted, e.g. TID=X'1012A'.

=ALL — Lists all tasks currently in the system.

TSN=tsn — Task sequence number. This operand selects a single task by specifying its 1- to 4-digit task sequence number.

UVM

=REF — The virtual user address space is output; that is, only the accessed pages of class 5 and class 6 memory.

=CL5 — Only the class 5 pages are output.

=CL6 — Only the class 6 pages are output.

=ALL — Both the class 5 and the class 6 pages are output.

=X'aaaaaaaa' — Specifies the start address of the virtual user address space to be output. It is rounded down to the next word boundary (4-byte boundary) if the end address is greater than or equal to the start address. If the end address is not specified or if X'0' is specified, the address is rounded down to the beginning of the page (X'aaaaa000').

,X'bbbbbbbb' — Specifies the end address of the virtual user address space to be output. This address is rounded up to the next word boundary minus 1 (4-byte boundary). If the end address is not specified or if X'0' is specified, X'aaaaaFFF' is assumed as the end address. It is thus possible to output the complete virtual page merely by specifying the start address.

i If the boundaries of the virtual user address space are specified, only the desired address space is output, i.e. not the other task data normally output for other operands. If no value is specified, X'0' is assumed by default.

STACKS=ALL — All PCBs associated with the task are to be output. If the operand is omitted, SODA outputs a maximum of 100 PCBs per task. This operand should be used with due caution, as it may cause SODA to be caught in an endless loop if the PBCs are connected cyclically.

TEXT
Insert text into title line

This statement enables text to be inserted into the title line of the output listing.

Format and operand description:

Operation	Operands
{ TEXT / X }	C‘text‘

text — A string of characters enclosed in apostrophes. Each TEXT statement contains the text to be inserted into the title line of the output listing. A maximum of 60 characters between apostrophes is allowed. If the string itself contains an apostrophe (’), it must be specified in duplicate (”).

12.5 Messages

The messages for the SODA routine are described in detail in the "System Messages" manuals [16], [17].

Notes

SODA routine notes contain information on SLED and SODA version numbers, on general diagnostic results, on invalid data and on special problems that may occur when using particular statements.

Some notes are output via SYSOUT, some via SYSLST.

12.5.1 SYSOUT error messages

SYSOUT is used for the output of those error messages intended to inform the user of processing problems, e.g.:

– incompatibility of the SLED and SODA versions used
– errors in the execution of a REQM, OPEN, FILE or PAM macro
– errors in user statements
– errors when accessing the EOLDTAB etc.

12.5.2 SYSLST error messages

SYSLST is used for the output of all system error messages.

System error messages occur when analysis reveals the existence of errors in system tables. These errors impair execution of certain functions of the SODA routine.

System error messages are subdivided into three groups:

– invalid addresses in task-related tables
– invalid addresses in general system tables
– system errors that cannot be assigned to either of the first two categories.

Depending on the error type, an appropriate message is issued:

– TASK system error messages associated with invalid addresses.

```
Format:???  vvvvvvv SYSTEM ERROR TASK wwwwwwww, xxx...x CONTAINS
            ADDRESS yyy...y (zzz...z) ???
```

– NONTASK system error messages associated with invalid addresses.

```
Format:???  vvvvvv SYSTEM ERROR: xxx...x CONTAINS
            ADDRESS yyy...y (zzz...z) ???
```

– General system error messages

```
Format:???  vvvvvvv SYSTEM ERROR:
            ttt...t ???
```

Meaning of variables in the system error messages:

vvvvvvv	Message number
wwwwwwww	Internal task number (TID, task identifier)
xxx...x	Symbolic address of the invalid word
yyy...y	Contents of the invalid word
zzz...z	Error type: NOT ON H-WORD NOT ON F-WORD NOT ON D-WORD NOT 8W-BOUND NOT 64W-BOUND NOT 1K-BOUND NOT 2K-BOUND NOT 4K-BOUND NOT 2**yy B NOT INITIAL ‘D NOT AVAILABLE NOT ALLOCATED NOT ASAM ADDR NOT DEFINED NOT RESIDENT NOT VALID STL NOT VALID ID NOT VALID TID NOT IN CLASS1 NOT IN VM NOT IN SYS.VM NOT IN UVM NOT < MMSIZE NOT >= SYSBASE NOT <= SYSBASE NOT >= SYS-8MB NOT = 16 MB NOT < 16 MB NOT >= 2 MB
ttt...t	Explanatory text

Example

```
/SET-LOGON-PARAMETERS USER-ID=TSOS ——————————————————————————— (1)
/ADD-FILE-LINK LINK-NAME=SODA01,FILE-NAME=DUMP.SETS ————————————— (2)
/START-PROGRAM FROM-FILE=SODA ———————————————————————————————— (3)
TEXT C'BEISPIEL EINER SODA-AUSWERTUNG' ——————————————————————— (4)
BASE SYS=Y, TRACE=Y, MAP=Y ——————————————————————————————————— (5)
MEMORY REAL=(X'3000',X'4FFF'),VIRTUAL=(X'345ABC',X'346124'),CL4=PP ——— (6)
CAT TSN=546 —————————————————————————————————————————————— (7)
FILE CONSLOG ————————————————————————————————————————————— (8)
END ——————————————————————————————————————————————————— (9)
/EXIT-JOB ———————————————————————————————————————————————— (10)
```

(1) Initiates the job.

(2) Assigns dump file DUMP.SETS to SODA.

(3) Loads the SODA routine and executes it.

(4) Writes a header line with explanatory text on each output page.

(5) Outputs system information as well the system trace table and the directory for the Control System modules (EOLDTAB).

(6) Outputs the real address area from address X’3OOO’ to X’4FFF’, the virtual system address area bounded by the limits X’345ABC’ and X’346124’, the VAT and those privileged and nonprivileged pages of the class 4 address space that are only partially allocated.

(7) Prints the catalog for the user ID belonging to TSN 546.

(8) Prints the CONSLOG system file.

(9) Indicates the end of the SODA statements.

(10) Terminates the job.

12.6 Software prerequisites

SODA is dependent on the operating system version. A dump file from a specific BS2000 version cannot be analyzed without the corresponding version of the SODA routine.

The following files and libraries are required for the analysis of a SLED or SNAP file:

SODA for the corresponding BS2000/OSD-BC version

$TSOS.SYSLNK.SODA.xxx
where xxx is the number of the corresponding BS2000/OSD-BC version.

$TSOS.SYSLNK.ANITA
contains modules for the analysis of binder-loader structures

The file $TSOS.SYSLNK.SODA.112 must contain the modules of external SODA routines.

Users in possession of the unbundled product DCM and intending to edit DCM-specific tables must also have access to the file $TSOS.SYSLNK.BCAM.DUMP.

SODA and external SODA routines of BS2000/OSD-BC V2.0 can only run on BS2000 versions ≥ V9.5.

13 SPCCNTRL Checking and managing storage allocations on disk

Version: **SPCCNTRL V11.2A**

The SPCCNTRL routine (SPaCe CoNTRoL) serves to control storage allocations on disk.

The routine is available to nonprivileged users as well as to system administration, although a number of its functions are reserved for system administration (see the descriptions of the individual statements).

13.1 Operating instructions

Before the routine is called, the SPCCNTRL message file should be assigned using the MODIFY-MSG-FILE-ASSIGNMENT command under the TSOS user ID (see "Commands, Volumes 1 - 3" [1], [2], [3]).

```
/MODIFY-MSG-FILE-ASSIGNMENT ADD-FILE=SYSMES.SPCCNTRL.112
```

Alternatively, the file can be assigned subsequently by means of the SPCCNTRL command MODIFY.

The routine can be started in two ways:

```
/START-SPCCNTRL
```

START-SPCCNTRL
CPU-LIMIT = ***JOB-REST** / <integer 1..32767>

The CPU-LIMIT operand is also available for calling the routine. For a description of this operand, see the START-PROGRAM command in the "Commands, Volume 3" manual [3].

```
/START-PROGRAM FROM-FILE=$SPCCNTRL
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are also available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in the "Commands, Volume 3" manual [3].

The control statements can be entered once the program has been called.

13.2 Statements

13.2.1 Overview of all SPCCNTRL statements

Statement	Function
BKPT	Interrupt the SPCCNTRL routine.
CHECK	Perform allocation checks for a public disk and checks for errored entries in the system catalog or VTOC area of a private disk.
DISPLAY	Output to SYSOUT, i.e. display on the screen in interactive mode.
EDT	Call the file editor.
END	Terminate the SPCCNTRL routine.
HELP	Provide supplementary information on one specific statement or output an overview of all SPCCNTRL statements.
LIST	Direct output to SYSLST.
MODIFY	Modify the SPCCNTRL default values.
PURGE	Remove "dead" space on public disks and delete catalog entries from the system catalog and from the VTOC area of a private disk.
TRACE	Locate and output blocks and entries in the system catalog and VTOC area of a private disk, or output any block of a disk.

13.2.2 Description of the statements

In the functions listed below, "filename" must be specified as a fully qualified file name (unless specified otherwise). In certain cases it is also possible to specify a coded file ID in the form X’hhhhhhhh’. If no entry is made for "type", it is assumed that the specified disk is available online.

SPCCNTRL interprets every input that is not a SPCCNTRL statement as a BS2000 command, i.e. in program mode system commands can also be entered and processed.

BKPT
Interrupt SPCCNTRL routine

The BKPT statement interrupts the program (in interactive mode) or results in a wait state (batch mode).

Operation	Operands
{ / BKPT }	

BKPT can be replaced by a slash in column 1 only. If another system command or a SPC-CNTRL statement is entered after the slash, the command or statement is executed before the program is interrupted.

The slash should be used only in interactive mode. In batch mode the slash corresponds to the system command EOF, which serves to terminate the reading of data from SYSDTA. A return to program mode during batch processing is possible only by entering a SEND-MSG command at the console.

CHECK
Perform allocation checks

The following operations can be performed by means of the CHECK statement:

– check the allocation of a public disk
– check for errored entries in the system catalog
– check the contents of the VTOC area (F1 label) of a private disk for errored entries

This statement is reserved for use by system administration.

Operation	Operation
C CHECK	A[LLOCATION],vsn C[ATALOG][,CATID=catid vsn[,typ] ENTRY=filename (filename,JV)] V[TOC],vsn[,typ] [,ENTRY=filename]

A[LLOCATION] The allocation of a public volume is to be subjected to a general check, i.e. as to whether there are any "dead" areas or "doublers".

An overview of the disk allocation is output. The check is aborted if cases of double allocation are detected. Such cases can be localized by means of LIST ALLOCATION. Specification of a catalog ID is not mandatory.

vsn Specifies the public disk that is to be checked.

C[ATALOG] The system catalog is to be checked for errored entries. Error codes are assigned; these can be interrogated by means of HELP.

CATID=catid Specifies the catalog ID of the catalog to be checked. The home catalog is the default value.

vsn[,type] This entry can be made instead of CATID=catid: it enables access to a catalog (not imported) of disk "vsn" of type "type".

ENTRY=filename Specifies a fully qualified file name limiting the check to a single file entry.

=(filename,JV)	The check is to be restricted to the job variable identified by "filename". The "filename" entry can also be made in the form of a coded file ID.
V[TOC]	The contents of the F1 label of a private disk are to be checked for errored entries. This function corresponds to the CHECK CATALOG function.
vsn[,type]	Specifies the private disk whose VTOC area is to be checked. If the disk is not available (online), the disk type ("type") must also be specified.
ENTRY=filename	Restricts the check to a specific file entry with the file name "filename". A code file ID can also be specified.

Examples

1. Output of information to public disk with the following statement:

```
*check allocation,1SBZ.0
 ASSIGNMENT SUMMARY FOR VOLUME 1SBZ.0
 ------------------------------------

 NUMBER OF FILE ENTRIES:             902
 NUMBER OF EXTENTS:                 1369

 PAM PAGES ASSIGNED TO FILES:     216486
 PAM PAGES NOT ASSIGNED:          115185
 PAM PAGES FREE:                       0
*
```

2. Output of information to catalog ID:

```
*CHECK CATALOG,CATID=1SBZ
%  SPC0060 NO ERROR IN THE CATALOG FILE OF PVS 1SBZ
*
```

or

```
CHECK CATALOG,CATID=A
ERROR SUMMARY FOR CATALOG :A:$TSOSCAT
```

ERROR	BLOCK	BYTE	USER	FILENAME
SPC1506	1635	584	TEXACO	DATOUT
SPC1503	1800	403	ULTIMO	PROC03

Tabelle 7:

HELP MSG INFORMATION SPC1506 produces the following information:

```
% SPC1506 VSN IN VOLUME TABLE ENTRY NOT PRINTABLE
```

DISPLAY
Direct output to SYSOUT

The DISPLAY statement outputs the information requested in the operands to SYSOUT. In interactive mode, the information is displayed on the terminal..

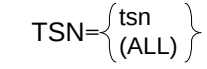
Operation	Operation
{ D / DISPLAY }	{ S[PACE], { P[UBLIC][,vsn][,CATID=catid] / PR[IVATE][,vsn[,typ]] / E[AM][, { TSN={ tsn / (ALL) } / USER=userid / CATID=catid }] / C[ATALOG][, { CATID=catid / vsn[,typ] }] } / C[ATALOG],ENTRY={ filename / (filename,JV) } [,CATID=catid] [,HEX={ N[O] / Y[ES] }] / V[TOC],vsn[,typ] ,ENTRY=filename [,HEX={ N[O] / Y[ES] }] }

S[PACE] The space assigned for memory areas on the disk specified by "vsn" is to be output. This operand is reserved for system administration.

P[UBLIC] The total amount of free and occupied memory space on all public volumes of the home pubset is to be output.

P[UBLIC],vsn Specifies one specific public disk, for which information on the space allocation is requested.

CATID=catid Information is requested on the space allocation of disks that are entered in a specific catalog. If no entry is made, the home catalog is assumed.

PR[IVATE]	The total amount of free and occupied space on all private disks (including labels) is to be output.
PR[IVATE],vsn	Identifies one specific private disk, on which information is requested.
type	Specifies the disk type (e.g. D3468). This entry is required if the private disk is not online.
E[AM]	Information on the system file SYSEAM is to be output. If no entry is made for "tsn" or ALL, an overview of the overall allocation for SYSEAM is output. If "tsn" is specified, the information is restricted to the allocation on the specified TSN. This operand is reserved for system administration.
TSN= 	The information on the system file SYSEAM is to refer solely to the specified TSN ("tsn") or to all TSNs (ALL).
USER=userid	The information on the system file SYSEAM is to refer to the specified user ID.
CATID=catid	This is the catalog ID of the requested catalog. The home catalog is assumed by default if the operand is omitted.
C[ATALOG]	Information on the overall allocation of the system catalog is requested. By specifying a catalog ID CATID=catid or CATALOG,vsn[,type] a specific catalog can be selected. Otherwise the home catalog is always assumed.
vsn[,type]	This entry can be made instead of CATID=catid. It enables access to a catalog (not imported) of disk "vsn" of type "type".
C[ATALOG]	This operand provides supplementary information not supplied by the system command SHOW-FILE-ATTRIBUTES. The operand is available to all users.
ENTRY	
=filename	Output of the catalog entry is requested for the file specified by "filename". "filename" can also be specified in the form X'coded file-id'.
=(filename,JV)	Output of the catalog is requested for the job variable specified by "filename". "filename" can also be specified in the form X'coded file-id'.
CATID=catid	Specifies the catalog ID of the requested catalog. The home catalog is assumed by default if the operand is omitted.
HEX	

=YES	The entire catalog entry is to be output in hexadecimal form.
=NO	It is not to be output in hexadecimal form (default value).
V[TOC]	Output of the information contained in the F1 label on a private volume is requested. The same information is supplied as for DISPLAY CATALOG, with the difference that the VTOC area of the private disk serves as the source.
vsn[,type]	Specifies the volume from whose VTOC area information is requested. If the disk is not online, the disk type ("type") must also be specified.
ENTRY=filename	Specifies the file entry in the F1 label on which information is to be output. "filename" can also be specified in the form of a coded file ID.
HEX={YES / NO}	If YES is specified, the entire entry is output in hexadecimal form.

Examples

1. Output from DISPLAY SPACE,PUBLIC

```
DISPLAY S,PUBLIC
```

VOLUME	DEVICE	PAGES FREE	PAGES USED	FILES
PUBA06	D3470	30333	172578	2372
PUBA04	D3470	25683	177228	2156
PUBA01	D3470	29109	173802	2189
PUBA02	D3470	32955	169956	2176
PUBA03	D3470	29007	173904	2678
PUBA00	D3465	159	61317	1754
PUBA05	D3465	6453	55023	1757
PUBA07	D3468	18141	112101	1757
TOTAL		171840	1095909	16839

2. Output from DISPLAY SPACE,EAM

```
D S,EAM
 SUMMARY FOR :K:$TSOS.SYSEAM:

 SYSEAM SPACE ASSIGNED:              348 PAGES
 SYSEAM SPACE ALLOCATED:           38712 PAGES
 SYSEAM CATALOG ENTRY SIZE:          468 BYTES
```

3. Output from DISPLAY SPACE,EAM,TSN=(ALL)

```
D S,EAM,TSN=(ALL)
```

CATID	TSN	USER	TYPE	FILES	PAGES USED	ALLOCATED
K	5015	TSOS	3	2	6	6
K	5060	A2A3A4A5	3	5	6	6
K	5068	RZDATEX	3	1	32	33
K	5158	TERM	3	1	0	0
K	5167	DATA	3	40	1	3
K	5208	BASIS	3	3	3	6
K	5232	ARCHIV	3	3	0	0
K	5233	CONS	3	2	62	63
K	5239	DELTA	3	6	23	24
K	5328	F1F2F3F4	3	1	5	6
K	5346	BERGER	3	17	4	6
K	5352	FIFO	3	5	45	45

4. Output from DISPLAY SPACE,CATALOG

```
D S,C
 TSOSCAT/HOME

 USERID‘ S IN JOINFILE               69
 EFFECTIVE FILE SIZE               2049
 PAGES USED (4K)    (COUNTER)       279
 PAGES USED (4K)    (BIT TABLE)     279
 PREASSIGNED (ERROR IF NOT 0)         0
*
```

5. Output from DISPLAY CATALOG,ENTRY=TSOSJOIN

