1 Preface

The complexity of BS2000 processing has been reduced by breaking BS2000/OSD down into functional units (subsystems). SSCM and DSSM are the two software products with which all subsystems are declared and managed.

The static subsystem catalog (SSMCAT) is managed with SSCM (Static Subsystem Catalog Management). The declarations required for all subsystems belonging to a configuration are stored in the subsystem catalog. SSMCAT forms the database for DSSM.

DSSM (Dynamic Subsystem Management) is the central instance in BS2000/OSD for dynamic subsystem management.

The versions on which this description is based are DSSM V4.0 and SSCM V2.3.

1.1 Target group and summary of contents

This manual is intended for BS2000/OSD systems support staff.

This chapter “**Preface**” provides a general overview of the manual, a list of the changes made since DSSM V3.6 and SSCM V2.1, and a description of the SDF syntax used for the DSSM commands and SSCM statements.

The manual is divided into the following main chapters:

The chapter “**The subsystem concept in BS2000/OSD**” explains concepts and describes relations that play an important role in the subsystem concept. A description of the version dependencies between BS2000/OSD, DSSM and SSCM is followed by an overview of important DSSM-compatible products included in the basic configuration of BS2000 and by brief descriptions of selected unbundled DSSM-compatible products.

The chapter “**DSSM**” describes dynamic subsystem management (DSSM):
topics dealt with include tasks and functions, management strategies and error handling, how to activate DSSM via the parameter service at startup time, and the accounting records of DSSM.
The DSSM commands are described in detail.
The chapter is concluded by two comprehensive examples.

The chapter “**SSCM**” describes subsystem catalog management (SSCM). Following a brief introduction, the SSCM statements are described in detail. The chapter concludes with notes on the installation of SSCM and a number of examples.

At the back of the manual you will find a list of related publications and an index.

Other publications referred to in the text are given in the form of abbreviated titles accompanied by a number in square brackets. The full title of each of these publications can be found under “Related publications” as of page 293.

1.2 Changes since the edition December 1996 (DSSM V3.6 and SSCM V2.1)

The main changes with respect to the previous version are listed below:

New functions in DSSM V4.0

- The new SHOW-SUBSYSTEM-ATTRIBUTES command informs the user about the attributes of global and local subsystems. The information can be output to SYSLST, SYSOUT or in S-variables, see page 122.
- The SHOW-SUBSYSTEM-STATUS command shows the internal status of a subsystem more accurately than previously for subsystem status IN-CREATE, IN-RESUME, IN-DELETE or IN-HOLD, see page 142.
- The new MONJV operand in the START-SUBSYSTEM command allows subsystems to be monitored with a monitoring job variable. MONJV indicates whether the subsystem is active, stopped, interrupted or locked, see page 153.
- The new value *HIGHEST in the VERSION operand can be used with the following commands to select the highest version of the subsystem that is entered in the static subsystem catalog:
  - HOLD-SUBSYSTEM
  - RESUME-SUBSYSTEM
  - START-LOCAL-SUBSYSTEM
  - START-SUBSYSTEM
  - STOP-LOCAL-SUBSYSTEM
  - STOP-SUBSYSTEM
- Connection to non-privileged subsystems is improved by dynamically supplying the entry points from the BLS name list at load time.

- The improved error handling during a system run allows the operator to, e.g. specify a new, valid name for a file that was not found and then continue the system run (see page 64).

New functions in SSCM V2.3

- With the new GENERATE-CATALOG-SOURCE statement, SSCM creates a file containing a list of all SSCM statements that are required for (re)generating a subsystem catalog.
- The definitions of all versions of the specified subsystem can be deleted from the catalog by specifying VERSION=*ALL in the REMOVE-CATALOG-ENTRY statement.
- Using the new value *BY-PROGRAM(...) in the SUBSYSTEM-ENTRIES operand of the SET-SUBSYSTEM-ATTRIBUTES statement supplies the entries of the specified subsystems dynamically at load time from the BLS name list. The *BY-PROGRAM(...) setting concerned can be modified in the MODIFY-SUBSYSTEM-ATTRIBUTES statement.