```
D C,ENTRY=$TSOS.TSOSJOIN
 PUBLIC DISK FILE:   :A:  $TSOS.TSOSJOIN

 TOTAL SPACE ALLOCATION:            1536 PAGES
 CODED FILE IDENTIFICATION:     A002F585
 FLT LOCK ENTRY:                00000000 C1782F3E A002F585 800F0000
 PAM LOCK ENTRY:                    NONE

 PRIMARY BLOCK NUMBER:                2
 CURRENT BLOCK NUMBER:             2706    CATALOG ENTRY AT BYTE:        17

 CATALOG ENTRY SIZE:               1202 BYTES
 # OF EXTENT ENTRIES:              181
```

EXTENT	VOLUME	LOW	HIGH	PaGES ALLOCATED
1	PUBA00	6025	6048	24
2	PUBA00	23017	23064	48
3	PUBA00	27625	27648	24
4	PUBA00	36409	36432	24
5	PUBA00	36457	36480	24
6	PUBA00	39697	39720	24

```
% SPC0020 SCREEN OWERFLOW, CONTINUE (Y/N) ?
```

6. Output from DISPLAY VTOC,RZWORK,ENTRY=XYZ

```
D V,RZWORK,ENTRY=SPC.TEST.PR
 PRIVATE DISK FILE: :*:$TSOS.SPC.TEST.PR

TOTAL SPACE ALLOCATED:              3 PAGES
CODED FILE IDENTIFICATION:   92032342
FLT LOCK ENTRY:                    NONE
PAM LOCK ENTRY:                    NONE
VTOC LOCK ENTRY:                   NONE
VTOC SHARER ENTRIES:               NONE

PRIMARY BLOCK NUMBER:               2
CURRENT BLOCK NUMBER:            2426        CATALOG ENTRY AT BYTE:      ?
VTOC/F1 ENTRY ON VOLUME:        RZWORK       ON-LINE
VTOC/F1 BLOCK POINTER:              10
VTOC/F1 ENTRY IN BLOCK:             10       CATALOG ENTRY AT BYTE:    1874

CATALOG ENTRY SIZE:                125 BYTES
# OF EXTENT ENTRIES:                 1
```

EXTENT	VOLUME	LOW	HIGH	PaGES ALLOCATED
1	RZWORK	2878	2880	3

EDT
Call EDT

This statement calls EDT in full screen mode. Switching to SPCCNTRL program mode is possible by means of the HALT statement or by pressing the K1 key. If @E was entered to switch to EDT line mode, it is also possible to return to SPCCNTRL by entering the @RETURN or @HALT statement. This does not cause the EDT virtual file to be destroyed. The EDT statements @SYS and @RUN are rejected.

Operation	Operands
EDT	

END
Terminate SPCCNTRL

This statement terminates SPCCNTRL.

Operation	Operands
{ E END }	

If E is entered followed by an operand, SPCCNTRL assumes the ENTER-JOB command is meant and executes this command.

HELP
Describe specified statement or list all SPCCNTRL statements

Operation	Operation
{ H / HELP }	[stmt[,mode]]

stmt — Specifies the name of an SPCCNTRL statement; a description of this statement is to be output. If "stmt" is not specified, an overview of all statements is output.

mode — Specifies one of the following keywords of the statement designated by "stmt":

```
A[LLOCATION]
S[PACE]
C[ATALOG]
V[TOC]
```

A description of the specified function is output. If no operand is specified, an overview of the available statement function modes is output.

If the HELP statement is specified in unabbreviated form and an invalid operand is specified for "stmt" instead of an SPCCNTRL statement, the system command HELP-MSG-INFORMATION is executed.

LIST
Direct output to SYSLST

The LIST statement enables information to be written to SYSLST. This makes it useful for redirecting information which is too extensive to be output on the terminal screen.

This statement is reserved for system administration.

Operation	Operands
{ L | LIST }	{ A[LLOCATION],vsn[,type][,USER=userid] | C[ATALOG][,{ CATID=catid | vsn[,type] }][,USER=userid] | V[TOC],vsn[,type] [,ENTRY=filename] }

A[LLOCATION] The general allocation information for a volume is to be output.

The list sorts the allocation information according to extent, the start of each extent being specified in the form "cchhr".

Information can only be output on volumes that are identified by their volume serial number. When specifying operands it is not necessary to distinguish between public and private volumes.

The following comments may occur in the output list:

NO VTOC REF.	-	No entry in F1 label for this area.
RESERVED	-	Space allocated, but no file name present.
OVERLAY!	-	Space allocated twice.
NOT ASSIGNED	-	Area not assigned.
NO REFERENCE	-	Allocator requirements missing or an error was indicated by CMS.
FREE SPACE	-	Free area
SPACE PROBLEM	-	Bad area

vsn[,type] Specifies the disk for which allocation information is to be listed. If a private disk is not online, the disk type "type" must also be specified. In the case of public disks, both free and "dead" space are listed.

	"Dead" space is an area which the entry in the F5 label identifies as being allocated but for which there is no file in the system catalog.
USER=userid	Specifying a user ID limits output. In this case no information on free or dead space is output, even for public volumes.
C[ATALOG]	A list of the allocation of the system catalog is to be output. The occupied blocks are listed for each user in the order in which they were chained.
CATID=catid	Specifies the catalog ID of the catalog on which information is requested. The home catalog is the default value.
vsn[,type]	This entry can be made instead of CATID=catid. It enables access to a catalog (not imported) on disk "vsn" of type "type".
USER=userid	Specifying a "userid" limits the output information to this user ID.
V[TOC]	This specification requests information from the F1 label of a private disk.
vsn[,type]	Specifies the volume serial number of the disk from whose VTOC area information is requested. If the volume is not online, the disk type "type" must also be specified.
ENTRY=filename	This specification limits output to the file entry "filename". Fully qualified file names are not prohibited, but only the specification of partially qualified file names is practical.

Examples

1. Output of LIST ALLOCATION,PUB

```
S P C C N T R L    VERSION  11.2A70 (112)      DATE: 1995-01-26  TIME: 09:47:17          PAGE     1
COMMAND IN EXECUTION:   LIST ALLOCATION,1SBZ.0
     EXTENTS ON VOLUME 1SBZ.0
    CCHHR (LOW) ||LOW   |HIGH || ASSIGNMENT| PAM PAGES ||   USERID   |  FILENAME
 ---------------||------|------||------------|-----------||------------|-------------------------------
  00000-00-01   ||     1|    21|| RESERVED   |        21 ||            |
  00000-01-04   ||    22| 2070||             |      2049 ||  TSOS      |  CONVCAT
  00005-05-16   || 2071| 6168||             |      4098 ||  TSOS      |  TSOSCAT
  00015-15-04   || 6169| 6192||             |        24 ||  TSOS      |  GPSP.LNK
  00015-16-10   || 6193| 6216||             |        24 ||  TSOS      |  H900250
  00015-17-16   || 6217| 6321||             |       105 ||  TSOS      |  LOR
  00016-01-16   || 6322| 6336||             |        15 ||  TSOS      |  MACROLIB
  00016-02-13   || 6337| 6561||             |       225 ||  TSOS      |  ADAM08C1
  00016-15-04   || 6562| 6576||             |        15 ||  TSOS      |  ARCHIVE
  00016-16-01   || 6577| 6600||             |        24 ||  UTM3      |  SYSLNK.KDCMON.112
  00016-17-07   || 6601| 6624||             |        24 ||  TSOS      |  SIPLIB.XHCS-SYS.010
  00016-18-13   || 6625| 6648||             |        24 ||  TSOS      |  SYSDAT.PRO.021.SIECC1.SEM
  00016-20-01   || 6649| 6720||             |        72 ||  TSOS      |  SYSDOC.UTM.033.READ-ME.D
  00017-02-04   || 6721| 6978||             |       258 ||  TSOS      |  ATSSPVS.LIB
  00017-16-10   || 6979| 6981||             |         3 ||  TSOS      |  GPSP.LNK
  00017-16-13   || 6982| 6984||             |         3 ||  TSOS      |  HPFILE
  00017-16-16   || 6985| 7104||             |       120 ||  TSOS      |  FACT
  00018-01-13   || 7105| 7968||             |       864 ||  TSOS      |  COBOL85
  00020-06-01   || 7969| 8052||             |        84 ||  TSOS      |  C700315
  00020-10-13   || 8053| 8064||             |        12 ||  TSOS      |  H12008C1
  00020-11-07   || 8065| 9405||             |      1341 ||  TSOS      |  HPFILE
  00023-20-07   || 9406| 9537||             |       132 ||  TSOS      |  H10098C0
  00024-05-16   || 9538| 9597||             |        60 ||  TSOS      |  H130023S
  00024-09-04   || 9598| 9600||             |         3 ||  TSOS      |  H130231S
  00024-09-07   || 9601| 9819||             |       219 ||  TSOS      |  H12008C1
  00024-21-10   || 9820| 9981||             |       162 ||  TSOS      |  H900250
  00025-08-13   || 9982| 9984||             |         3 ||  TSOS      |  IFG
  00025-08-16   || 9985|10248||             |       264 ||  TSOS      |  H130023S
  00026-01-13   ||10249|10272||             |        24 ||  TSOS      |  SYSFGM.FHS.081.E
  00026-03-01   ||10273|10296||             |        24 ||  TSOS      |  SYSLNK.BCAM.120.COS
  00026-04-07   ||10297|10320||             |        24 ||  TSOS      |  SYSLNK.LLMAID.010
  00026-05-13   ||10321|10344||             |        24 ||  TSOS      |  SYSLNK.PRO.021.C.LIBDUMMY
  00026-07-01   ||10345|10368||             |        24 ||  TSOS      |  SYSLNK.PROP-XT.010
  00026-08-07   ||10369|10584||             |       216 ||  TSOS      |  H130231S
  00026-20-07   ||10585|10752||             |       168 ||  TSOS      |  SYSLIB.TV-NTAC2E.051.IAC2
  00027-07-16   ||10753|13776||             |      3024 ||  TSOS      |  SINLIB.POSIX-SH.010
  00035-01-04   ||13777|13803||             |        27 ||  TSOS      |  SYSLIB.XAF.020
  00035-02-13   ||13804|13818||             |        15 ||  TSOS      |  SYSLNK.ADAM.112
  00035-03-10   ||13819|13824||             |         6 ||  TSOS      |  SYSLNK.ANITA.112
  00035-03-16   ||13825|14019||             |       195 ||  TSOS      |  SIPLIB.SPSERVE.020
  00035-14-13   ||14020|14022||             |         3 ||  K52       |  TLS.SYSENT.MAREN.065.MARENUCP
  00035-14-16   ||14023|14031||             |         9 ||  K52       |  TLS.SYSFGM.MAREN.065.E
  00035-15-07   ||14032|14058||             |        27 ||  TSOS      |  SYSFHS.IMON.010.D
  00035-16-16   ||14059|14061||             |         3 ||  TSOS      |  Z14.D
  00035-17-01   ||14062|14064||             |         3 ||  K52       |  TLS.REP.GA
  00035-17-04   ||14065|14067||             |         3 ||  FASTPAM    |  TA-DFP.T50.GR10
  00035-17-07   ||14068|14070||             |         3 ||  TSOS      |  SYSNRF.UTM.034
  00035-17-10   ||14071|14073||             |         3 ||  TSOS      |  S.FT-BS2.26870024.LST
```

Note

For 3436 and 3437 disks, there is no entry in the column CCHHR (LOW).

2. Output of LIST ALLOCATION,V2A.00,D3480

```
S P C C N T R L    VERSION  11.2A70 (112)      DATE: 1995-01-26  TIME: 09:50:18          PAGE     1
COMMAND IN EXECUTION:   LIST ALLOCATION,V2A.00,D3480
EXTENTS ON VOLUME V2A.00

   CCHHR (LOW) ||LOW   |HIGH || ASSIGNMENT| PAM PAGES ||   USERID   |  FILENAME
 --------------||-----|-----||-----------|-----------||------------|-------------------------------
   0000-00-01  ||    1|   18|| RESERVED  |       18  ||            |
   0000-18-01  ||   19| 4116||            |     4098  ||  TSOS      |  TSOSCAT
   0016-36-01  || 4117| 4128||            |       12  ||  TSOS      |  TSOSJOIN
   0016-48-01  || 4129| 4131||            |        3  ||  TSOS      |  SYSMSA.SDF-P-BASYS.010
   0016-51-01  || 4132| 4146||            |       15  ||  TSOS      |  SYS.AIDIT0
   0016-66-01  || 4147| 4152||            |        6  ||  TSOS      |  REPFILE.AIDSTART
   0016-72-01  || 4153| 4176||            |       24  ||  TSOS      |  SYSSDF.AIDSYS.100.SYSTEM
   0016-96-01  || 4177| 4200||            |       24  ||  TSOS      |  SYSSDF.AIDSYS.100.SYSAUDIT
   0016-20-01  || 4201| 4224||            |       24  ||  TSOS      |  SYSSDF.AIDSYS.100.SYSPRIV
   0016-44-01  || 4225| 4227||            |        3  ||  TSOS      |  DO.MSGLIB.5
   0016-47-01  || 4228| 4230||            |        3  ||  TSOS      |  DO.MSGLIB.1
   0016-50-01  || 4231| 4233||            |        3  ||  TSOS      |  SYSSSD.AIDSYS.100
   0016-53-01  || 4234| 4245||            |       12  ||  TSOS      |  SYSLIB.ARCHIVE.027
   0016-65-01  || 4246| 4248||            |        3  ||  TSOS      |  SYSSSD.ARCHIVE.027.CL4
   0016-68-01  || 4249| 4272||            |       24  ||  TSOS      |  ADIAG
    016-92-01  || 4273| 4275||            |        3  ||  TSOS      |  UGENINPUT.DSSMCAT.VM3
    016-95-01  || 4276| 4278||            |        3  ||  TSOS      |  SYSREP.PLAM.020
    016-98-01  || 4279| 4281||            |        3  ||  TSOS      |  ASSXTC
   0016-10-01  || 4291| 4293||            |        3  ||  TSOS      |  PSE.CONSOLEN
   0016-13-01  || 4294| 4296||            |        3  ||  TSOS      |  SYSMSV.ASSEMBH.010.LG
   0016-16-01  || 4297| 4305||            |        9  ||  TSOS      |  SYSMSA.ASSEMBH.010.LG
```

3. Output of LIST CATALOG

```
S P C C N T R L   VERSION 11.2A80 (112)
COMMAND IN EXECUTION: LIST CATALOG,CATID=A,USER=TSOS
SUMMARY OF CONTENTS IN :A:TSOS.TSOSCAT
FOR USERID TSOS
```

USERID	BLK COUNT	LBN(DEC)	LBN(HEX)	#ENTRIES	UNUSED	BLK COUNT	LBN(DEC)	LBN(HEX)	#ENTRIES	UNUSED
TSOS	1	2	0002	12	564	1	1492	05D4	9	175
	2	3	0003	16	54					
	3	4	0004	15	118					
	4	7	0007	16	4					
	5	8	0008	15	96					
	6	9	0009	16	4					
	7	10	000A	16	6					
	8	11	000B	15	112					
	9	12	000C	16	3					
	10	15	000F	16	28					
	11	16	0010	16	24					
	12	17	0011	16	11					
	13	18	0012	15	2					
	14	19	0013	16	23					
	15	20	0014	16	0					
	16	21	0015	16	13					
	17	22	0016	15	52					
	18	38	0026	14	53					
	19	39	0027	16	22					
	20	40	0028	16	29					
	21	41	0029	16	65					
	22	50	0032	15	110					
	23	53	0035	16	38					
	24	54	0036	15	109					
	25	57	0039	15	79					
	26	60	003C	15	39					
	27	63	003F	15	36					
	28	66	00A2	16	26					
	29	67	00A3	15	69					
	30	68	00A4	16	8					
	31	192	00C0	16	4					

USERID	BLK COUNT	LBN(DEC)	LBN(HEX)	#ENTRIES	UNUSED	BLK COUNT	LBN(DEC)	LBN(HEX)	#ENTRIES	UNUSED
	32	195	003C	15	38					
	33	310	0136	15	44					
	34	376	0178	15	82					
	35	746	02EA	15	4					
	36	763	0300	15	99					
	37	777	0309	15	8					
	38	987	03DB	15	67					
	39	1000	03E3	15	109					
	40	1001	03E9	15	30					
	41	1005	03ED	16	14					
	42	1006	03EE	16	13					
	43	1007	03EF	16	11					
	44	1008	03F0	15	53					

MODIFY
Modify default values

The MODIFY statement enables SPCCNTRL default values to be modified if other values are desired. Any such modification applies only to the current program run.

This statement can also be used for the subsequent assignment of the SPCCNTRL message file or of any other message file. This is possible only under the TSOS user ID. The message file remains in the system until the end of the session, unless the assignment is changed by means of the system command MODIFY-MSG-FILE-ASSIGNMENT.

Operation	Operands
{ M / MODIFY }	{ LINES=nn / ,MSG=[{ SPC[CNTRL] / (FILE,filename) }] }

LINES=nn — Specifies the desired number ("nn") of lines per page for the list output (where $5 \leq nn \leq 127$). If no value is specified, the default value is 60.

MSG

=SPCCNTRL — TSOS only
Specifies that the message file valid for SPCCNTRL is to be assigned. The expected file name is SYSMES.SPCCNTRL.version where "version" is the version number of the SPCCNTRL routine used.

=(FILE,filename) — Messages from the file specified here by "filename" are to be dynamically loaded.

PURGE
Remove dead space and delete catalog entries

The PURGE statement can be used to remove dead space from public volumes and to delete catalog entries from the system catalog or the VTOC area of private disks. It can also be used to remove unused space in individual files.

Operation	Operands
{ P PURGE }	{ S[PACE],{ P[UBLIC][,{ vsn CATID=catid }] PR[IVATE][,vsn[,type]] } [,USER=userid] A[LLOCATION],vsn C[ATALOG] [,CATID=catid], ENTRY={ filename (filename,JV) } V[TOC],vsn[,type],ENTRY=filename }

S[PACE] Release unused space in the individual user files.
The assigned buffer sizes are taken into account, i.e. only as much space is released as will enable the file in question to be processed without errors. This function is available to all users.
Password-protected files are not affected.

P[UBLIC] Unused space on public volumes is to be released.

vsn Specifies the volume serial number of a specific public volume on which unused space is to be released.

CATID=catid Specifies the catalog ID of the catalog from which entries are to be deleted.
If neither vsn nor CATID= is specified, the default catalog ID of the user ID specified via USER= is assumed.

P[RIVATE] Unused space on private volumes is to be released. Nonprivileged users can only process files under their own user ID and which also have an entry in the system catalog.
For nonprivileged users both "vsn" and "type" must be specified; under TSOS only the VSN need be specified.

vsn[,type] Specifies the disk on which space is to be released. If it is not available (online), the disk type "type" must also be specified.

USER=userid Specifies the user ID whose files are to be processed. If no entry is made and SPCCNTRL is running under the TSOS user ID, the files of all users except TSOS are processed by
PURGE SPACE,PUBLIC.

	PURGE SPACE,PRIVATE processes the files of all users (including TSOS) if USER is not specified and SPCCNTRL is running under TSOS.
A[LLOCATION]	The allocation of space on a public volume is to be reorganized, i.e. dead space is to be removed. This does not happen in the same SPCCNTRL run, however; instead, the run has to be terminated and an IMPORT-CATID command entered first. Dead space may have been created by system errors or explicitly created by means of the statement: PURGE CATALOG,ENTRY=. The A[LLOCATION] operand is reserved for system administration.
vsn	Specifies the volume serial number of a public volume to be cleaned up.
C[ATALOG]	An entry is to be removed from the system catalog. This is useful in the case of errored entries which are no longer to be accessed.
CATID=catid	Specifies the catalog from which an entry is to be deleted. If this operand is omitted, the home catalog is assumed.
ENTRY=filename	Specifies the name of the file whose catalog entry is to be deleted.
=(filename,JV)	Specifies the name of the job variable whose catalog entry is to be deleted.
V[TOC]	An entry is to be deleted from the VTOC area (F1 label) of a private volume. This function corresponds to the statement PURGE CATALOG and is reserved for system administration.
vsn[,type]	Specifies the disk containing the VTOC area from which an entry is to be deleted. If the volume is not available (online), the disk type "type" must also be specified.
ENTRY=filename	Specifies the entry to be deleted.

Note

The operands SPACE,... and ALLOCATION,... are invalid for the 3410 External High-Speed Storage Unit.

Example

Output from PURGE SPACE,PUBLIC