1.3 Readme file - modifications incorporated in this version

Information on any functional changes and additions to the current product versions described in this manual can be found in the product-specific README files. You will find the README files on your BS2000/OSD system under the file names:

`SYSRME.`*product.version.language*

The user ID under which the README files are cataloged can be obtained from your system support personnel. With IMON you can determine the filename with the following command:

`/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=`*product*`, LOGICAL-ID=SYSRME.`*language*

You can view the README files using /SHOW-FILE or an editor, and print them out on a standard printer using the following commands:

`/PRINT-DOCUMENT` *filename*`, LINE-SPACING=*BY-EBCDIC-CONTROL`

1.4 Notational conventions

The following notational conventions are used in this manual:

i For drawing your attention to particularly important information.

- Terms in continuous text that are to be particularly highlighted are shown in **bold**.
- In the examples, **`bold type`** indicates a user entry and this `typewriter font` is used for system outputs.
- Short names and numbers are used for references to other publications. The complete title of each publication is contained next to the number in the list of related publications at the back of the manual.

1.5 SDF syntax representation

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

HELP-SDF	Alias: **HPSD**
GUIDANCE**-MODE** = ***NO** / ***Y**ES	
,**SDF-COM**MANDS = ***NO** / ***Y**ES	
,**ABBR**EVIATION**-RULES** = ***NO** / ***Y**ES	
,**GUID**ED**-DIA**LOG = ***Y**ES (...)	
***Y**ES(...)	
| **SCREEN-STEPS** = ***NO** / ***Y**ES	
| ,**SPEC**IAL**-FUNC**TIONS = ***NO** / ***Y**ES	
| ,**FUNC**TION**-KEYS** = ***NO** / ***Y**ES	
| ,**NEXT-FIELD** = ***NO** / ***Y**ES	
,**UNGUID**ED**-DIA**LOG = ***Y**ES (...) / ***NO**	
***Y**ES(...)	
| **SPEC**IAL**-FUNC**TIONS = ***NO** / ***Y**ES	
| ,**FUNC**TION**-KEYS** = ***NO** / ***Y**ES	

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

Table 1: Notational conventions

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

Table 2: Data types

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

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

Table 3: Suffixes for data types

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

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

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

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

Metasyntax

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

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

Table 1: Metasyntax (part 1 of 2)

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

Table 1: Metasyntax (part 2 of 2)

Data types

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

Table 2: Data types (part 1 of 6)

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

Table 2: Data types (part 2 of 6)

Data type	Character set	Special rules
filename (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 $\left\{\begin{matrix}\text{(*abs)}\\ \text{(+rel)}\\ \text{(-rel)}\end{matrix}\right\}$ (*abs) absolute generation number (1-9999); * and parentheses must be specified. (+rel) (-rel) relative generation number (0-99); sign and parentheses must be specified.
integer	0…9, +, -	+ or -, if specified, must be the first character.
name	A…Z 0…9 $, #, @	Must not begin with 0...9.

Table 2: Data types (part 3 of 6)

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

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

Table 2: Data types (part 4 of 6)

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

Table 2: Data types (part 5 of 6)

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

Table 2: Data types (part 6 of 6)

Suffixes for data types

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

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

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

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

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

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

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

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

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

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

Suffix	Meaning
with-wild-constr (continued)	Examples:

Auswahlmuster	Auswahl	Konstruktionsmuster	neuer Name
A//*	AB1 AB2 A.B.C	D<3><2>	D1 D2 D.CB
C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<3>.XY<2>	G.A.D.XYA G.A.D.XYB G.B.F.XYA G.B.F.XYB
C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<2>.XY<2>	G.A.A.XYA G.A.B.XYB G.B.A.XYA G.B.B.XYB
A//B	ACDB ACEB AC.B A.CB	G/XY/	GCXYD GCXYE GCXY. 1) G.XYC

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

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

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

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

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

2 The subsystem concept in BS2000/OSD

In this chapter, important terms are explained and then information is provided for

- managing subsystems with DSSM and SSCM (see page 26)
- local subsystem management (see page 27)
- the version dependencies between BS2000/OSD, DSSM and SSCM (see page 28)
- the main DSSM-compatible products in the BS2000 basic configuration (see page 31)
- selected, unbundled DSSM-compatible products (see page 33).