```
PURGE S,PUBLIC
% SPC0025 PURGE PROCESSING COMPLETED
NUMBER OF FILES SELECTED FOR PURGE:     1062 ————————————————————————— (1)
NUMBER OF ERRORS DURING DEALLOCATION:    121 ————————————————————————— (2)
AMOUNTED OF PAM PAGES DEALLOCATED:     20787 ————————————————————————— (3)
```

Explanation

(1) Number of files found to contain purgeable memory.

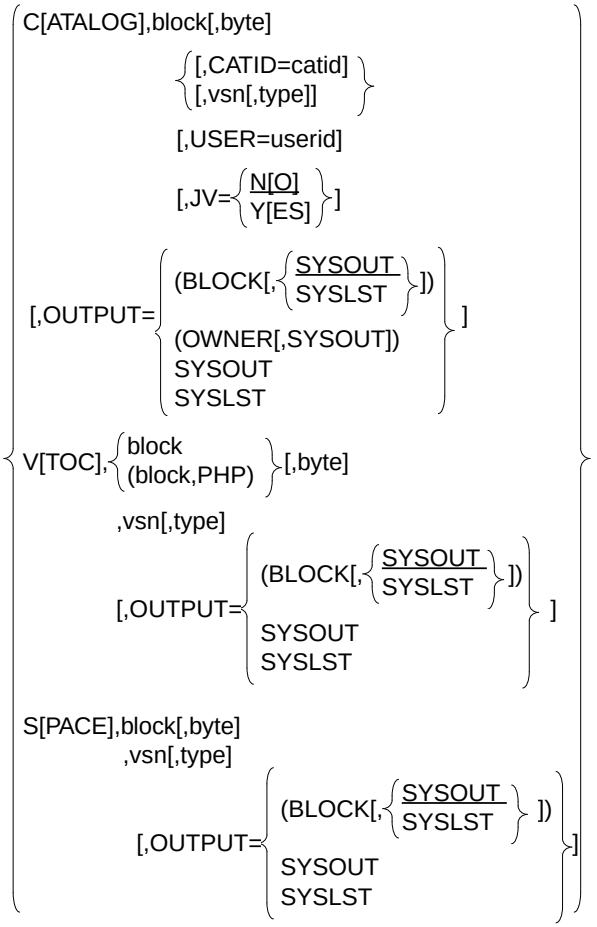
(2) Number of files which could not be purged.

(3) Total number of PAM pages released by the purge.

TRACE
Identify and output blocks and entries in system catalog and VTOC area of private disk

The TRACE statement enables blocks and entries in the system catalog and the VTOC area of a private disk to be identified and output to SYSOUT or SYSLST. In addition, any block of a public or private disk can also be displayed. In order to save write operations, the default values for the keyword operands remain valid until they are modified by a subsequent entry.

This statement is reserved for system administration.

Operation	Operands
{ T / TRACE }	

C[ATALOG] Information from the system catalog is to be supplied. Any block, including the control blocks with the bit tables, can be displayed, even if it is not assigned to a user.

block Specifies the number of the block to be output. This entry can also be made in hexadecimal form (X'hhhh'). If a user ID is specified under USER=userid, the block specification is relative to the user's catalog chain; otherwise "block" is interpreted as an absolute block number.
Together with OUTPUT=(BLOCK,SYSLST), "block" can be omitted. In this case the complete catalog is output.

byte Specifies the byte number within the block from which output is to start. This entry can also be made in hexadecimal form (X'hhhh').

CATID=catid Specifies the catalog ID of the desired catalog. The default value is the home catalog.

vsn[,type] This entry can be made instead of CATID=catid. It enables access to a catalog (not imported) on disk "vsn" of type "type".

USER=userid Specifies the user ID of the user from whose catalog entry blocks are to be output.

JV=YES Blocks from a job variable chain are to be displayed. This entry is practical only in conjunction with USER=userid. It is ignored in all other cases.

=NO Job variables are to be ignored. A block from the user's file name chain is output.

OUTPUT

=(BLOCK,SYSOUT)
The information is to displayed on the terminal screen. SYSOUT is the default value.

=(BLOCK,SYSLST)
Output is to be directed to printer.

=(OWNER,SYSOUT)
Specifiesthe owner of a block. The user ID to which "block" belongs is output. This output is possible only via SYSOUT.

V[TOC] Information is to be output from the VTOC area of a private disk.

block Specifies the number of the block to be output. This entry can also be made in hexadecimal form (X'hhhh').
Together with OUTPUT=(BLOCK,SYSLST), "block" can be omitted. In this case the entire VTOC area is output.

PHP — Specifies a physical block number, i.e. the block number is not relative to the beginning of the VTOC area.

byte — Specifies the byte number within the block from which output is to start. This entry can also be made in hexadecimal form (X'hhhh').

vsn[,type] — Specifies the volume serial number of the private disk whose VTOC area is to be output. If the disk is not available (online), the disk type "type" must also be specified.

OUTPUT

=(BLOCK,SYSOUT)
The information is to be displayed on the terminal screen.

=(BLOCK,SYSLST)
Output is to be directed to printer.

S[PACE] — A freely selectable item of information from the specified disk is to be output.

block — Specifies the number of the block to be output. This entry can also be made in hexadecimal form (X'hhhh'). Together with OUTPUT=SYSLST, the "block" specification can be omitted, in which case the entire SVL together with the F5 label is output.

byte — Specifies the byte number within the block from which output is to start. This entry can also be made in hexadecimal form (X'hhhh').

vsn[,type] — Specifies the volume serial number of the public or private disk on which information is to be output. If the disk is not available (online), the disk type "type" must also be specified.

OUTPUT

=(BLOCK,SYSOUT)
The information is to be displayed on the terminal screen. If no entry is made, the default value is SYSOUT.

=(BLOCK,SYSLST)
Output is to be directed to printer.

Example

```
TRACE CATALOG,1,10,CATID=JA
```

:A:TSOSCAT	LHP: 1 (0001)		PHP: 10 ON VOLUME PUBA00	
0009 (000):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0019 (010):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0029 (020):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0039 (030):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0049 (040):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0059 (050):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0069 (060):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0079 (070):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0089 (080):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0099 (090):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
00A9 (0A0):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
00B9 (0B0):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
00C9 (0C0):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
00D9 (0D0):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
00E9 (0E0):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
00F9 (0F0):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0109 (100):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0119 (110):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0129 (120):	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF

```
% SPC0020 SCREEN OVERFLOW, CONTINUE (Y/N) ?
```

14 TPCOMP2 Comparison of data on two magnetic tapes

Version: **TPCOMP2 V11.2A**

The TPCOMP2 utility routine serves to compare data recorded on two magnetic tapes. It provides a listing of all portions of these tapes that are not identical.

Comparison of data is done on a decade basis. This allows for the printing of 5 groups of variant data in one line on a 132-character printer or 6 groups per line on a 160-character printer.

14.1 Input

Input to the tape compare routine consists of 2 magnetic tapes recorded in the same format.

This routine requires no parameters for its standard functions. For other options, parameters are entered via SYSDTA.

14.2 Output

Output is a printed listing of any discrepancies that exist between the tapes being compared.

14.3 Functions

In addition to the standard functions listed below, this routine provides other functions which can be selected by specifying parameters (see “Optional functions” below).

14.3.1 Standard functions

TPCOMP2 has a preset option which provides the following standard functions for tapes:

1) Rewind both tapes to BOT at the start and end of the comparison run.

2) Display differences in both hexadecimal and EBCDIC format on a 132-character printer.

3) Terminate the comparison if a double tape mark is sensed.
(If a double tape mark for one tape is sensed, and the other tape is not also positioned at a double tape mark, the remaining data on the second tape is printed until the double tape mark is encountered.)

14.3.2 Optional functions

By the use of parameters (entered via SYSDTA), the following options can be selected:

1) Positioning of tapes in a forward or reverse direction, based on a count of tape marks or blocks.

2) Terminating the comparison on the basis of a given number of tape marks or blocks.

3) Designating a 160-character printer; specifying print formats of hexadecimal, EBCDIC, or a combination of both.

4) Processing of unblocked variable-length records or blocked fixed-length records.

14.4 Starting the program run

The $TPCOMP2 utility routine is started by means of the following command:

```
/START-PROGRAM $TPCOMP2
```

The following messages appear and you are requested to input statements:

```
%  BLS0500 PROGRAM 'TPCOMP2', VERSION '11.2A' OF '1994-12-12' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993. ALL
RIGHTS RESERVED
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in the "Commands, Volume 3" manual [3].

14.5 Statements

14.5.1 Overview of the TPCOMP2 statements

Operation	Operands	Function
COM	stcnnnnn	Compares certain areas of the tape and specifies the print line length and printer mode
END		Terminates statement input
LIM	nnnnn	Terminates comparison after the specified number of blocks
POS1 POS2	pppnnnnn pppnnnnn	Positions the magnetic tapes
RCD	t␣nnnnn	Specifies the record format and (for fixed-length records) the record length
STP		Stops program execution when a mismatch is detected

14.5.2 Descriptions of the statements

COM statement

This statement specifies that only certain areas of the input tapes are to be compared. It can also be used to specify the print line length and to select the print format.

Operation	Operands
COM	stcnnnnn

s	s=0	132-character print line (default value)
	s=1	160-character print line
t	Print format:	
	t=H	Hexadecimal
	t=G	EBCDIC
	t=C	Hexadecimal and EBCDIC format (default value)
c	c=T	Terminate after reading nnnnn tape marks on both tapes.
	c=B	Terminate after reading nnnnn blocks on both tapes.
nnnnn	Decimal number of tape marks or blocks (with leading zeros).	

Examples

```
COM␣OCT00006
COM␣1HB00672
```

Note

After the operation specified by a COM statement has been completed, the routine will accept a POS statement, if issued. At this point, the comparison process can be terminated by means of an END statement, or the user can call for further comparisons of the input tapes by entering additional COM and POS statements.

Example

```
COM      POS      COM      POS      COM      COM      POS
END      COM      COM      COM      POS      COM      POS
POS      COM      COM      COM      POS      END
END      END      END      POS      END
                           END
```

If two consecutive POS statements are used, the comparison process is carried out using default values.

END statement

This statement indicates the end of parameter input.

Operation	Operands
END	

LIM statement

With this statement, tape comparison is terminated on both tapes after a certain number of blocks.

Operation	Operands
LIM	nnnnn

nnnnn Number of blocks on the two tapes to be compared. If this operand is omitted, all blocks are compared.

POS statement

This statement positions the magnetic tapes.

Operation	Operands
POS1 POS2	pppnnnnn pppnnnnn

The POS2 statement with its operands is needed only when both magnetic tapes are to be positioned.

ppp	ppp=FTM	Read forward nnnnn tape marks.
	ppp=RTM	Read reverse nnnnn tape marks.
	ppp=FBK	Read forward nnnnn blocks.
	ppp=RBK	Read reverse nnnnn blocks.
	ppp=BOT	Rewind to BOT.

nnnnn Decimal number of tape marks or blocks (with leading zeros).

Examples

```
POS1␣FTM00002
POS2␣FBK00333
POS1␣FBK00017,POS2␣FBK00017
```

RCD statement

This statement signifies that input records are variable in length or that blocked, fixed-length records are to be processed on a logical record level.

Operation	Operands
RCD	t␣nnnnn

t
- t=F Fixed-length records
- t=V Variable-length records

nnnnn Fixed-length record size (with leading zeros). This entry does not apply to variable-length records.

Note

This statement is required when comparing variable-length records or blocked, fixed-length records.

STP statement

This statement is used to stop TPCOMP2 when data discrepancies are encountered during comparison of the data on two magnetic tapes.

Operation	Operands
STP	

This statement must be repeated before each COM statement or before two consecutive POS statements.

14.6 Use of the ADD-FILE-LINK command

Prior to calling TPCOMP2, the user must define a file link name for each tape file to be compared (ADD-FILE-LINK command): LINK-NAME=COM001 for the first tape, LINK-NAME=COM002 for the second tape.

For noncataloged tape files, a catalog entry must first be generated by means of the IMPORT-FILE command.

The following parameters of the required commands can be specified:

Cataloged tape file:

```
ADD-FILE-LINK   LINK-NAME=COM00X
                ,FILE-NAME=filename
                [,SUPPORT=*TAPE(VOLUME-LIST=*CATALOG(VOL-SEQ-NUm=n))]
```

Noncataloged tape file:

```
IMPORT-FILE     SUPPORT=*TAPE
                (VOLUME=vsn
                ,DEVICE-TYPE=TXXXX
                ,FILE-NAME=filename
                [,PREMOUNT-LIST=n])
ADD-FILE-LINK   LINK-NAME=COM00X
                ,FILE-NAME=filename
```

In addition, the following operands may be specified in the ADD-FILE-LINK command:

```
,SUPPORT=*TAPE(LABEL-PROCESSNG=*PAR(LABEL-TYPE=...)).
```

When comparing two master tapes, LABEL-TYPE=*NON-STD must be entered. The type of labeling has to be specified via the program parameters in this case.

Example 1

The preset compare option is to be applied to a cataloged and a noncataloged tape file.

```
/ADD-FILE-LINK LINK-NAME=COM001,FILE-NAME=FILE1
/IMPORT-FILE SUPPORT=*TAPE(VOLUME=BAND1,DEVICE-TYPE=T1600,FILE-NAME=FILE2)
/ADD-FILE-LINK LINK-NAME=COM002,FILE-NAME=FILE2
/START-PROGRAM FROM-FILE=$TPCOMP2
END
```

Example 2

Data located between the 4th and 5th tape marks on one magnetic tape is to be compared with data between the first 2 tape marks on another magnetic tape. Any discrepancies are to be output in EBCDIC mode on a 160-character printer. The tape files to be compared have both been cataloged.

```
/ADD-FILE-LINK LINK-NAME=COM001,FILE-NAME=FILE1
/ADD-FILE-LINK LINK-NAME=COM002,FILE-NAME=FILE2
/START-PROGRAM FROM=FILE=$TPCOMP2
POS1 FTM00004,POS2 FTM00001
COM  1GT00001
END
```

14.7 TPCOMP2 messages

If an unusual condition is encountered, one of the following messages is output to SYSOUT:

```
2801A PARAMETER ERROR
```

Meaning
Invalid format.

Response
Terminate.

```
2802 ERROR READING PARAMETER CARD
```

Meaning
Self-explanatory.

Response
Terminate.

```
2803 RECORD TRUNCATED FILE zzz BLK yyyyy IPT x
```

Meaning
Block read is greater than block size specified.

Response
Terminate.

```
2807
```

Meaning
Illegal input device.

Response
Terminate.

2809A BOT OR DOUBLE TAPE MARK WHILE POSITIONING IPT

Meaning
While positioning input tape, double tape mark or EOT was sensed before specified block count was reached.

Response
Continue. Tape will rewind and restart in the case of initial positioning. In the case of final positioning, final printout will occur.

2810 UNMATCHED TM ON IPTx

Meaning
No suitable tape mark found on input tape (x=1/2).

2810A DBL TM ON IPT

Meaning
Double tape mark sensed during comparison before specified block count is reached.

Response
Continue. Remaining tape will be printed.

2821A SHORT RECORD

Meaning
Record length is less than fixed length specified.

Response
Continue.

2830 yyyyy BLOCK IPTx READ ERROR

Meaning
Unrecoverable read error in block yyyyy of input tape, where x=1 stands for 1st tape and x=2 for 2nd tape.

Response
Block yyyyy is compared with corresponding block of other tape. Processing continues.

2800 VERSION xx OF TAPE COMPARE READY

Meaning
Program is loaded. "xx" = version number.

Response
Processing continues.

```
2835 OPEN ERROR xy CODE=xxxx COMPARE TERMINATED
```

Meaning
OPEN error for input tape "xy". "xxxx" = DMS error code.

Response
Terminate.

```
2836 CLOSE ERROR xy CODE=xxxx COMPARE TERMINATED
```

Meaning
CLOSE error for input tape "xy". "xxxx" = DMS error code.

Response
Terminate.

```
2840 BTAM ERROR CODE=xxxx
```

Meaning
I/O error. "xxxx" = DMS error code.