2.1 Definitions

In BS2000/OSD, both global subsystems that are valid throughout the system and local subsystems can be grouped together to form a local subsystem configuration and administered task-local in a local subsystem catalog. Whenever reference is made in the following to subsystems, configuration and catalog, what is meant is always the global subsystems valid throughout the system, and their configuration and catalog. Local subsystems, their configuration and catalog are always identified by the word “local”.

Subsystem

Within the context of dynamic subsystem management (DSSM), a subsystem is a unit that executes a function and that can be loaded, started and terminated automatically and independently, taking into account dependency relations to other subsystems.
A subsystem may consist of a number of subsystem components.

Each subsystem is identified uniquely to DSSM by its name and version number.
A subsystem is defined for DSSM by specifying its components and attributes.

The attributes of a subsystem provide information on:
- identification (name and version)
- access privileges
- loading and processing mechanisms
- environment and dependency relations
- required address spaces
- call-up preferences
- existing call entries

The relations between different subsystems are marked by
- address space separations
- link relations
- functional dependencies
- holder task sharing

Subsystem components

The term “subsystem components” is used to denote the subsystem module (program module) and the ancillary components, known as “subsystem satellites”.
The subsystem module is a component that is absolutely essential for installing the subsystem. A subsystem may be composed of a number of different modules.
Satellites for subsystems include REP files, NOREF files, message files, syntax files and information files; these files must be declared during installation and will be required to activate the subsystem.

Subsystem call entries

The subsystem call entry is the visible point of entry through which the subsystem is accessible to user tasks.
A call entry is characterized by its name, type, scope and lifetime. A subsystem may have a number of call entries with different attributes. They are administered by DSSM.

Subsystem configuration

A subsystem configuration is the set of all subsystems that are to be available in a session. Since only one subsystem configuration can be operated in the system at a given time, this configuration must contain all the subsystems that have to be available in the system at the same time.

A subsystem configuration is defined by its subsystems and their components and attributes.

A subsystem (or subsystem version) may be compatible within a given subsystem configuration and in a certain combination, i.e. the subsystem can run in the system with certain other subsystems. A subsystem may be optional or mandatory, i.e. the subsystem configuration may or may not run without this subsystem.

When the system is in operation, a given subsystem configuration may be in one of two states:

a) Load state:
the subsystems of the configuration are currently **active**, i.e. can actually be used;

b) Declaration state:
the subsystems or subsystem versions of the configuration are defined in the subsystem catalog, i.e. **could be activated**.

Several different load states can be derived from a declaration state.

Subsystem catalog

Static subsystem catalog management (SSCM) generates a static subsystem catalog (SSMCAT) in which the subsystem configuration is defined.
This subsystem catalog is loaded by dynamic subsystem management (DSSM) when the system is started up. Once loaded, the static subsystem catalog becomes a dynamic subsystem catalog.
During the session the subsystem catalog can be managed by means of the DSSM commands ADD-SUBSYSTEM, MODIFY-SUBSYSTEM-PARAMETER and REMOVE-SUBSYSTEM.

Both SSCM and DSSM use the data structure of the static subsystem catalog.
The subsystem catalog can contain up to 1000 subsystems with maximum totals of 16,000 call entries, 16,000 functional dependencies and 200 address space restrictions.

2.2 Administering subsystems with DSSM and SSCM

The purpose of dynamic subsystem management (DSSM) is to reduce the complexity of BS2000 operation by dividing the operating system into functional units (subsystems).

DSSM supports the following as subsystems:

– software products (if expressly designated as DSSM-compatible, see tables 4 and 5 as of page 31),
– system exit routines and
– shared products.

In addition, system administration can make customized TU programs and system exit routines DSSM-compatible and activate them as subsystems.

DSSM functions distinguish between privileged subsystems (TPR subsystems), which are part of BS2000/OSD, and nonprivileged (or unprivileged) subsystems (TU subsystems), such as user programs.
DSSM also distinguishes between subsystems that are loaded in system address space and those that are loaded in user address space. TPR subsystems are always loaded in system address space, whereas TU subsystems can be loaded either in system address space or in user address space.

The SSCM subsystem enables flexible and user-friendly management of the static subsystem catalog (SSMCAT).