Response
Terminate.

```
2870 NO CHAINING,OPEN ERROR INPUT,CODE=xxxx
```

Meaning
Page chaining required for specified block size.

15 VOLIN
Initialization of disk storage units

Version: **VOLIN V11.2A**

VOLIN (VOLume INitializer) is a utility routine that prepares a disk pack so that it can be used as a new (virgin) volume in BS2000. Whether the prepared volume is actually new or has already been used in BS2000 or in another operating system (e.g. BS1000) is not important. All that matters is that the disk type is supported by BS2000. A list of disk types supported by BS2000 appears in the table at the end of this description.

The VOLIN utility routine has four basic functions:

1. Volume formatting.
   The volume receives the block structure required by BS2000.
2. Volume checking.
   The disk pack is checked for errors by writing and reading a test pattern. Alternate tracks are assigned to any defective tracks.
3. Label generation.
   The blocks reserved by BS2000 for labels are supplied with initial values.
4. Track repair
   A fully initialized CKD disk pack undergoes track check and repair, limited to one or more tracks. The track contents are restored as far as possible.

Functions 1 and 2, volume formatting and checking, are linked. They are always followed by label generation.
Function 3, label generation, can be performed separately using the statement FMT=NO. This presupposes that the volume has already been used in BS2000.
Function 4, track repair, deals only with the selected tracks. The DEFECTS statement is used to select the tracks and the statement REPAIR=YES is used to repair them. Other tracks remain unchanged.
The repair function offered by VOLIN can only be used under the BS2000 operating system, but not under SIROS or in the SIR-IP.

In addition to the basic functions described, VOLIN allows the mode of use to be defined for a disk pack. The Data Management System uses this to determine whether the access functions implied by the presence of a PAM key are permitted.

If the usage mode is non-key (NK), only NK access methods are allowed. If the usage mode is changed, the disk surface may need reformatting and the capacity values may change depending on disk type. The relevant information is recorded in the list of types.

VOLIN can be used to initialize volumes that cannot be used under BS2000 versions older than the version used to run VOLIN.

As of BS2000 V10, the volume types in question are:
data volumes of usage type NK (no PAM key).

As of BS2000OSD-BC V1.0, the volume types in question are:

- volumes with NK4 format
- volumes with the new F5 label layout due to
  - a change in the smallest allocation unit: 8 or 64 Kbytes instead of 6 Kbytes
  - the hard disk being much larger than those generally employed to date.

A number of new volume types applicable only to public volumes and not to private volumes have been introduced with BS2000/OSD-BC V1.0.
The volume type is defined by the following characteristics:

- DMS PAM key characteristic, i.e. usage type (PK): key (K) / no key (NK)
- smallest allocation unit (AU), i.e. size of the smallest file or more precisely, the smallest unit of memory: 6 Kbytes / 8 Kbytes / 64 Kbytes.
- smallest transfer unit (TU), i.e. smallest unit of transfer between the disk and main memory: 2 Kbytes / 4 Kbytes.

Combinations of these characteristics defined with VOLIN yield the following volume types available under BS2000/OSD-BC V1.0:

PK	AU	TU	Abbreviation	Supported as of	For disk type
K	6K	2K	K disk	< BS2000 V10.0	Private, public
NK	6K	2K	NK2 disk	BS2000 V10.0	Private, public
NK	8K	2K	NK2(8K) disk	BS2000/OSD-BC V1.0	Public
NK	64K	2K	NK2(64K) disk	BS2000/OSD-BC V1.0	Public
NK	8K	4K	NK4(8K) disk	BS2000/OSD-BC V1.0	Public
NK	64K	4K	NK4(64K) disk	BS2000/OSD-BC V1.0	Public

The disk surface must be reformatted before the transfer unit can be changed.

As of BS2000/OSD-BC V2.0, the DUALCOPY function is available. This allows hardware-supported dual copying to disk (RAID1). DUALCOPY enables track repair on one disk of the DUALCOPY disk pair while normal operation is continued without interruption on the other one.

15.1 Installation

The VOLIN subsystem cannot be loaded by means of the START-SUBSYSTEM command; instead, it is automatically loaded dynamically with the utility routine call:

```
/START-PROGRAM FROM-FILE=$VOLIN.
```

When the utility routine is terminated, unloading of the subsystem is initiated automatically.

VOLIN requires the following files in order to run under BS2000:

Files:

VOLIN	Runtime-version-dependent phase (load module)
SYSLNK.VOLIN.112	DSSM linkage editor OML
SYSREP.VOLIN.112	REP file for object correction in the dynamically loaded code
SYSNRF.VOLIN.112	File for controlled transfer of VOLIN REPs
SYSSSC.VOLIN.112	Object file of the subsystem declarations integrated in the central subsystem catalog
SYSMES.VOLIN.112	Message file containing the VOLIN messages

15.2 Functions

Volume formatting

In accordance with the technical organization of the read/write system, a disk pack is divided into a certain number of cylinders and tracks. Formatting involves a further subdivision into blocks. VOLIN formats a disk pack in accordance with BS2000 conventions, which stipulate a fixed block structure for each disk type.

The logical unit for this subdivision is

– either a 2-Kbyte block (transfer unit = 2 Kbytes); with NK formatting, this consists of a data section 2048 bytes in length; with K formatting, this data section is preceded by a 16-byte key.

– or a 4-Kbyte block (transfer unit = 4 Kbytes); this consists of a data section 4096 bytes in length.

The physical implementation of a PAM block and the distribution of the blocks over the tracks and cylinders of a disk pack vary depending on the disk type (see the table at the end of this description).

Notes

D3475, D348x, D349x (CKD track-format disks):
Formatting entails applying a format write command for each individual block in the disk pack; this command also initiates recording of the data of a test pattern.

D348x, D349x connected to 3860-43/51:
Any data remaining in the controller's cache when VOLIN is called is automatically deleted before formatting commences.

D3492:
For performance reasons, processing is via the controller's write cache (DASD fast write ON). This status remains activated even if the user has prevented formatting in interactive mode.

D3475-8F (bus disks):
The disks were already formatted outside BS2000. VOLIN does not format the disks and does not check the volumes.

D3435:
The disks must be preformatted. VOLIN does not format the disks, but it does check the volumes (see notes at the end of this description).

D3409:
Formatting entails writing data to all blocks for the first time as part of the volume check, once a device has been created by means of the CREATE-FASTDISK command. In this instance, this special formatting command is only used to define the block-length-related device parameters.

D3434, D3436 - D3439, D34211 (all other FBA track-format disks):
Formatting is implemented by a special formatting command that does not write user data to the disks. If this command is used to format the entire usable area of the disk, the VOLIN run cannot be interrupted by pressing K2 for the first few minutes after the program starts. If VOLIN is started in parallel for a number of disks connected to the same controller, this period can be prolonged dramatically.
If the system disk is connected to the same string as the disk to be formatted, BS2000 operations are severely degraded until formatting is completed.

DUALCOPY disk pair:
The volumes of a DUALCOPY disk pair cannot be formatted in the DUALCOPY status ACTIVE, IN-HOLD or IN-COPY.

Volume checking and assignment of alternate tracks (areas)

VOLIN checks the formatted disk areas by writing a bit pattern and reading it back. If an error is detected during this operation, the area concerned is flagged as defective and at the same time an alternate area is assigned. In this way the defect on the volume is hidden from the eventual user. With the disk types currently supported by BS2000, the areas that can be locked and replaced by alternate areas are a track (CKD) or a block (FBA).

Alternate area assignment can be controlled using three VOLIN statements:

1. DEFECTS statement
   DEFECTS statements are used to specify areas to which alternate areas are to be assigned no matter what. VOLIN masks out these areas on the volume without checking them first.
2. RECLAIM statement
   This statement can be used to specify the assignment of alternate areas to areas already flagged as defective without running a new check. (With areas marked at the factory prior to shipment, this takes place independently of the RECLAIM specification.)
3. RETRY statement (CKD track format only)
   This statement defines a repetition counter specifying the number of times a read-after-write check is to be repeated in problem cases before a disk area is flagged as defective. If the repetition counter is set to zero, even errors of a purely statistical nature will result in assignment of an alternate area (where possible, VOLIN works without the standard repetitions of the device). With the repetition counter set to zero, different VOLIN runs with the same disk pack can lead to very different results in respect of faults detected.

A high repetition counter value reduces the rigor of the test and limits the assignment of alternate areas largely to those cases in which actual defects are present on the disk surfaces. On the other hand, there is the danger that errors bypassed during the VOLIN run by means of repetitions will become unrecoverable errors in the course of operation and with other data patterns. The recommended value for the repetition counter is one (this is also the VOLIN default).

Certain errors in a disk pack cannot be corrected by assigning an alternate area. In such cases, VOLIN aborts initialization and warns against the further use of that volume. Prior to this, a message is issued indicating one of the following conditions:

1. A defective area has been detected, but due to a fault on the volume it was not possible to flag the defective area as defective (message NVL0020).
2. A defective area has been detected, but the reserves of alternate areas on the volume have already been exhausted (message NVL0021).
3. A defect has been found in the area provided for the initial program loader (IPL) or for the format 5 SVL labels. According to the specifications for disk packs, no alternate area may be assigned to this area (message NVL0025).

Repeated checking of a formatted volume

The formatting of a disk pack is completed when each disk area has been formatted and checked using a bit pattern. VOLIN indicates that the formatting of a disk pack has been completed with the message:

```
NVL0022 FORMATTING COMPLETED. TEST PATTERN xxx WRITTEN ON ALL BLOCKS OF DISK
vsn
```

The FMT statement can be used to specify multiple checking of disk pack formatting. The test pattern is changed before each new run. A set of test patterns is provided for each disk type. The patterns are rotated within this set.

In a subsequent run, only those disk areas are checked that have not yet been flagged as defective. In particular, this means that areas that were flagged as defective in the first formatting run as a result of the DEFECTS and RECLAIM statements retain this flag throughout all runs.

When specifying the number of runs, you should bear in mind that the process is time-consuming and that there is only a slight probability that additional runs will detect defects not found in the initial run.

Notes

- The description given above in sections "Volume checking and assignment of alternate tracks" on page 528 and "Repeated checking of a formatted volume" on page 529 applies unconditionally only to devices not subjected to extended track checking.

  Extended track checking:

  1. Basic check of entire track using one or more bit patterns
  2. Skip displacement analysis if surface defects were detected during the basic check.

- Extended track checking is performed for devices D3480, D348E, D348F, D3490-10, -20, -30 and D4390-40 connected to a 3860-3/4 controller and for devices D3490-10, -20, -30 and D3490-40 connected to a 3860-5x controller:

  - a DEFECTS statement for a track directly initiates a skip displacement analysis but not necessarily alternative track allocation.
  - RECLAIM=NO is ignored. An attempt is made to correct each track by skipping whenever necessary.
  - the RETRY statement is meaningless.
  - FMT=n defines the number of test patterns used in the basic check. Only one complete check is run, i.e. the message NVL0022 appears once only.
  - The volume is not formatted until the extended track check is complete. The key/data fields in the formatted blocks contain binary zeros.

The duration of a VOLIN run depends primarily on the disk type (capacity) and the checking method. The extended track check takes approximately 75 % longer than the older, less precise method. It can take even longer if defective tracks are detected.

Because of the high capacity involved and for other technical reasons, track checking and formatting of D3490-40 disks can take an extremely long time (5.3 hours for a single run and up to 12.3 hours for a parallel run of 8 VOLIN runs on a 4-path controller).

By not rewriting the track addresses, the processing time required for extended track checking can be reduced by about 30% for D3490-40 disks and by about 15% for the other disk types (see OPTION=NO-WR-HA). This is only advisable, however, for straightforward reformatting, and not after the disk storage unit has been newly installed or after HDA replacement.

Generating BS2000 labels (label generation)

VOLIN writes new labels on a volume either

– after formatting is completed or

– without formatting, if the volume has already been formatted in accordance with BS2000 conventions (statement FMT=NO).

If a disk pack has a valid BS2000 standard volume label (SVL), VOLIN assumes that the volume is formatted.

VOLIN concludes the initialization of the disk pack by transferring the new standard volume label. Its presence is a guarantee that the other labels on the volume have also been renewed.

How the disk pack is used following the VOLIN run depends on whether the initialization has created a private volume, a public volume or the PUBRES disk of a pubset.

1. A private volume can be used by the BS2000 Data Management System immediately following the VOLIN run.
2. A public volume can be mounted during a subsequent loading of a BS2000 system (startup) in order to extend an existing public volume set (pubset). The following attributes have global validity in the pubset and must be taken into account:
   – The VSN must match the pubset.
   – The PK (PAM key), AU (allocation unit) and TU (transfer unit) attributes must be the same as those of the PUBRES. The following pubset types are thus possible:
     – K pubset (implicit AU=6 Kbytes)
     – NK2 pubset (implicit AU=6 Kbytes)
     – NK2 pubset with AU = 8 Kbytes
     – NK2 pubset with AU = 64 Kbytes
     – NK4 pubset with AU = 8 Kbytes
     – NK4 pubset with AU = 64 Kbytes
3. A volume which, according to the labels, has been generated by VOLIN for use as PUBRES is inoperable and has no direct use in BS2000. An operational pubset can be directly created only with SIR.
4. A disk pack initialized with VOLIN, no matter whether it is a private or public volume, can serve as the target volume in any physical data-saving operation (cf. the COPY or RLOD function of FDDRL). Note, however, that the block size of the target volume (2 Kbytes with PAM key / 2 Kbytes without PAM key / 4 Kbytes) must be the same as that of the source volume.

Note on DUALCOPY

Labels can be generated for DUALCOPY disk pairs in all DUALCOPY statuses.

Explanatory remarks on the various labels:

1. IPL block

   The IPL block is the first block (2 or 4 Kbytes) on the volume. VOLIN uses a dummy IPL here. If an attempt is made to load from this volume, the dummy IPL issues the message ILLEGAL LOAD FROM BS2000 DATA VOLUME vsn and causes the system to be placed in the wait state.

2. Standard volume label (SVL)

   The standard volume label is the second block (2 or 4 Kbytes) on the volume. It contains flags that identify the disk pack to the system as a particular volume, pointers to the other labels, and formatting information. If no formatting is carried out, VOLIN takes the formatting information from the old SVL. Formatting-related information includes:

   a) the address of the first alternate area on the disk pack for CKD track format disks only (see table on page 553)

   b) the address of the alternate area following the last alternate area already assigned (only for CKD track format disks)

   c) the address of the last usable track. Usable tracks are any sections of the disk pack that are not reserved for device maintenance.

3. Format 5 label (F5)

   The start of this label is anchored in the standard volume label (SVL). The total number of blocks (2 or 4 Kbytes) allocated to this label depends on the type of disk (see table on page 553). The format 5 label is used for managing the storage space on the volume. The smallest allocation quantity used by the BS2000 Data Management System is called a unit (1 unit = 6 / 8 / 64 Kbytes, see description on page 524, blocks are allocated consecutively, beginning with the first 2-Kbyte block). The most important part of the format 5 label is a table specifying which units on the volume are occupied and which are free. After a VOLIN run the following units are marked as occupied:

   a) all units preceding the F1 start address, as well as the units of the F1 label (see item 4 below).

   b) All units in alternate areas, irrespective of whether these areas have been assigned. This only applies to disks usable under BS2000 versions prior to V10.0.

   Space for alternate areas is reserved at the end of the volume. The amount of space reserved depends on the disk type.

Note

The contents of the units designated as free are not deleted if the volume is not formatted. Following a VOLIN run with formatting, all blocks outside the labels contain a test pattern or binary zeros.

4. Format 1 label (F1)

   This label begins at a unit boundary after the F5 label. Its start address is identical for all the disk packs of one type.

   The format 1 label exists only on private volumes. It contains the catalog of the files that begin on that volume. On public volumes, this position corresponds to the beginning of the storage space available for files. Together with the F5 label, the F1 label forms an area on the disk referred to as the VTOC area (VTOC = volume table of contents).

   System administration can define the length of the F1 label during the VOLIN run (by means of the F1SIZE statement; see "Operation" below, point c). VOLIN initializes the F1 label with empty catalog blocks.

Track repair (repair function)

The track repair function can only be used on a fully initialized CKD disk storage unit.
Its use is restricted to device types with the extended track checking facility (see notes on page 535).
For public volumes, the pubset must be in exported status (except for DUALCOPY, see page 536).
The (single) track repair function which is quicker than backing up, formatting and restoring the entire disk is designed for the following purposes:

1. Preventative maintenance
   In the event of frequent media SIMs (service information sessages) on a device or other entries of the HERSFILE device error handling function caused by interface errors, it is advisable to check and repair the tracks affected.

2. Selective track repair in an emergency
   A block can no longer be read due to a permanent error (device error message EXC0857 ....PERR..).
   The affected files must be restored from the backup stock after the track repair.

How track repair works

Most irrecoverable errors are the result of minor damage to the track surface. A track check pinpoints these areas and skips them with the help of the skip displacement analysis function (errored areas are not used for data). If this does not solve the problem, an alternate track is assigned.

During a track check, the track is overwritten. Consequently, if possible and if necessary, VOLIN must back up the old contents first.
If a track block is not readable, it cannot be restored. However, the track can be repaired, which is a precondition for subsequent restoration of the block from the backup (as part of a file recovery).
If a block still remains unreadable despite all VOLIN's recovery attempts, the number of the physical half-page (PHP) of the block is output in an error message. The file assigned that block number can then be identified using the /SHOW-BLOCK-TO-FILE-ASSIGNMENT command.
Another possibility is to read the message from the device error handling function which identifies the device, track and, if applicable, block. The file can then be found with the aid of the SPCCNTRL utility routine (see page 481) and must be read in from the backup.

The track check does not depend on whether a block is readable or not. When the blocks are written back, the unreadable ones are omitted and therefore retain their formatting contents (binary 0). Each block that was not readable and omitted when the blocks were written back is listed in the NVL0124 message.

Track repair is slightly different in a DUALCOPY environment. In particular, the track to be repaired does not need to be backed up (see "Track repair and volume assignment with DUALCOPY" on page 536).

Track repair can be influenced by the following VOLIN statements:

1. REPAIR statement
   This statement identifies the track repair function.

2. VSN statement
   This statement specifies the vsn of the disk pack on which the track repair is to be carried out.

   Note

   The VSN statement differs here from the VOLIN "label generation" function. The standard volume label (SVL) is read only.