The SSD object

The SSD object is an ISAM file with a key length of 11 bytes. It contains the definition(s) of one or more subsystems (but not the definitions of different versions of one and the same subsystem).
Each of these definitions contains attributes, call entries, references and dependencies of the relevant subsystem, as well as information on disjunctive subsystems and the holder task.

The SSD object itself cannot be loaded. Before it can be activated, it must be linked into a static subsystem catalog using the SSCM statement ADD-CATALOG-ENTRY.

2.3 Local subsystem management

In BS2000/OSD it is possible to create and administer a local subsystem configuration on a task-local basis. A local subsystem catalog can be loaded and unloaded, and local subsystems can be activated and deactivated.
Local subsystem management is restricted to nonprivileged subsystems.

A **local subsystem configuration** consists of a group of subsystems which are available locally in the user address space. These are known as **local subsystems**.
Any nonprivileged subsystem can be used as a local subsystem.

The parts making up a local subsystem configuration may include both subsystems or subsystem versions which are not contained in the global subsystem configuration and those which are already available in the global subsystem configuration. If the concerned Subsystem is already included in the global subsystem configuration, it must be additionally defined as a local subsystem for the calling task.

It is necessary to create a **local subsystem catalog** containing the subsystem definitions (SSD) of the required local subsystem configuration.
Only nonprivileged (TU) subsystems can be included in this catalog. Once a subsystem has been defined as a local subsystem it has the following characteristics:

– it is always loaded in the user address space (MEMORY-CLASS=*LOCAL-UNPRIVILEGED)
– it is always loaded immediately at the time of its request by the START-LOCAL-SUBSYSTEM command (CREATION-TIME=*AT-CREATION-REQUEST)
– it can be halted and unloaded at any time (SUBSYSTEM-HOLD=*ALLOWED)
– no more than one version of the same subsystem may be active at any one time (VERSION-COEXISTENCE=*FORBIDDEN)
– loading the subsystem in exchange mode is not permitted (VERSION-EXCHANGE=*FORBIDDEN)

These characteristics are independent from the characteristics that the same subsystem has in the global subsystem catalog.

Provision of local subsystems

The local subsystem catalog is created with the SSCM statement START-CATALOG-CREATION, and the subsystem definitions (SSD) of the required local subsystem configuration are added to the catalog by means of the ADD-CATALOG-ENTRY statement.

The subsystem catalog that has been created is loaded for the local task with the LOAD-LOCAL-SUBSYSTEM-CATALOG command. As a result, the local subsystems included in it are available, but not yet activated. Activation of the local subsystems is brought about by means of the START-LOCAL-SUBSYSTEM command.

When the task then calls a subsystem, an attempt is initially made to set up a link to the local subsystem in the local subsystem catalog. If the subsystem is not contained in the local catalog - in other words it is not a local subsystem - or if the local subsystem catalog is not loaded, a connection is established to the global subsystem.

The SELECT-PRODUCT-VERSION command provides the option of dynamic assignment of priorities to the various versions of a subsystem.

Local subsystems can be deactivated and the local subsystem catalog can be unloaded with the STOP-LOCAL-SUBSYSTEM and UNLOAD-LOCAL-SUBSYSTEM-CATALOG commands respectively.

For more details of working with local subsystems refer to the example on page 170.

2.4 Version dependencies between BS2000/OSD, DSSM and SSCM

Which SSCM and DSSM versions are available under the various BS2000 versions?

BS2000/OSD-BC V3.0:	with DSSM V3.6	SSCM V1.0, V2.0, V2.1
	with DSSM V3.8	SSCM V1.0, V2.0, V2.1, V2.2
	with DSSM V3.9	SSCM V1.0, V2.0, V2.1, V2.2, V 2.3A
	with DSSM V4.0	SSCM V1.0, V2.0, V2.1, V2.2, V 2.3A, V 2.3B
BS2000/OSD-BC V3.1	with DSSM V3.7	SSCM V1.0, V2.0, V2.1
(in OSD-SVP V1.0/V2.0):	with DSSM V3.8	SSCM V1.0, V2.0, V2.1, V2.2
	with DSSM V3.9	SSCM V1.0, V2.0, V2.1, V2.2, V 2.3A
	with DSSM V4.0	SSCM V1.0, V2.0, V2.1, V2.2, V 2.3A, V 2.3B
BS2000/OSD-BC V4.0:	with DSSM V3.8	SSCM V1.0, V2.0, V2.1, V2.2
	with DSSM V3.9	SSCM V1.0, V2.0, V2.1, V2.2, V2.3A
	with DSSM V4.0	SSCM V1.0, V2.0, V2.1, V2.2, V 2.3A, V 2.3B
BS2000/OSD-BC V5.0:	with DSSM V4.0	SSCM V1.0, V2.0, V2.1, V2.2, V 2.3A, V 2.3B

There are one or two particular DSSM version(s) for each BS2000/OSD version. However, higher DSSM versions up to V3.9 can run without problems on lower BS2000/OSD versions. BS2000/OSD V3.0 or higher is required for DSSM V4.0.

Which catalog format of the subsystem catalog (corresponds to the SSCM version) is evaluated by the various DSSM versions?

SSCM V1.0:	as of DSSM V3.0
SSCM V2.0:	as of DSSM V3.5
SSCM V2.1:	as of DSSM V3.6
SSCM V2.2:	as of DSSM V3.8
SSCM V2.3:	as of DSSM V3.9

The creation or modification of a subsystem catalog is not dependent on the DSSM version that is currently running.
However, each DSSM version is able to evaluate only those catalog formats which have been created by its “own” SSCM version or a lower SSCM version.
The full functionality of a DSSM version is never exploited unless it is run with the associated SSCM version.

The catalog format of a SSCM version is supported by higher DSSM and SSCM versions for compatibility reasons, and it is converted to the higher format when it is read and written back.

Example

When a catalog generated with SSCM V2.1 is read and written back by DSSM V4.0 or SSCM V2.3, it is given the format of SSCM V2.3.

Under what conditions can SSCM versions V1.0 to V2.3B be loaded at the same time?

Up to DSSM V3.4, the appropriate SYSPRC.SSCM.0xx procedure must be used to load the higher version.

Example

In order to allow SSCM V2.3 to be loaded when SSCM V1.0 is in use as a subsystem, the SYSPRC.SSCM.023 procedure must be used; this loads SSCM V2.3 as a program.

As of DSSM V3.5, the version can be selected with the SELECT-PRODUCT-VERSION command. The START-SSCM command then starts the selected version.

However, as the full functionality of a particular DSSM version can only be exploited with its “own”, specially developed SSCM version, loading older SSCM versions at the same time does not provide any advantages for the current BS2000 session.

On the other hand, parallel loading of different SSCM versions is necessary if a subsystem declaration file SYSSSC or SYSSSD is to be generated for an SSCM/DSSM version that is older than the current version (e.g. for version downgrading).

2.5 Overview of important DSSM-compatible products in the BS2000 basic configuration

MC:	Memory classes (3-6) in which the product can be loaded
TU/TPR:	Processor state of the subsystems (TU: nonprivileged / TPR: privileged)
Rf.:	Reference in the list of related publications as of page 293

Product	Function	MC	TU/TPR	Rf.
ACS	Support for file access via aliases	3,4	TPR	14
ADAM	Operation of devices which are not supported by the logical access methods of BS2000/OSD	3,4	TPR	1
AIDSYS	System-related part of AID (Advanced Interactive Debugger; cf. table 5)	3,4	TPR	-
BINDER	Binding of a compiled source program with other object modules and LLMs to create a loadable unit	4,6	TU	5
CALENDAR	Creation of user-specific calendars	3,4	TPR	6
DIV	Object-oriented access method used especially for processing unstructured data	3,4	TPR	13
FASTPAM	Block-oriented access method for NK4 disk files	3,4	TU/TPR	13
GET-TIME	Provision of the date, standardized world time and local time	4	TU	14
GSMAN	Main memory management	3,4	TPR	14
IMON	Monitor for software installation			17
IMON	(IMON-BAS) Installation and registration of software	4	TPR	-
IMON-GPN	Support for the allocation of logical names and path names of files	4	TPR	-
INIT	Initialization of magnetic tapes, MTCs and floppy disks	4,6	TU/TPR	9
MIP	Management and output of system messages	4	TPR	9
NDM	Device management			14
NKS	Monitoring of resource reservations	4	TPR	-
NKV[T] NKV[D]	Monitoring of mounted data volumes (T=Tape; D=Disk)	4	TPR	-
NKISAM	Record-oriented access method for files with “DATA” block format (without separate PAM key)	4	TPR	13
PAMCONV	Conversion of file formats	4,6	TU	9