3. DEFECTS statement
   This statement specifies tracks to be repaired. A track is identified by its track address or the PHP number of the block containing the track.

   Note

   The DEFECTS statement is used here differently than in the VOLIN function "Volume checking and assignment of alternate track areas" (see page 528).

4. DUALCOPY statement
   This statement specifies dual recording by the disk controller. The DUALCOPY disk pair must have the status IN-HOLD.
5. DEVICE statement
   This statement specifies the disk pack device type.
6. UNIT statement
   This statement specifies the mnemonic device name of the disk pack. This is necessary when there are several disk packs with the same SVL (standard volume label) and assignment with the VSN statement is ambiguous.

Notes

The statements CHECK, FMT, FAST, RETRY, RECLAIM, UNAME, F1SIZE, VTOC, and FORMAT are irrelevant to the track repair function and are ignored.

The extended track check facility is used for D3480, D348E, D348F, D3490-10, D3490-20, D3490-30 and D3490-40 devices connected to 3860-3/4 controllers and for D3490-10, -20, -30 and D3490-40 devices connected to 3860-5x controllers (see table on page 553).

Track repair on a D3492 device uses a different procedure. For further details, see "Track repair of D3492 disks" on page 537.

The VOLIN repair function can only be used under the BS2000 operating system, i.e. it cannot be used under SIROS or in SIR-IP.

If the VOLIN repair function is used in the DRV (dual recording by volume) environment, the user must first switch to SRV (single recording by volume) mode.

Track repair in the DUALCOPY environment

The DUALCOPY function supports dual recording on a DUALCOPY disk pair by the disk controller.
DUALCOPY [19] enables one disk of the DUALCOPY disk pair to be repaired while the other one continues to be used without interruption in normal operation.
Track repair is executed in the DUALCOPY status IN-HOLD in which the applications on the primary disk continue running while the SECONDARY disk is processed by VOLIN.

figure 16 illustrates the DUALCOPY statuses and transitions at the time of track repair. Only the statuses relevant to working with VOLIN are represented.

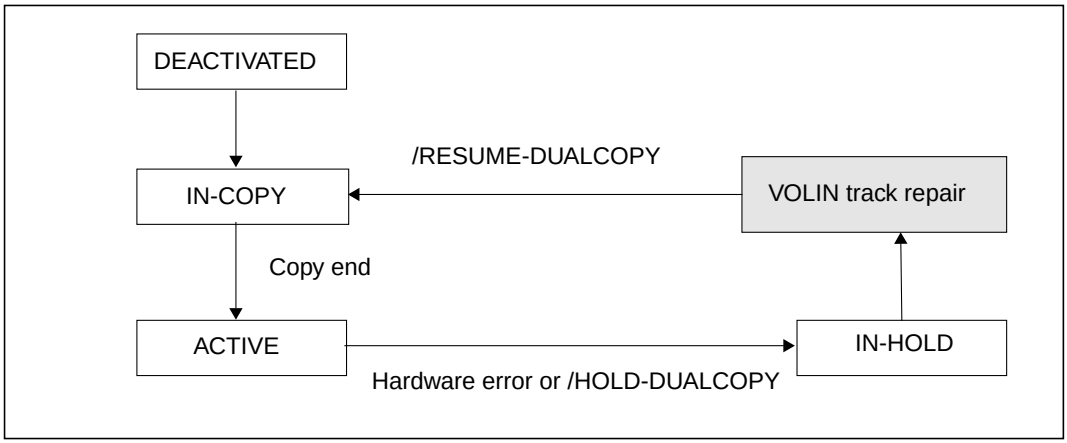


Figure 16: Time of track repair and DUALCOPY status transitions

The tracks selected with DEFECTS statements on the secondary disk are subjected to an extended track check. Surface errors are located and skipped with the help of the skip displacement analysis function. If this is unsuccessful, an alternative track is assigned. The contents of a track on the secondary disk are not saved before the track check because the current contents of the track on the primary disk are continuously being changed.

If the track repair is successful, ACTIVE status can be resumed using the /RESUME-DUALCOPY command. During the equalization phase, the controller copies all changed tracks from the primary disk to the secondary disk. This includes not only the tracks changed on the primary disk while the secondary disk could not be accessed by the application, but also the tracks changed by VOLIN during track repair.

If track repair was unsuccessful due to a volume or device error, the system administrator must set the status DEACTIVATED.

Track repair and volume assignment in DUALCOPY

If the DUALCOPY disk pair has the IN-HOLD status, VOLIN does not reserve the primary disk exclusively to ensure that the applications can continue to run. The secondary disk does not need assignment because, in terms of device management, VOLIN works with the primary disk. VOLIN's inputs and outputs are diverted to the secondary disk (with direct I/O).
Volume assignment in the DUALCOPY environment can be influenced by the following VOLIN statements (in addition to the statements listed on page 534).

1. DUALCOPY statement
   This statement specifies dual recording by the disk controller. The DUALCOPY disk pair must have the status IN-HOLD.

2. VSN statement
   This statement specifies the VSN of the disk pack on which the track repair is to be carried out. A DUALCOPY disk pair has identical SVLs (standard volume label). VOLIN ensures that it is the SECONDARY disk that is processed.

3. UNIT statement
   This statement can be used to specify a mnemonic device name. If it is specified, it must be the mnemonic device name of the MAIN-UNIT, even if the primary and secondary disk are switched.

Track repair of D3492 disks

D3492 disks have RAID5 architecture. The specifications and track repair method of operation described so far only apply in part to this type of disk.

Track repair is only necessary on D3492 disks when "track in error" occurs.

– D3492 disks have a spare drawer to which data is copied in the event of an error or prior to maintenance work. If, in spite of the RAID5 technology, an error occurs during a recovery copy procedure, the track containing the unreadable section is identified as unavailable, i.e. as "track in error".

– During copying, a "media service information message" (media SIM) identifying the track is generated. The media SIM is entered in the HERSFILE, from which the system administrator can pinpoint the “tracks in error” that occurred during copying.

– After copying, the spare drawer and the original device are switched. Any attempt to access blocks containing the "track in error" track is terminated with an error. The files that were on the "track in error" track must be read in again from the backup because the track contents cannot be restored.

– The "track in error" track must be dealt with by the repair function to restore it to normal availability. This track repair does not include a track check because the track is error-free. Only the home address and the track descriptor record containing the lock comment are written anew and the track is reformatted.

VOLIN functionality in the DUALCOPY environment

BS2000/OSD-BC V2.0 is the first version to feature DUALCOPY, which supports dual recording by disk controllers (RAID1).
The DUALCOPY disk pair statuses and the VOLIN functions available are shown in the table below.

VOLIN function	DUALCOPY status			
	ACTIVE	**IN-HOLD**	**IN-COPY**	**DEACTIVATED**
Initialize with formatting (FMT=Y)	no	no	no	yes
Initialize without formatting (FMT=N)	yes	yes	yes	yes
Single track repair (REPAIR=Y)	no	yes	no	yes

15.3 Program execution

General information

VOLIN runs under BS2000 and SIROS. Its functionality is also integrated in the SIR-IP installation program. This section describes execution of VOLIN under BS2000. Any differences between this and the SIROS version or the version integrated in SIR-IP are described in the "System Installation" manual [15].

VOLIN uses privileged functions and can run only under the TSOS user ID. Any attempt to execute VOLIN under another user ID results in message NVL0001 and immediate termination of the routine.

VOLIN can be called in an interactive job or in a batch job. In interactive mode VOLIN handles questions and answers regarding disk initialization via the terminal, otherwise it handles them via the console to which system administrator duties have been assigned.

Any number of disk packs can be initialized in one VOLIN run. Disk packs are initialized one after the other. Every initialization is controlled by a set of statements read in from SYSDTA. In the event of an error, initialization of the volume currently being processed is terminated. In interactive mode, a new set of statements may then be entered; in a batch job, the VOLIN run is also terminated (TERM MODE=ABNORMAL).

Note

Multiple concurrent VOLIN runs can degrade BS2000 operation and should therefore be avoided. At the very least, disks to be formatted should not be accessed via a channel or controller that is also being used for other heavily used disks.

Assignment of volumes

VOLIN requests the disk pack to be initialized via the system device management facility and reserves it exclusively at its own processor.
This does not apply to track repair for DUALCOPY in the status IN-HOLD (see page 536).

Assignment of volumes for VOLIN functions without track repair

If a specific disk pack is to be initialized, VOLIN must be provided with the old volume serial number as well as the disk type (via the CHECK statement; see "Operation" below, point c). If the volume thus specified is already mounted but not reserved, then it is immediately assigned by device management; if it is not mounted, it is requested using its old volume serial number. If device management cannot reserve the volume specified, VOLIN terminates initialization.

If VOLIN is given only the disk type, any disk pack of this type can be assigned ("scratch volume"). VOLIN uses the SVL to inform system administration about the mounted volume and asks whether the volume may be overwritten. If the answer is YES (i.e. system administration permits overwriting), the mounted disk pack is initialized. If the answer is negative, three different results are possible:

– Initialization is terminated if the volume is on a fixed-disk device specified with the UNIT statement.
– If no disk device was specified with the UNIT statement, a different disk device is requested.
– If the operator declines to provide another disk pack when responding to the message from device management, initialization is terminated.

If the volume does not have a valid SVL, initialization must include formatting. In interactive mode it is possible to agree to formatting retroactively, thus revoking an earlier FMT=NO statement.

The same applies when a change in volume type (see the FORMAT statement) causes the block structure of the volume to change (see list of supported disk types on page 553).

It is important to ensure that initialization is coordinated with the work done by other processors that also have access to the device occupied by the volume.

The following message offers a chance to reject a device generated as shareable:

```
NVL0033 DISC DEVICE mn ASSIGNED TO VOLIN IS SHARABLE TO THIS MOMENT.
REJECT DEVICE? REPLY (Y=YES;N=NO)
```

If the device is a fixed-disk device requested by means of the UNIT statement, or if the volume was requested by means of the CHECK statement, initialization is aborted after the rejection; otherwise a new volume is requested.

Volume assignment for track repair

If a specific disk pack is to be repaired, VOLIN requires the VSN and the disk type to be predefined. In addition, the mnemonic device name can be specified by means of the UNIT statement. This is necessary when there are several disk packs with the same SVL (standard volume label) and assignment by means of the VSN statement is ambiguous. If the device management function cannot reserve the specified volume, VOLIN aborts the function.

The system administrator must ensure that the track repair is coordinated with work on other computers which also access the device reserved for the volume.

15.4 Operation

a) Command level:

The routine is started under TSOS by

```
/START-PROGRAM FROM-FILE=$VOLIN
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of the START-PROGRAM command are available for calling the routine, e.g. to monitor the program run. For descriptions of these operands, see the START-PROGRAM command in the "Commands, Volume 3" manual [3].

b) Console level:

VOLIN sends messages to the console (routing code A) and to SYSOUT. When VOLIN is running in a batch job, responses to messages are also read in from the console. See "Program execution" above.

c) VOLIN statements:

Every initialization of a disk pack is controlled by a set of statements that VOLIN reads in from SYSDTA. Once a set of statements has been read in, VOLIN checks it for consistency and completeness. If no errors are present, initialization is performed and a new set of statements is then requested.

The statements are passed to VOLIN in input lines. Each input line may contain several statements, separated by commas, but the length of an input line must not exceed 251 characters. In accordance with the rules of the BS2000 command language it is possible to insert blanks and comments.

VOLIN requests the first line of a set of statements with the message:

```
NVL0002 ENTER VSN AND DEVICE TYPE OR TERMINATE VOLIN WITH 'END'
```

All subsequent input lines are requested with the message:

```
NVL0003 ENTER ADDITIONAL INITIALIZATION PARAMETERS OR TERMINATE INPUT
WITH 'EOT=YES'
```

A set of statements is concluded with a line containing EOT=YES, either alone or together with other statements. A blank line (ETX) has the same effect as EOT=YES. Each line may contain any combination of different statements. Statements that have already been specified can be repeated in new input lines and thus modified.

VOLIN is terminated when message NVL0002 is answered with END. If the same response is given to message NVL0003, VOLIN first generates an EOT=YES condition and processes the set of statements thus concluded. Subsequently, no new set of statements is requested and the run is terminated.

End of file (EOF) on SYSDTA has the same effect as END.

15.5 Statements

15.5.1 Overview of the VOLIN statements

Format	Meaning
VSN=vsn	Volume serial number after initialization (must be specified) or for track repair, VSN of disk pack to be repaired (must be specified)
DEVICE=device	Device type of the disk pack (must be specified)
[C[HECK]={vsn / SCRATCH}]	Volume serial number prior to initialization, or SCRATCH for an arbitrary disk pack
[UNIT=mn]	Mnemonic device name; must be specified when initialization/track repair is to take place on a specific device
[F[MT]={n / Y[ES] / N[O]}]	Initialization with (number = n) or without formatting
[FAST={Y[ES] / N[O]}]	Specifies whether or not formatting is to be accelerated at the expense of other jobs
[REPAIR ={Y[ES] / N[O]}]	Execution of track repair, restricted to one or more tracks of a fully initialized CKD disk storage unit
[RETRY=n]	Number of repetitions before a disk area is treated as defective
[DEFECTS={(c:h[,c:h,...]) / TRACK(c:h) / TRACK((c:h,c:h,...)) / RECORD(nn) / RECORD((nn,nn,...))}]	Addresses of tracks that are to be treated as defective, or, for single track repair, addresses of tracks to be repaired.
[R[ECLAIM]={Y[ES] / N[O]}]	Rechecks disk areas identified as defective.
[UNAME=name]	User (owner) name to be left in the SVL (not evaluated by the system)

Format	Meaning
[{F1SIZE / VTOC} =n]	Number of PAM blocks for the format 1 label

Format	Meaning
[FORMAT= {U[NCHANGED] / K / NK(PHYSICAL-BLOCK-SIZE= {2K(ALLOCATION-UNIT={6 / 8 / 64}) / 4K(ALLOCATION-UNIT={8 / 64})})}]	Defines the volume type
[DUALCOPY={Y[ES] / N[O]}]	Identifies a DUALCOPY disk pair with IN-HOLD status during track repair
[OPTION=NO-WR-HA]	Track check without write-HA
[EOT=Y[ES]]	Concludes the set statements with current set of statements
[{END / H[ALT]}]	Terminates the VOLIN run after processing the current set of statements

15.5.2 Descriptions of the individual statements

VSN=vsn

Volume serial number (must be specified).

For all VOLIN functions without track repair:

The alphanumeric value, max. 6 characters, is written in the volume serial number field of the standard volume label (SVL).
If the volume is to be initialized for later use as a public volume, the following should be noted, depending on the length of the pubset VSI:

– single character VSI for the pubset
The volume serial number begins with the characters PUB. The fourth character is the VSI. The final two characters must be numeric, e.g. PUBA01. Zeros in these two positions identify a volume as the system residence, e.g. PUBA00.

– two- to four-character VSI for the pubset
The first characters are the pubset identifier. The end of the identifier is indicated by a period in the next position. The conventions applying to the remaining positions are as follows:

– NK4 and NK2(8K,64K) disks:
digits (0..9) and letters (A..Z) are permissible

– all other disk types:
the last character must be a digit or a letter from the set A..V. All other positions must contain zeros.

Examples

– VSN with two-character pubset identifier : AB.005

– VSN with three-character pubset identifier : ABC.0F

– VSN with four-character pubset identifier : ABCD.0

For track repair:

VSN of the disk pack to be repaired; used exclusively by the device management function to reserve the volume to be repaired.

DEVICE=device

Device type (must be specified).
Type of disk pack to be initialized/repaired. The permissible values for this statement can be taken from the table at the end of this section.

C[HECK]

=vsn — Volume serial number (max. 6 characters).
Old volume serial number of the disk pack to be initialized. Only a volume bearing this volume serial number will be initialized. Following assignment, the volume is overwritten without first requesting confirmation from the user.

=SCRATCH — Entry (also default value) when no specific volume is to be initialized. VOLIN reports on the volume assigned and asks whether it is to be overwritten.

Note

CHECK is irrelevant if REPAIR=YES is specified.

UNIT=mn — Mnemonic device name.
The mnemonic device name is the 2- or 4-character code assigned to a particular device at system generation. This statement must be entered if the user wishes to specify a particular device for initialization. If the statement UNIT=mn is not specified, either the system selects an available device of the correct type, or the operator does so by responding to the MOUNT message.

The initialization is not carried out if the DEVICE and UNIT specifications contradict each other.

Note

UNIT can be specified for REPAIR=YES. This is necessary when there are several disk packs with the same SVL.
If UNIT is specified in conjunction with REPAIR=YES and DUALCOPY=YES, the mnemonic device name of the MAIN-UNIT of the DUALCOPY disk pair must be specified here.

F[MT]

=Y[ES] — The disk pack is to be formatted once during the initialization run (default value). YES is equivalent to FMT=1 (see FMT=n below).

=N[O] — Initialization of the disk pack is to be carried out without formatting and surface checking. In this case the disk pack must already be formatted and have a valid SVL.

=n — During initialization, the disk pack is to be formatted and checked in (n) formatting and checking passes, where n is a decimal number between 0 and 10.

0 is equivalent to FMT=NO, i.e. no formatting/checking. In each formatting pass, a specific test pattern is recorded.
For a new pass, the test pattern is changed to the next one in the (cyclical) sequence.

One run is sufficient to format a new disk pack. To optimize program runtime, the number of checking passes performed on disk packs that are no longer new should not exceed 2 or 3. (see note in section "Repeated checking of a formatted volume", page 529).

Note

FMT is irrelevant if REPAIR=YES is specified.

FAST

=Y[ES] Formatting is to be performed as quickly as possible, without regard to other jobs in the system.
Specifying this operand has no impact on the duration of the VOLIN run for disks of type 3480 and 3490.

=N[O] Formatting is not to be expedited at the expense of other jobs (default value).

Note

FAST is irrelevant if FMT=NO and REPAIR=YES are specified. FAST works only with disks with CKD track format without extended track checking.

RETRY=n Decimal number (0-127).
This statement provides a count indicating how often an I/O operation is to be repeated during volume checking before a track is to be considered defective.
If no value is specified, the repetition counter is set to 1 (recommended value).

Note

RETRY is irrelevant if FMT=NO is specified and disks with FBA track format and D3410 disks are used (see overview of disk types on page 553 and note in section "Repeated checking of a formatted volume", page 529). RETRY is irrelevant for disks with extended track checking and REPAIR=YES.

DEFECTS

=(c:h,c:h,...) Track addresses: list of decimal numbers; maximum: 64elements. c is the cylinder number and h the head number of a track that is to be treated as defective during formatting or repaired during track repair. This operand entry makes sense if errors had already occurred sporadically prior to a VOLIN run, with the result that defective tracks are already known.

=TRACK{(c:h) / ((c:h,c:h,..))}

Track addresses: list of decimal numbers, maximum 64. c is the cylinder number and h head number of a track that is to be treated as defective during formatting or repaired during track repair. This operand entry makes sense if errors had already occurred sporadically prior to a VOLIN run, with the result that defective tracks are already known.

= RECORD{(nn) / ((nn,nn, ...))}

Number of the PAM block which

- is to be treated as defective on a disk with FBA track format and
- is on a track that is to be treated as defective or repaired during track repair on a disk of any other type.

nn is the RBN (relative block number) in the case of disks with FBA track format, and the PHP (physical half-page) for all other disk types. A maximum of 64 numbers may be specified.

Notes

- DEFECTS is irrelevant if FMT=NO is specified.
- The meaning of DEFECTS when REPAIR=YES is specified differs from that when REPAIR=NO is specified.
- For disks with FBA track format, only the RECORD specification is permissible.
- DEFECTS is ignored for disk types D3409 and D3410.
- When REPAIR=YES is specified, DEFECTS is only accepted for disks with extended track checking.
- DEFECTS is ignored if the physical block structure is changed. This is logged by message NVL0108.

Examples

- Changed usage mode (PK) requires reformatting
- Changed TU (transfer unit).

R[ECLAIM]

=YES Tracks (or blocks in the case of disks with FBA track format) already flagged as defective are to be checked by VOLIN and released if no error is detected (default value).

Note

Areas locked by the disk manufacturer are not released.

=NO All tracks/blocks already flagged as defective are to remain locked. If disks with the FBA track format are formatted repeatedly with RECLAIM=NO, you should bear in mind that only one entry is made in the DEFECTS statements for a specific block. If the same relative block number is specified again, the replacement block is shifted each time. The constantly increasing number of defective blocks causes the data module to be considered unusable after a few runs, since there are too many defective blocks per cylinder. This can also be caused by an entry in the DEFECTS statement in which too many blocks of a cylinder were flagged as defective.
In such cases, the disk should be formatted with RECLAIM=YES and a limited list in the DEFECTS statement.

Note

RECLAIM is irrelevant if FMT=NO is specified and for D3435, D3409 and D3410 disks (see also note in section "Repeated checking of a formatted volume", page 529).
RECLAIM is irrelevant for disks with extended track checking.
RECLAIM is irrelevant if REPAIR=YES is specified.

UNAME=name Owner name consisting of up to 10 alphanumeric characters and written in the name field of the SVL. The first character must be a letter or one of the following characters: $, # or @. If no value is specified, blanks are used.

Note

UNAME is irrelevant if REPAIR=YES is specified.

F1SIZE=n — Decimal number (1 to 32766).
This statement applies only to private volumes. The decimal number is the number of PAM blocks to be reserved for the format 1 label. If necessary, the specified value is rounded up to give an integral number of units. A maximum of about 16 files can be cataloged in one block.
If no value is specified, 60 PAM blocks are allocated for the F1 label on a private volume. (For compatibility reasons, the keyword VTOC may be used instead of F1SIZE.)

Note

F1SIZE is irrelevant if REPAIR=YES is specified.

FORMAT

=UNCHANGED — The volume type is to remain unchanged. This value assumes the volume has a valid SVL, otherwise value K is used by default.

=K — The volume is to be usable with PAM keys.

=NK(PHYSICAL-BLOCK-SIZE=2K(ALLOCATION-UNIT=6/8/64))
Volume formatting:
The volume is to be usable without PAM keys. Only the NK DMS access methods are permissible. The physical block size is to be 2 Kbytes, the smallest allocation unit (i.e. the smallest memory unit provided by the system) is to be 6, 8 or 64 Kbytes.
Default allocation unit: 6 Kbytes.

=NK(PHYSICAL-BLOCK-SIZE=4K(ALLOCATION-UNIT=8/64))
Volume formatting:
The volume is to be usable without PAM keys. Only the NK DMS access methods are permissible. The physical block size is to be 4K bytes, the smallest allocation unit (i.e. the smallest memory unit provided by the system) is to be 8 or 64 Kbytes.
Default allocation unit: 8 Kbytes.

PHYSICAL-BLOCK-SIZE

=2K
=4K — Length of the smallest transfer unit (TU), i.e. size (in Kbytes) of the smallest possible data block for transfer between the disk and main memory.

ALLOCATION-UNIT

=6
=8
=64 — Smallest allocation unit in Kbytes, i.e. smallest unit in memory made available by the system when a file is created.

Restrictions

- NK cannot be defined as the usage type for a D3480 disk connected to a 3418 Disk Control Unit. Processing of this volume is terminated.
- Disks cannot be configured with 4 Kbytes as their TU if they are connected to a 3418 Disk Control Unit.
- D3435 and D3475 disks cannot be configured with 4 Kbytes as their TU.
- The following can be specified for public volumes only:
  FORMAT=NK(PHYSICAL-BLOCK-SIZE=4K(...))
  FORMAT=NK(PHYSICAL-BLOCK-SIZE=2K(ALLOCATION-UNIT=8))
  FORMAT=NK(PHYSICAL-BLOCK-SIZE=2K(ALLOCATION-UNIT=64))

Note

FORMAT is irrelevant when REPAIR=YES is specified.

OPTION=NO-WR-HA

Extended track checking without rewriting of track address (HA). This reduces the VOLIN runtime (cf. page 529) and can be used for reformatting but not for new installation or HDA replacement.

Note

OPTION is irrelevant when REPAIR=YES is specified.

REPAIR

=Y[ES] Track repair of a fully initialized CKD disk storage, reduced to one or more tracks specified by means of the DEFECTS statement.

=N[O] No track repair as VOLIN function but initialization with or without formatting.

Note

REPAIR=YES is only accepted for devices with extended track checking. REPAIR=YES is rejected if the device type does not permit track repair. This is recorded in message NVL0122.

DUALCOPY

=Y[ES] Indicates that track repair is to take place in a DUALCOPY environment. IN-HOLD status must be set.

=N[O] Indicates that track repair is to take place without a DUALCOPY environment and with the status DEACTIVATED.

Note

DUALCOPY is irrelevant when REPAIR=NO is specified. If DUALCOPY=YES is specified, the system checks whether the DUALCOPY disk pair has the status IN-HOLD. If not, track repair is rejected. This is recorded in message NVL0127.
If the DUALCOPY disk pair has the status IN-HOLD and DUALCOPY=NO is specified for track repair, message NVL0123 is output asking for further instructions on how to proceed. If the DUALCOPY IN-HOLD status is not confirmed, track repair is rejected.

EOT=Y[ES] The set of statements is to be concluded with the current input line and initialization carried out. (All statements in the current input line are processed.)

END VOLIN is to be terminated after the current set of statements has been processed. (For reasons of compatibility, H or HALT may be used instead of END. Apart from END or HALT no other statements will be accepted in the input line.)

15.6 Overview of disk types supported by BS2000

1	2	3	4	5	6	7	8	9
D3409	3409 -26, -46 *1)	85	N	K/NK	2K	1907340	FBA	N
				NK	4K	1907340		
D34211 -2	34211-21, -24	86	N	K/NK	2K	1026022	FBA	N
				NK	4K	1028880		
D3434 -10	3434-10, -11, -12	81	N	K/NK	2K	502800	FBA	N
				NK	4K	469280		
D3434 -20	3434-20, -21, -22	82	N	K/NK	2K	827412	FBA	N
				NK	4K	829704		
D3435	3435 *2)	A5	N	K	2K	228398	FBA	N
				NK	2K	285498		
D3436	3436, -10, -12, -2	A2	N	K/NK	2K	170488	FBA	N
				NK	4K	170488		
D3437	3437, 3437-2	A3	N	K/NK	2K	331692	FBA	N
				NK	4K	330848		
D3438 -20	3438-20, -232, -22	A4	N	K/NK	2K	608733	FBA	N
				NK	4K	640998		
D3438 -30	3438-31, -32 34211-11	83	N	K/NK	2K	827412	FBA	N
				NK	4K	829704		
D3439 -10	3439-10, -12	A1	N	K/NK	2K	599696	FBA	N
				NK	4K	626034		
D3475	3475	AB	N	K/ NK	2K	183260	CKD	N
D3475 -8F	Bus disks 74305, 75435 *2)	8F	Y	K	2K	509520	FBA	N
				NK	2K	509520		
D3480	3480-1, -2, -11, -12 3848-A4, -B4, -AD4, -BD4 3410 *1)	AC	Y	K	2K	225675	CKD	Y
				NK	2K 4K	238950 265500		
D348E	3480-21, -22 3848-AE4, -BE4	AD	Y	K	2K	451350	CKD	Y
				NK	2K 4K	477900 531000		

*1) Maximum values are shown. Cylinder allocation is random.
*2) These disks are preformatted. VOLIN does not format them again but only gives them labels. If the volume checking function finds that the disk needs formatting, please contact the Siemens Nixdorf technician responsible.

1	2	3	4	5	6	7	8	9
D348F	3480-131, -132	AE	Y	K	2K	677025	CKD	Y
				NK	2K 4K	716850 796500		
D3490 -10	3490-1A4, -1A8, -1B4, -1B8, -1BC	A7	Y	K	2K	317205	CKD	Y
				NK	2K 4K	350595 400680		
D3490 -20	3490-2A4, -2A8, -2B4, -2B8, -2BC	AF	Y	K	2K	634410	CKD	Y
				NK	2K 4K	701190 801360		
D3490 -30	3490-3A4, -3A8, -3B4, -3B8, -3BC	89	Y	K	2K	951615	CKD	Y
				NK	2K 4K	1051785 1202040		
D3490 -40	3490-4A4, -4A8, -4B4, -4B8, -4BC	8A	Y	K	2K	2854845	CKD	Y
				NK	2K 4K	3155355 3606120		
D3492	3492-141, 34921-121	8E	Y	K	2K	951615	CKD	Y
				NK	2K 4K	1051785 1202040		

*1) Maximum values are shown. Cylinder allocation is random.
*2) These disks are preformatted. VOLIN does not format them again but only gives them labels. If the volume checking function finds that the disk needs formatting, please contact the Siemens Nixdorf technician responsible.

Meanings of the columns:

1. Device type as specified in BS2000 commands
2. Disk storage unit product number
3. Device type code
4. For changing the usage mode with reference to PAM key application with 2K:

   N: no formatting is required
   Y: formatting is required
5. Usage mode K with PAM key, NK without PAM key
6. Smallest transfer unit (TU) between disk and main memory
7. Net capacity in units of 2 Kbytes (PAM blocks) per volume
8. Track format
9. REPAIR=YES possible: N : no, Y : yes

Restrictions

– NK cannot be specified as the usage mode for D3480 disks connected to 3418 Disk Control Units. Processing of this volume is aborted without a result.

– Disks with a TU of 4 Kbytes cannot be formatted on 3418 Disk Control Units. Processing of this volume is aborted without a result.

– Track repair (REPAIR=YES) is not possible for D3480, D348E disks connected to a 3418 Disk Control Unit.

– Changing the TU always necessitates reformatting.

– Changing the AU without changing the TU does not necessitate reformatting.

– The following can be specified for public volumes only:

  FORMAT=NK(PHYSICAL-BLOCK-SIZE=4K(...))
  FORMAT=NK(PHYSICAL-BLOCK-SIZE=2K(ALLOCATION-UNIT=8))
  FORMAT=NK(PHYSICAL-BLOCK-SIZE=2K(ALLOCATION-UNIT=64))

Note

3410 External High-Speed Storage Unit:
The 3410 External High-Speed Storage Unit must be formatted for initial startup and every time the capacity is changed.
Initialization without formatting is not permissible after changing the capacity of the external high-speed storage unit. This device is not subjected to a volume check, so no alternative areas are allocated. This, in turn, means that the DEFECTS, RECLAIM and RETRY statements are meaningless; these statements are ignored if issued. The statement REPAIR=YES is rejected. The 3410 External High-Speed Storage Unit should be coded as DEVICE=D3480.
The disk is handled like the corresponding device type, but the capacity values differ. See the appropriate device description for details.
In VOLIN statement specifications relating to the 3410 External High-Speed Storage Unit, the F1SIZE or VTOC block number entry should be commensurate with the real device size compared with the 3480 (when creating a file, for example, specifying F1SIZE=3 or VTOC=3 is sufficient).
All other VOLIN statements can be used normally.

16 SYSUPD: utility routine supported for reasons of compatibility

Version: **SYSUPD V11.2A**

The SYSUPD routine is the logical counterpart of the TSOSMT file saving routine. Files that were saved to tape with TSOSMT are transferred back to public or private disks in the system by means of SYSUPD.

The nonprivileged user, i.e. the user not working under the system administration ID TSOS, can use SYSUPD only to spool in SAVTAPs (user save tapes). System administration, under the TSOS ID, can also spool in MASTAPs (system master tapes).

Note

Files which are protected by a password are restored without a password. This cannot be avoided in a system with password encryption.

16.1 Program execution

SYSUPD can be started in two ways:

- `START-SYSUPD`
- `START-PROGRAM FROM-FILE=$SYSUPD`

After the loading message the routine awaits the input of statements from the SYSDTA system file. During input, the statements are subjected to a syntax check and any errors are reported to SYSOUT. The length of one statement is limited to 256 characters. All statements that have been accepted by SYSUPD (i.e. that are free from syntax errors) are stored; they are not executed until the END statement has terminated statement input.

The SYSUPD statements are divided into three functional groups:

1. Statements for the input tape

   The OPN statement
   opens one or more input tapes that are to be spooled into the system. All files to be spooled in must be specified in a file selection statement following the OPN statement.

   The LST statement
   outputs a directory of a tape. This may be useful for determining which files are stored on the tape. After a LST statement the SYSUPD run is automatically terminated.

2. File selection statements

   File selection statements always follow the OPN statement which opened an input tape. They specify those files of the input tape that are to be spooled into the system and define the disks on which they will be stored.

   The SEL statement
   selects one or more files from the input tape and specifies the volume on which they are to be spooled in.

   The ALL statement
   defines all files or certain file groups of the input tape and specifies on which disks these files are to be spooled in.

   The REN statement
   spools in one or more files from the input tape, assigns a new name to the file(s) being spooled in, and defines the disks to which the file(s) is/are to be transferred.

   The PROTECT statement
   prevents files protected with ACCESS=READ from being overwritten.

3. Program control statements

   Additional statements control the SYSUPD run and the output of information to the user.

   The DISPLAY statement
   outputs to SYSOUT the name of each file that has been successfully transferred; where this is not the case, the corresponding names are output with an additional comment.

   The HELP statement
   outputs to SYSOUT a list of all SYSUPD statements, including the valid operands.

   The END statement
   must be issued to terminate the input of statements to SYSUPD; only then will file transfer start.

16.2 Statements

The statements are read from SYSDTA. If SYSDTA is a cataloged file, the file type must be SAM or ISAM. The HELP statement informs the user briefly of the options available in SYSUPD

16.2.1 Overview of all SYSUPD control statements

The statements are read from SYSDTA. If SYSDTA is a cataloged file, the file type must be SAM or ISAM. The HELP statement informs the user briefly of the options available in SYSUPD.

Operation	Operands	Meaning
ALL	{ [$userid.] filename. $userid. [filename.] }] [,VSN={ SYSTEM private PUBnn SYSRES }] [,TAPES=n]	Copies all files from SAVTAP to public or private random access volumes
DISPL[AY]		Displays actions taken by SYSUPD on SYSOUT (interactive mode only)
END		Indicates end of operand input
HELP		Provides a listing of all statements with short descriptions
LST	{ SAVTAP[,SYSOUT] [,FNIN=inputfilename] ,VSNIN={vsn|(vsn0,vsn1,...,vsn9)} [,D=P|G|TAPE-C1|T-C1|“device“] }	Outputs tape contents and terminates the routine
OPN	{ SAVTAP[,FNIN=inputfilename] ,VSNIN={vsn|(vsn0,vsn1,...,vsn9)} [,D=P|G|TAPE-C1|T-C1|“device“] }	Opens a SAVTAP
PRO[TECT]		Files protected by ACCESS=READ are not overwritten

Operation	Operands	Meaning
REN	(filename-old,filename-new)[,...] [,VSN={SYSTEM / privat / PUBnn / SYSRES}]	Changes the name of a file on a SAVTAP or MASTAP and copies it to a public or private volume
SEL	filename[,filename][,...] [,VSN={SYSTEM / private / PUBnn / SYSRES}]	Selects files from a SAVTAP or MASTAP and transfers them to the system

16.2.2 Descriptions of the statements

ALL statement for file selection

The ALL statement spools in either all files of an input tape or certain file groups of the tape.

The nonprivileged user can specify whether the files are to be transferred to private or public disks. System administration can also specify the name of a specific public disk or that of the SYSRES disk as destinations.

A MASTAP must not be processed with the ALL statement without operands.

Operation	Operands
ALL	[{[$userid.]filename. / $userid.[filename.]}] [,VSN={SYSTEM / private / PUBnn / SYSRES}] [,TAPES=n]

$userid. — Means that all input tape files whose name starts with "$userid." will be spooled in.

"$userid." is integrated in the file name only if the ALL statement was issued during saving with TSOSMT.

Nonprivileged users can specify only their own user ID for "userid". Only system administration can specify any user ID.

filename. — Is a partially qualified file name. SYSUPD processes any file that starts with this prefix.

If the ALL statement is specified without operands, SYSUPD ignores any previous selection statements and retrieves all files residing on the tape.

VSN

=SYSTEM — Causes SYSUPD to recreate the specified files on public disks. Storage is allocated by DMS depending on the public space available.

=private

SYSUPD attempts to transcribe the selected files of the input tape to the private disk specified. Beforehand the operator **must** be informed of the volume serial number of the private disk and requested to ready it. (SYSUPD will request the operator to acknowledge the VSN entered by the user.)

The VSN must not exceed 6 characters and must not begin with the character string PUB.

VSN

=PUBnnn

Is reserved for system administration. SYSUPD attempts to accommodate all the selected tape files on the disk specified by system administration.

=SYSRES

Is reserved for system administration. SYSRES (sometimes called PUBRES) is the system residence, which includes the initial program loader (SYSBOOT), the file catalog (TSOSCAT), and the pageable storage area (PAGINGAREA).

If the VSN operand is omitted, SYSUPD retransfers the files onto public disks.

Note

When the VSN operand is used there should be no doubts as to the storage allocation of the files to be transferred. For if a certain file is directed to a volume despite already having reserved storage space on other public volumes, the spoolin is rejected.

TAPES=n

Specifies on how many input tapes the selected files reside. "n" can be any number from 1 to 10. The number of tapes must match the number of VSNs defined in the OPN statement.

The default value for "n" is 1 for unlabeled input tapes, and is equal to the number of VSNs specified in OPN for labeled tapes.

TAPES is essential when an input tape ends in a partial file whose remainder resides on a continuation volume. In this case SYSUPD will keep requesting additional volumes until the remainder of the partially read file has been found and can be read in.

DISPLAY statement

The DISPLAY statement is only meaningful in interactive mode. It requests all SYSUPD operations to be logged on the terminal.

The operations of every SYSUPD run are logged on SYSLST. On the terminal, the first 34 characters of each line of this listing are displayed.

The output listing contains the names of all the files requested plus a processing comment:

CREATED means the file has been added to the system.

REPLACED means an existing file of the same name has been replaced.

The attempt to overwrite files which are already in the system and which are protected by PASSWORD is rejected by SYSUPD with the message:

```
"DMSE-XXXX REJECTED <filename>"
```

Operation	Operands
DISPL[AY]	

Example

```
/START-SYSUPD

%  BLS0523 ELEMENT 'SYSUPLLM', VERSION 'V11.2N10' FROM LIBRARY
':1ONN:$TSOS.SYSLNK.SYSUPD.112' BEING PROCESSED
%  BLS0524 LLM 'SYSUPD', VERSION '11.2A' OF '1994-11-17:14:22:52' LOADED
%  BLS0551 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993. ALL
RIGHTS RESERVED
SYSUPD   CR DATE=19941021    VER=V11.2A00
FGG-SUPPORT=YES,SYSRES=1ONN.0,USER=D890950
USE LINKNAME MSTIN TO RELEASE TP DRIVE
ENTER CONTROLS
*
*opn savtap,vsnin=c2051a
%  DDE3 VOL C2051A IS MOUNTED FOR THE FILE T.A.P.E(8552I)
*all
*display
*end
STARTING I/O NOW
                    REPLACED HERL.B.B
                    REPLACED HERL.B.B8
                    REPLACED HERL.B.C
*SAVE TAPE* PROCESSED
/REMOVE-FILE-LINK MSTIN
```

END statement

The END statement terminates the input of statements to SYSUPD. Only after this statement has been issued will SYSUPD start transferring the files (exception: LST statement).
The message 'STARTING I/O NOW' indicates the start of file transfer.

Operation	Operands
END	

HELP statement

The HELP statement outputs to SYSOUT a survey of all the valid statements. The function of each statement is briefly explained.

Operation	Operands
HELP	

LST statement for tape initialization

The LST statement outputs a directory of all files of a tape. This directory is written to SYSLST; in interactive mode it can also be directed to SYSOUT.
Any previous statements are ignored. The directory is output immediately and the program run is terminated automatically (no END statement required).
In addition to a SAVTAP, system administration can open a MASTAP and output its directory.

Operation	Operands
LST	SAVTAP[,SYSOUT] [,FNIN=inputfilename] ,VSNIN={vsn|(vsn0,vsn1,...,vsn9)} [,D=P|G|TAPE-C1|T-C1|“device“]

SAVTAP — Is the readied user save tape which is mounted on the selected drive and opened.

SYSOUT — Is only relevant in interactive mode; it causes the tape directory to be written to SYSOUT (i.e. the terminal). If this entry is omitted, SYSUPD writes the directory to SYSLST.

FNIN=inputfilename — Specifies the name of the tape file and writes it into the HDR1 label of the reel; the name must not exceed 17 characters.

VSNIN=vsn — Defines the volume serial number of the user save tape; it is written into the VOL1 label of the reel and must not exceed 6 characters. This operand is mandatory.

Up to 10 VSN entries can be made in succession.

D — Specifies the recording density or device type with which the SAVTAP was generated. The recording density is measured in bpi (bits per inch).

=P — means 1600 bpi (9-track tape unit with PE recording method).

=G — means 6250 bpi (9-track GRZ unit).

={TAPE-C1 / T-C1} — means an 18-track cartridge device.

="device" — stands for the device type of the volume.
(Example: T9P / TAPE-C1 / ...)

Note

If magnetic tape cartridges are involved, specification of "device" is mandatory. The LST statement has requested a tape device. This device should be released as soon as possible and made available to other users. The corresponding command is

```
/REMOVE-FILE-LINK MSTIN
```

Example

```
/START-SYSUPD

%  BLS0523 ELEMENT 'SYSUPLLM', VERSION 'V11.2N10' OF LIBRARY
':1ONN:$TSOS.SYSLNK.SYSUPD.112' BEING PROCESSED
%  BLS0524 LLM 'SYSUPD', VERSION '11.2A' OF '1994-11-17:14:22:52' LOADED
%  BLS0551 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993. ALL
RIGHTS RESERVED
SYSUPD   CR DATE=19941021    VER=V11.2A00
FGG-SUPPORT=YES,SYSRES=1ONN.0,USER=D890950
USE LINKNAME MSTIN TO RELEASE TP DRIVE
ENTER CONTROLS
*1ST SAVTAP,SYSOUT,VSNIN=C2051A
TAPE SCAN ACTIVATED
%  DDE3 VOL C2051A IS MOUNTED FOR THE FILE T.A.P.E(8552I)
HERL.B.B
HERL.B.B8
HERL.B.C
*SAVE TAPE* PROCESSED
/REMOVE-FILE-LINK MSTIN
/
```

OPN statement for tape initialization

The OPN statement must be issued as the first SYSUPD statement when files are to be spooled in from a SAVTAP (user save tape).

OPN requests a tape unit and opens the readied SAVTAP. In addition to a SAVTAP, system administration can open a MASTAP and output its directory.

Operation	Operands
OPN	{ SAVTAP[,FNIN=inputfilename] ,VSNIN={vsn|(vsn0,vsn1,...,vsn9)} [,D=P|G|TAPE-C1|T-C1|“device“] / MASTAP }

SAVTAP — Is the readied user save tape which is mounted on the selected drive and opened.

FNIN=inputfilename — Is the tape file name which is written into the HDR1 label of the reel and must not exceed 17 characters.

VSNIN=vsn — Is the volume serial number of the user save tape. It is written into the VOL1 label of the reel and must not exceed 6 characters. This operand is mandatory.

Up to 10 VSN entries can be made in succession.

D — Specifies the recording density or device type with which the SAVTAP was generated. The recording density is measured in bpi (bits per inch).

=P — means 1600 bpi (9-track tape unit with PE recording method).

=G — means 6250 bpi (9-track GRZ unit).

={TAPE-C1 / T-C1} — means an 18-track cartridge device .

="device" — stands for the device type of the volume.

Note

If magnetic tape cartridge devices are involved, specification of "device" is mandatory. The OPN statement has requested a tape drive. This device should be released as soon as possible and made available to other users. The corresponding command is

```
/REMOVE-FILE-LINK MSTIN
```

Example

```
/START-SYSUPD

%  BLS0523 ELEMENT 'SYSUPLLM', VERSION 'V11.2N10' OF LIBRARY
':1ONN:$TSOS.SYSLNK.SYSUPD.112' BEING PROCESSED
%  BLS0524 LLM 'SYSUPD', VERSION '11.2A' OF '1994-11-17:14:22:52' LOADED
%  BLS0551 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1993. ALL
RIGHTS RESERVED
SYSUPD   CR DATE=19941021    VER=V11.2A00
FGG-SUPPORT=YES,SYSRES=1ONN.0,USER=D890950
USE LINKNAME MSTIN TO RELEASE TP DRIVE
ENTER CONTROLS
*opn savtap,vsnin=c2051a
%  DDE3 VOL C2051A IS MOUNTED FOR THE FILE T.A.P.E(8552I)
*sel herl.b.c
*end
STARTING I/O NOW
*SAVE TAPE* PROCESSED
/REMOVE-FILE-LINK MSTIN
/
```

PROTECT statement

Files protected by ACCESS=READ are normally overwritten by SYSUPD. If overwriting is to be avoided, the PROTECT statement can be issued.

Operation	Operands
PRO[TECT]	

REN statement

Like the SEL statement, the REN statement selects up to 10 input tape files that are to be spooled into the system. In addition, a new name is assigned under which the file(s) is/are to be cataloged.

The nonprivileged user processes SAVTAP files only, while system administration can use both SAVTAPs and a MASTAP for spoolin.

The nonprivileged user can specify whether the files are to be transferred to private or public disks. System administration can also specify the name of a particular public disk or the system residence (SYSRES) as destinations.

Operation	Operands
REN	(filename-old,filename-new)[,...] [,VSN=⟨SYSTEM / private- / PUBnn / SYSRES⟩]

filename — Is the fully qualified name of a file and must not exceed 40 characters. Nonprivileged users can access only their own files and must catalog files under their own user ID. System administration can access files with any user ID if "$userid." is prefixed to the file name.

Up to 10 file names may be specified. The user need not input the file names in the same sequence in which the files reside on the input tape.

VSN

=SYSTEM — Causes the files to be transferred to public disks. Storage is allocated by DMS depending on the public space available.

=private — SYSUPD attempts to transfer the selected input tape files to the private disk specified. Beforehand the operator **must** be notified of the VSN of the private disk and requested to start it. (SYSUPD will request the operator to acknowledge the VSN entered by the user.)

The VSN must not exceed 6 characters and must not begin with the character string PUB.

=PUBnnn — Is reserved for system administration. SYSUPD attempts to transfer all the selected tape files to the disk specified by system administration.

=SYSRES — Is reserved for system administration. SYSRES (often called PUBRES) is the name of the system residence. It includes such components as the initial program loader (SYSBOOT), the file catalog (TSOSCAT) and the pageable storage area (PAGINGAREA).

If the VSN operand is omitted, the files are transferred to public disks.

Note

If VSN is used, there should be no doubts as to the storage allocation of the files to be transferred. For if a certain file is directed to a volume when this file has already reserved storage space on other public volumes, spoolin is rejected.

SEL statement

This statement selects up to 10 input tape files.

The nonprivileged user can only process SAVTAP files, whereas system administration can use both SAVTAPs and a MASTAP for spoolin.

The nonprivileged user can specify whether the files are to be transferred to private or public disks. System administration can also specify the name of a particular public disk or the system residence (SYSRES) as destinations.

Operation	Operands
SEL	filename[,filename][,...] [,VSN={SYSTEM | private | PUBnn | SYSRES}]

filename — Is the fully qualified name of a file and must not exceed 41 characters. Nonprivileged users can access only their own files. System administration can access files under any user ID if "$userid." is prefixed to the file name.

Up to 10 file names may be specified. The user need not input the file names in the same sequence in which the files reside on the input tape.

VSN

=SYSTEM — Causes the files to be transferred to public disks. Storage is allocated by DMS depending on the public space available.

=private	SYSUPD attempts to transfer the selected input tape files to the private disk specified. Beforehand the operator **must** be notified of the VSN of the private disk and requested to ready it. (SYSUPD will request the operator to acknowledge the VSN entered by the user.) The VSN must not exceed 6 characters and must not begin with the character string PUB.
=PUBnnn	Is reserved for system administration. SYSUPD attempts to transfer all the selected tape files to the disk specified by system administration.
=SYSRES	Is reserved for system administration. SYSRES (often called PUBRES) is the name of the system residence. It includes such components as the initial program loader (SYSBOOT), the file catalog (TSOSCAT) and the pageable storage area (PAGINGAREA).

If the VSN operand is omitted, the files are transferred to public disks.

Notes

– TSOSMT saves files with names of no more than 50 characters. File names up to 50 characters long are processed in all SYSUPD statements apart from the REN statement, where only file names up to 40 characters long are accepted.

– If VSN is used, there should be no doubts as to the storage allocation of the files to be transferred. For if a certain file is directed to a volume when this file has already reserved storage space on other public volumes, spoolin is rejected.

16.3 SYSUPD error codes

Code	Meaning
00	Immediate termination
01	VSN error
02	SYSDTA EOF (no END statement)
03	Error occurred during processing
09	RDATA macro error
16	Password error on input tape
20	Input tape error ERRADDR (EXLST)
22	RECSIZE error on input tape
23	Error on disk OPENER (EXLST)
24	Error on disk ERRADDR (EXLST)
25	Error on disk COMMON (EXLST)
26	Error on input tape COMMON (EXLST)
28	Error on input tape OPENER (EXLST)
29	Error on input tape CLOSER (EXLST)

16.4 SYSUPD messages

`INPUT TAPE IS ALREADY OPENED.STATEMENT REJECTED`

Meaning
The tape file is already open. The program run continues.

`REJECTED-INVALID STATEMENT`

Meaning
Illegal operand in statement.

Response
Enter corrected statement.

`XXX REJECTED-FORMAT ERROR text`

Meaning
Syntax error in operand of statement.

Response
Enter corrected statement.

`INVALID FILENAME OR USERID`

Meaning
The relevant statement is rejected because of:

– partially qualified file name in SEL/REN
– fully qualified file name in ALL
– invalid/illegal user ID.

Response
Enter corrected statement.

`VSN REJECTED FOR FILE=filename`

Meaning
File has already been allocated space on volume other than requested.

Response
Cancel VSN or enter corrected VSN.

`PDUMP ACTIVATED`

Meaning
A prior message indicates cause of dump.

Response
Refer to dump for error diagnosis.

JOB-ABORTED

Meaning
An illogical condition forced program abortion.

Response
None; the message is issued for information only.

MASTER TAPE PROCESSED

Meaning
End of job.

Response
None; successful completion.

MASTER TAPE PROCESSED WITH ERRORS

Meaning
One or more statements have been rejected.

Response
Refer to final disposition listing for rejects.

SAVE TAPE PROCESSED

Meaning
End of job.

Response
None; successful completion.

SAVE TAPE PROCESSED WITH ERRORS

Meaning
One or more statements have been rejected.

Response
Refer to final disposition listing for rejects.

STATEMENT TABLE O/S-DUMP TAKEN

Meaning
Program error during sorting of statement table. Program terminates.

Response
Send dump to software developer.

`INPUT TAPE O/S—DUMP TAKEN`

Meaning
Files on MASTAP or SAVTAP are out of sequence.

Response
Review documentation on input tape generation or updating.

`STARTING I/O NOW`

Meaning
Indicates a delay in response; will occur during tape/disk input/output processing.

Response
Wait for "tape processed" message.

`NO DEVICE FOR TAPEIN`

Meaning
No magnetic tape unit is available for input.

Response
Inform operator via TYPE command that a tape drive is needed. Retry when notified of availability.

`DISC FILE OPEN ERROR`

Meaning
File cannot be opened. Request for update is rejected.

Response
Next message indicates file name. Enter /PASSWORD and /RESUME commands.

`DISC FILE CLOSE ERROR`

Meaning
Program terminates.

`DISC FILE PASSWORD ERROR`

Meaning
No password entered. Request for update is rejected. Next message indicates file name.

`DISC FILE LOCKED`

Meaning
File is open. Request for update is rejected.

Response
Next message indicates file name.

`DISC FILE COMMON EXIT TAKEN`

Meaning
Refer to disposition listing for cause. Program continues.

`DISC ERRADDR TAKEN`

Meaning
Branch to EXLST ERRADR. Request for update is rejected.

Response
Next message indicates file name.

`FILE REJECTED=filename`

Meaning
Information supplementing previous error messages.

Response
None.

`TAPEIN RECSIZE ERROR`

Meaning
Invalid record length found on input tape.

Response
Review documentation on input tape generation or updating. The name of the rejected file is contained in the final disposition listing.

`TAPEIN ERROR TAKEN:X:DB:ERRORS`

Meaning
Hardware error on the tape.

X:	Interrupted operation R=Read B=Backspace U=Unwind
DB:	SDB1 (standard device byte 1 from tape FCB)
ERRORS:	3 sense bytes

The affected file (on disk) is closed and erased.

`TAPEIN OPENER EXIT TAKEN`

Meaning
Invalid input tape. File cannot be opened.

Response
Refer to "DMS Tape Processing" manual for detailed explanation.

TAPEIN FILE LOCKED

Meaning
File is already being processed.

Response
Rerun.

TAPEIN PASSWORD ERROR

Meaning
File is password-protected.

Response
Refer to "DMS Tape Processing" manual for detailed explanation.

TAPEIN CLOSE EXIT TAKEN

Meaning
File cannot be closed.

Response
Refer to "DMS Tape Processing" manual for detailed explanation.

TAPEIN COMMON EXIT TAKEN

Meaning
Illogical condition exists.

Response
Rerun.

CATID PRESENT IN FILENAME. CM TERM

Meaning
File names containing CATID will not be processed.

*****CONTENTS INCOMPLETE REFER TO FOLLOWING CONTROL STATEMENT DISPOSITION LISTING

Meaning
One or more operations were rejected.

Response
Refer to final disposition listing on SYSLST or SYSOUT.

NO VALID FILENAMES

Meaning
Input from SYSDTA contains rejected operations or file names, or no file names were present.

Response
Rerun.

`ONLY SYS.ADMIN MAY USE "ALL" TO PROCESS THIS TAPE`

Meaning
Wrong input tape mounted.

Response
Instruct operator to ready the correct input tape.

`INPUT IS NOT A SAVTAP`

Meaning
Input may be a MASTAP, which cannot be processed with the ALL statement.

Response
Create MASTAP directory and process with SEL and REN statements.

`TAPE SCAN ACTIVATED`

Meaning
A directory for MASTAP or SAVTAP is being created by reading the tape.

Response
Wait for termination.

`filename IS FIRST-SPLIT FILE ON TAPE`

Meaning
1st file selected from tape is a partial file.

Response
Mount initial reel and restart job.

`ALL OTHER (ALL, SEL, REN) STATEMENTS IGNORED`

Meaning
An ALL statement without operands was entered, which causes the whole tape to be copied.

`INPUT STMNT IGNORED; EXCEPT "ALL" STMNT`

Meaning
When ALL statement is given without file name, all files on tape are transferred.

Response
None.

`INCORRECT VSNIN SPECIFIED,RERUN JOB`

Meaning
VSN specified does not tally with VSN on input tape label.

Response
Job is aborted.