Table 4: DSSM-compatible products in the BS2000 basic configuration (part 1 of 2)

Product	**Function**	**MC**	**TU/TPR**	**Rf.**
PCA	Buffering of data outside the operating system in a disk control unit (hardware caching)	4	TPR	24
POSIX-BC	Basic configuration component of POSIX	3,4	TPR	26 27
POSPRRTS	Runtime functions for C (in TPR)	4	TPR	
SDF	System Dialog Facility; BS2000/OSD language interface	4	TU	35
SDFSYS	System-related part of SDF (discontinued after SDF V3.0)	4	TPR	35
SDF-CONV	Conversion of procedure formats	6	TU	33
SDF-P-BASYS	Basic configuration component of SDF-P (for information on SDF-P see table 5)	4	TPR	19
SHOW-FILE	Screen display of file contents	3, 4	TPR	20
SIR	Software installation	4	TPR	40
SPOOL	Input/output control for certain device families			
PRMPRES	Creation and management of print resources in BS2000 SPOOL (component of PRM)	4	TU	28
PRMMAN	Creation and management of print resources in BS2000 SPOOL (component of PRM)	4	TPR	28
SPOOL	Organization of spoolin/spoolout and management of print jobs	4	TPR	38
SRPMNUC	System Resources and Privilege Management (basic configuration component of SRPM; for information on SECOS see table 5)	4	TPR	35
SSCM	Generation of subsystem catalogs	4,6	TU, p.179ff	
SYSFILE	Support of the system files SYSLST and SYSOUT	4	TPR	19
TANGRAM	Allocation of related task groups to processors in accordance with performance requirements			
TANGRAM	Regulating function	3,4	TPR	14
TANGBAS	Management of task groups	3,4	TPR	-
TSDRIVER	Synchronization of system time with the time server	4	TPR	-
VOLIN	Formatting and initialization of hard disks	4,5	TPR	9
WARTOPT	Monitoring of the maintenance task running under the user ID HARDWARE-MAINTENANCE	3,4	TPR	14

Table 4: DSSM-compatible products in the BS2000 basic configuration (part 2 of 2)

2.6 Overview of selected unbundled, DSSM-compatible products

MC:	Memory classes (3-6) in which the product can be loaded
TU/TPR:	Processor state of the subsystems (TU: nonprivileged / TPR: privileged)
Rf.:	Reference in the list of related publications as of page 293

Product	Function	MC	TP/TPR	Rf.
AID	Test and diagnostics aid (Advanced Interactive Debugger)			
AID	Symbolic and non-symbolic debugging	4	TPR	2
LLMAID	Information on link and load modules	4	TPR	-
ARCHIVE	Saving, reconstruction and transfer of data in files and job variables	4	TPR	3
CRTE	Runtime functions for C and COBOL	4,6	TU	7
DAB	Caching in BS2000/OSD to avoid bottlenecks (software caching)	3,4	TPR	8
Dprint	(Distributed Print Services) Printing in computer networks			
DPRINTCL	Client part for Dprint: creation of print jobs	4	TPR	10
DPRINTCM	Base mechanisms: implementation of general services	4	TPR	10
DPRINTSV	Server part for Dprint: management of print jobs	4	TPR	10
DRV	Recording procedure for maintaining duplicate disks	4	TPR	11
DUALCOPY	Enhancing the reliability of systems and data security by means of redundant recording	4	TPR	14
EDT	Editor for SAM, ISAM and POSIX files and for elements of program libraries	4,6	TU	12
FDDRL	Physical data backup of disks and pubsets	4,5	TPR	15
HIPLEX MSCF	Creation of a computer network on the basis of the BCAM data communication network	3,4	TPR	23
HSMS	Backup, archival and reconstruction of data and support for data management on external storage units	4	TPR	16
HSMS-SV	Operation of HSMS-CL (HSMS client in SINIX)	4	TPR	16
IMON-XT	SHOW functions for IMON (for information on IMON see table 6)	4,6	TU	17
JV	Control of jobs and programs by means of job variables	4	TPR	18