```
INCORRECT VSNOUT SPECIFIED, RERUN JOB
```

Meaning
VSN specified does not tally with VSN on output tape label.

Response
Job is aborted.

```
INCORRECT FNIN SPECIFIED, RERUN JOB
```

Meaning
Specified input FN (tape file identifier) does not tally with FN on input file label.

Response
Job is aborted.

```
UNLABELLED INPUT TAPE MOUNTED AND VSNIN SPECIFIED, RERUN JOB AND/OR INFORM OPERATOR
```

Meaning
An input VSN is specified but an unlabeled tape is mounted. No VSN should be given for unlabeled tapes. If applicable, inform operator that the wrong tape has been mounted.

Response
Job is aborted.

```
FGG-SUPPORT=,SYSRES=vsn,USER=userid
```

Meaning
Message following SYSUPD initiation (for diagnostic purposes).

```
LAST TAPE ENDS WITH SPLITTED FILE. TAPEIN ERROR
```

Meaning
Last tape ends with an incomplete file.

Response
Job is aborted. The incomplete file is erased and processing terminated.

```
<filename> IS SPLITTED. IT NEEDS PRECEDING VOLUME
```

Meaning
A file at beginning of tape is incomplete. The file is not transferred. The program run continues.

```
OPN/LST FORMAT ERROR NO VSN GIVEN
```

Meaning
SAVTAP was specified in the statement without the VSNIN operand. At least 1 VSN must be given for SAVTAP.

Response
Rerun with corrected statement.

16.5 Disposition listing

This listing contains error flags and indicates actions taken as a result of the individual statements. For example, a file that has been successfully added (by means of ADD) is marked as ACCEPTED, and files which could not be added are marked as REJECTED, in which case the cause is also given.

Error flags in the disposition listing

Text	Meaning
DUP-FNOUT	Several input files were to receive the same output name.
DUP-FNIN	A file was to be transferred to disk several times.
VSN-CONFL	The file is on a disk other than specified in "VSN=".
NOT-FOUND	The file could not be located on the tape.
CR-INVNAM	The new name would exceed the maximum length allowed.
USERID-ER	Invalid/illegal user ID.
FNAME-ER	Fully qualified file name in an ALL statement, or partially qualified file name in SEL/REN.
TAPE-RDER	Hardware error on the tape.
DISK-WRER	Hardware error on the disk.
DISK-OPNR	Disk OPEN error.
DISK-CLSR	Disk CLOSE error.
DISK-PASS	Output file is password-protected.
DISK-LOCK	Output file is locked.
DUE TO PR	File could not be transferred (or statement could not be executed) because a previous severe error prevented any further processing.
FLERECON	The file is being reconstructed.
SPLIT END	The last tape specified ends with an incomplete (split) file. The continuation tape could not be readied. (The incomplete disk file is erased.)
SPLIT BEG	The 1st file on a tape was to be transferred but is incomplete (split).
DMSE-XXXX	The indicated DMS error occurred during file processing.

17 Appendix

The appendix contains the following tables:

- Device type code table
- Volume type table

17.1 Device type code table

F-C= FAMILY code

T-C = device type code

S/I = device channel class

S: block multiplexer channel type1 (SBL) or
byte multiplexer channel type 1 (SBY)

I: block multiplexer channel type 2 (IBL) or
byte multiplexer channel type 2 (IBY) or
block multiplexer channel type 2 extended distance (IBO) or
emulation of multiplexer channel type 2 (bus channel IE1) or
bit-serial channel type S (IBS)

Disk devices are never connected to byte multiplexer channels.
Magnetic tape devices can also be connected to byte multiplexer channels.

Device family	FAMILY name	F-C	S/I	T-C	Device type	Device name/ product number
Operator terminals	CONSOLE	00	S	02	CON3027	3027-1, -2 operator term. 3027-101, -102 operator term.
				03	CON3027C	3027-11, -21 operator term. 3027-111, -121 operator term. 3027-LRC operator term.
			I	0A	CON38	3809/3886 3884-1 cluster controller 75407-3, -4, -5
				0B	CON3803	75407-1, 3803-90, 3886-2, -3 (hardcopy unit on SVP)
				0C	CON3888	3888-3 hardcopy unit (for 3886 operator subterminal) on cluster controller 75407-1

Table 8: Device type code table

Device family	FAMILY name	F-C	S/I	T-C	Device type	Device name/ product number
Printers	PRINTER	20	S	26	PRLS333	3337-51, 3338-51, -511, -512, 3339-51, -512, 3348-110, 3349-110
			I	2A	PRLI333	3338-531, -53, -532, 3339-53, -532
			S/I	21	STDPRINT	all other printers
Special devices	FAM50	50	S/I	51	DSVP1	SVP hard disk
			I	52	DSVP2	SVP hard disk on the C40
				54	CTRL-DEV	CONTROL-DEVICE for PFA controller
				55	SCD	Type S connection director
				57	3920	Radio clock

Table 8: Device type code table

Device family	FAMILY name	F-C	S/I	T-C	Device type	Device name/ product number
Remote data processing system	TD	60	S	53	TD8170	8170-21 Local Cluster Controller
				61	TD960	9631-1,-2,-3
				62	ZAS-DUMP	9631-50,-51,-52,-55
				63	ZAS-BCAM	
				6C	ZAS-SIN	TRANSDATA ZAS with connection to SINIX
				6D	ZAS-LAN	9632-100
				6E	DAST	3612
			I	61	TD960	9631-1,-2,-3
				62	ZAS-DUMP	9631-60,-61,-62,-65
				63	ZAS-BCAM	
				64	SKP	
				6C	ZAS-SIN	TRANSDATA ZAS with connection to SINIX
				6D	ZAS-LAN	9632-200, 91848-M (ZAS-FDDI), HNC 91849 (high-speed net connect)
				6E	DAST	3801-B
Physically supported devices				71 . . 7F	Name of the physically supported device	The names are defined by ADAM. They are assigned to the device type codes with the UGEN statement ADT.
Disk storage devices	DISKETTE	90	S	93	FD3171	3171 with 31712 option [1)]
			I	9B	FD75407	75407-2 (C40)

Table 8: Device type code table

Device family	FAMILY name	F-C	S/I	T-C	Device type	Device name/ product number
Disk storage devices [2)]	DISK	80/ A0				
		80	S/I	81	D3434-10	3434-10, -11, -12
				82	D3434-20	3434-20, -21, -22
				83	D3438-30	3438-31, -32 34211-11, -14
				86	D34211-2	34211-21, -24
			I	85	D3409	Solid State Disk on 3409-26, -46 3411-2
				89	D3490-30	3490-3A4, -3A8, -3B4, 3490-3B8, -3BC
				8A	D3490-40	3490-4A4, -4A8, -4B4, 3490-4B8, -4BC
				8E	D3492	3492-141, 34921-121
				8F	D3475-8F	74305-12, -13, -140, -141, 74305-150, -151 75435

Table 8: Device type code table

Device family	FAMILY name	F-C	S/I	T-C	Device type	Device name/ product number
Disk storage devices [2]	DISK	80/ A0				
		A0	S/I	A1	D3439-10	3439-10,-12
				A2	D3436	3436, 3436-2,-10,-12
				A3	D3437	3437, 3437-2
				A4	D3438-20	3438-20,-232,-22
			I	A5	D3435	3435
				A7	D3490-10	3490-1A4, -1A8, -1B4, 3490-1B8, -1BC
			S/I	AB	D3475	3475-1,-2,-3
			I	AC	D3480	3410 [3] (external high-speed memory)
			S/I	AC	D3480	3480-1, -2, -11, -12, -111, -112 3848-A4, -B4, -AD4, -BD4
				AD	D348E	3480-21, -22 3848-AE4, -BE4
			I	AE	D348F	3480-131,-132
				AF	D3490-20	3490-2A4, -2A8, -2B4, 3490-2B8, -2BC

Table 8: Device type code table

Device family	FAMILY name	F-C	S/I	T-C	Device type	Device name/ product number
Tape devices	TAPE	B0/ C0/ E0				Controller, + drive unit, + subsystem +
Unimodal tape devices	UNMTAPE	B0	S	B4	UM6250	3513 + 3557, 3559 [4]
			I	B4	UM6250	3514 + 3557, 3559 [4]
			S	B7	UM1600-1	3534
			I	B9	UMVID-1	MTC 2,1 Gbytes Video 8
				BA	UMSC-1	MTC, 155 Mbytes (for SIR and ARCHIVE only)
Magnetic tape cartridge devices	MBK	C0	I	C1	3580	3580-A10 + 3580-B10 3580-A20 + 3580-B20 3585-L01 3585-L02 3586-M01 3586-M02 3590-D31 3590-D32
				C2	3590	3580-A10 + 3580-B10 [5] 3580-A20 + 3580-B20 [5] 3590-D31 [6] 3590-D32 [6] 3590-A01 + 3590-B02/-B04 3590-A02 + 3590-B04/-B04
				C4	3590E	3590-D41 3590-D42 3590-C10 3590-C11 3590-C1A 3590-C22 3590-C2A 3590-A10 +3590-B20/-B40 3590-A20 +3590-B20/-B40
Optical disks			I	C4	3590E (emulated)	9083-B1 9083-B2 9083-B3

Table 8: Device type code table

Device family	FAMILY name	F-C	S/I	T-C	Device type	Device name/ product number
Tape devices	TAPE	B0/ C0/ E0				Controller, + drive unit, + subsystemt +
Bimodal tape devices	BIMTAPE	E0	S	E2	BM1662	3513 + 3557, 3559 7)
			I	E2	BM1662	3514 + 3557, 3559 7)
			S	E2	BM1662	3515 + 3525 3516 + 3526 3517-1 + 3527-1 3519 + 3529 3535 + 3525 3536 + 3526 3537-1 + 3527-1
			I	E2	BM1662	3517-3 + 3527-3 3519-3 + 3529
			S	E3	BM1662S	3518 + 3528 3538 + 3528
			I	E4	BM1662S1	3506 (C40)
				E8	BM1662FS	3504-625, 3505-P/SD

Table 8: Device type code table

1) For these floppy devices, one CTL and two DVC statements (with addresses in ascending order) must be issued at system generation time.

2) For system storage devices, one DVC statement per drive must be issued at system generation time.

3) For the 3410 External High-Speed Storage Unit, the DYNREC=NO operand must be specified in the CTL statement.

4) For these devices, the value MBS must be specified with the typ operand in the CTL statement.

5) With 35830 Option for improved recording (IDRC, Improved Data Recording Capability)

6) With 35930 Option for improved recording (IDRC, Improved Data Recording Capability).

7) With 35506 Option
For these devices, the value MBS must be specified with the “type” operand in the CTL statement.

17.2 Volume type table

Volume type code	Volume type	Meaning
B2	T1600	Tapes with a recording density of 1600 bpi (device type codes: E2, E3, E4, E8)
B4	T6250	Tapes with a recording density of 6250 bpi (device type codes: B4, E2, E3, E4, E8)
B8	WORK / TAPE	Tapes with a recording density of 1600 or 6250 bpi
B5	TAPE-C1	18-track magnetic tape cartridge (device type codes: C1, C2)
B6	TAPE-C2	18-track magnetic tape cartridge, compressed (device type code: C2)
BB	TAPE-C3	36-track magnetic tape cartridge (device type code: C4)
BC	TAPE-C4	36-track magnetic tape cartridge, compressed (device type code: C4)
B9	TAPE-V1	Magnetic tape cartridge, 2.1 Gb, Video 8 (device type code: B9)
BA	TAPE-CS1	Magnetic tape cartridge 155 Mbytes (device type code: BA)
BD	OD-1	Optical disk (emulated)
BE	OD-WORM1	Optical disk (WORM, emulated)

Tabelle 9: Volume types

List of figures

References

[1] **BS2000/OSD-BC V2.0**
Commands, Volume 1, A-L
User Guide

Target group
The manual addresses both nonprivileged BS2000/OSD users and systems support.
Contents
This manual contains BS2000/OSD commands ADD-... to LOGOFF (basic configuration and selected products) with the functionality for all privileges. The introduction provides information on command input.

[2] **BS2000/OSD-BC V2.0**
Commands, Volume 2, M-SG
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
This manual contains BS2000/OSD commands MODIFY-... to SET-... (basic configuration and selected products) with the functionality for all privileges.

[3] **BS2000/OSD-BC V2.0**
Commands, Volume 3, SH-Z
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
This manual contains BS2000/OSD commands SHOW-... to WRITE-... (basic configuration and selected products), with the functionality for all privileges. By means of SDF-P users of SHOW commands can make use of output in structured S variables which are described in Volume 4.

[4] **BINDER V1.1A** (BS2000/OSD)
(BS2000/OSD)
User Guide

Target group
Software developers
Contents
The manual describes the BINDER functions, including examples. The reference section contains a description of the BINDER statements and BINDER macro. BINDER V1.1A can also be used in BS2000 V10.0A.

[5] **BS2000/OSD-BC V1.0**
Dynamic Binder Loader / Starter
User Guide

Target group
This manual is intended for software developers and experienced BS2000/OSD users
Contents
It describes the functions, subroutine interface and XS support of the dynamic binder loader DBL, plus the method used for calling it. It also includes a description of the commands for calling the static loader ELDE.

[6] **BS2000/OSD-BC V2.0**
Diagnostics Handbook
User Guide

Target group
The manual addresses system programmers, systems support and software maintenance.
Contents
The manual describes tools for identifying, logging and analyzing program execution data. It deals with dump generators (CDUMP, SNAP, SLED), dump analyzers (DAMP, SODA), logging tools (SERSLOG, AUDIT), the TRACE-MANAGER, and the log evaluator ELFE.

[7] **BS2000/OSD-BC V2.0**
DMS Macros
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
The manual describes the DMS macro interface for the BS2000/OSD basic configuration. There is a brief description of the access method-specific features relevant to programming, followed by a description of the macros in alphabetical order.

[8] **BS2000/OSD-BC V2.0**
DMS Introductory Guide
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
The manual describes file processing in BS2000, focussing on:
- file and catalog management
- files and data media
- file and data protection
- OPEN, CLOSE and EOV processing
- DMS access methods (SAM, ISAM ...).

[9] **EDT** (BS2000/OSD)
Statements
User Guide

Target group
EDT newcomers and EDT users
Contents
Processing of SAM and ISAM files and elements from program libraries and POSIX files.

[10] **JV V11.2A** (BS2000/OSD)
Job Variables
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
The manual describes management and possible uses of job variables. The command descriptions are divided according to function areas. The macro calls are described in a separate chapter.

[11] **LMS** (BS2000/OSD)
SDF Format
User Guide

Target group
BS2000 users.
Contents
Description of the statements for creating and managing PLAM libraries and the members these contain.
Frequent applications are illustrated with examples.

[12] **BS2000/OSD-BC V2.0**
Executive Macros
User Guide

Target group
The manual addresses all BS2000/OSD assembly language programmers.
Contents
The manual contains a summary of all Executive macros, detailed descriptions of each macro with notes and examples, including job variable macros, and a comprehensive general training section.

[13] **SDF V4.0A** (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.

[14] **BS2000/OSD-BC V2.0**
Introductory Guide to Systems Support
User Guide

Target group
The manual addresses BS2000/OSD systems support and operators.
Contents
The manual contains the following topics on management and monitoring of hte BS2000/OSD basic configuration: and monitoring: system introduction, parameter service, job and task control, memory/device/user/file management, assignment of privileges, accounting and operator functions.

[15] **BS2000/OSD-BC V2.0**
System Installation
User Guide

Target group
BS2000/OSD system administration
Contents
This manual describes
– the generation of the hardware and software configuration with UGEN
– the following installation services:
  – disk organization with MPVS
  – program system SIR
  – volume installation with SIR
  – configuration update (CONFUPD)
  – utility routine IOCFCOPY

[16] **BS2000/OSD-BC V2.0**
System Messages Volume 1
User Guide

Target group
The manual addresses systems support, operators and users.
Contents
Chapter 1 deals with message processing in BS2000/OSD.
Chapter 2 contains the system messages for the basic configuration of the BS2000/OSD operating system. The messages are arranged in alphabetical order by message classes.

[17] **BS2000/OSD-BC V2.0**
System Messages Volume 2
User Guide

Target group
The manual addresses systems support, operators and users.
Contents
This reference work contains the system messages for the basic configuration of the BS2000/OSD operating system in alphabetical order, arranged by message classes.

[18] **MSGMAKER V1.1B** (BS2000/OSD)
Creating and Editing BS2000 Message Files
User Guide

Target group
The 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 message system.

[19] **DUALCOPY V1.0A** (BS2000/OSD)
Redundant Data Recording (RAID1)
User Guide

Target group
Systems support and service technicians.
Contents
The DUALCOPY subsystem supports dual recording in 3860-43/51 controllers (RAID1). The manual describes the installation, operation and user interfaces of the subsystem. Detailed descriptions are provided of error recovery for read, write and disk errors and of controller maintenance during active operation

Ordering manuals

The manuals listed above and the corresponding order numbers can be found in the Siemens Nixdorf *List of Publications*. New publications are described in the *Druckschriften-Neuerscheinungen (New Publications)*.

You can arrange to have both of these sent to you regularly by having your name placed on the appropriate mailing list. Please apply to your local office, where you can also order the manuals.

Index

D

E

F

P

R

V

W

X

Z

Contents

BS2000/OSD-BC V2.0A

Utility Routines

Target group
The manual addresses both nonprivileged users and systems support.
Contents
The manual describes the utilities: CONDMPPD V11.2A, DPAGE V11.2A, INIT V11.2A, JMU V11.2A, LMSCONV V1.0B, PAMCONV V11.0A, PDPOOLS V11.2A, PVSREN V1.1A, RFUPD V11.0A, SODA V11.2A, SPCCNTRL V11.2A, TPCOMP2 V11.2A, VOLIN V11.2A.

Edition: July 1995

File: DIENSTP.PDF

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.