Table 5: DSSM-compatible, unbundled products (selection) (part 1 of 2)

Product	Function	MC	TP/TPR	Rf.
LMS	Library management	4,6	TU	21
MAREN	Management of data on archive volumes in BS2000 computer centers	3,4	TPR	22
PCS	Tool for systems support in ensuring the best possible setup and operation of the BS2000 system	3,4	TPR	25
PROP-XT	Computer center automation and user-specific problem-solving in computer centers	4	TPR	29
RSO	Control of the output of remote SPOOL jobs on decentralized (RSO) printers (only executable in conjunction with SPOOL)	4	TPR	31
SDF-A	Management and modification of the SDF user interface	6	TU	32
SDF-P	Procedure and variable concept, commands for procedure control, block-oriented error handling	4	TPR	34
SDF-P-BIF	Builtin function of SDF-P	4	TPR	34
SECOS	Access security control for BS2000/OSD			
GUARDS	Management of objects (guards) and evaluation of access conditions (component of GUARDS)	4	TPR	36
GUARDDEF	Management of standard guard as well as the attribute and objects paths	4	TPR	36
GUARDCOO	Management of co-ownership guard	4	TPR	36
SATCP	Monitoring of events and alarms (component of SAT)	4	TPR	36
SRPMOPT	(Component of SRPM)	4	TPR	36
SM2	Monitoring system for capturing and evaluating data on BS2000/OSD performance and the level of utilization of system resources	3,4	TPR	37
SPS	BS2000 SPOOL add-on component and printer driver for APA printers	4	TPR	39

Table 5: DSSM-compatible, unbundled products (selection) (part 2 of 2)

3 DSSM

The purpose of dynamic subsystem management (DSSM) is to reduce the complexity of BS2000 execution by dividing the operating system into functional units (subsystems).

Subsystems have the following features:

- each subsystem constitutes a self-contained unit
- subsystems can be activated, suspended, resumed and deactivated during the BS2000 session
- If a new version has been created for a subsystem, the old version can be exchanged for the new one or both versions can be operated in parallel.

DSSM is the central facility for BS2000/OSD subsystem configuration management. DSSM controls the loading, initialization, suspension, resumption and termination of subsystems during the session. DSSM can modify the current system dynamically (on-line) by incorporating subsystems in the subsystem catalog or removing them from it, without having to suspend and restart the system as a whole.

DSSM is mandatory for BS2000/OSD since important components of the basic configuration (see product overviews on page 31) can be operated only with DSSM.

DSSM V4.0 is available for BS2000 versions from BS2000/OSD-BC V3.0 onwards. Subsystem catalogs can be generated as of SSCM V1.0 for DSSM V4.0.

For information on the compatibility and portability of DSSM with regard to SSCM and BS2000/OSD, see page 28.

DSSM command privileges

As part of the BS2000 privilege concept, a system-global privilege for execution of DSSM commands exists. The privilege to carry out subsystem management is referred to in commands and messages as SUBSYSTEM-MANAGEMENT. This privilege is allocated by default to the TSOS user ID after first startup. However, when SECOS is being used, the privilege can be assigned to any desired user ID; this user ID will then have the sole authority to issue these commands for execution (see also the “SECOS” manual [36]).

The following table contains an overview of all DSSM commands and the privileges required for command execution.

Command	**Meaning**	**Required privilege**				**Page**
		SM	**O**	**STD**	**HM**	
ADD-SUBSYSTEM	Extend dynamic subsystem catalog	X				75
HOLD-SUBSYSTEM	Place a subsystem in wait state	X	X			80
LOAD-LOCAL-SUBSYSTEM-CATALOG	Load a local subsystem catalog			X		83
MODIFY-SUBSYSTEM-PARAMETER	Modify subsystem parameters	X				85
RELEASE-SUBSYSTEM-SPACE	Release reserved address space for subsystems	X		X	X	110
REMOVE-SUBSYSTEM	Remove inactive subsystem from dynamic catalog	X				111
RESUME-SUBSYSTEM	Cancel wait state for subsystem	X	X			113
SAVE-SUBSYSTEM-CATALOG	Save changes to dynamic subsystem catalog	X				116
SET-DSSM-OPTIONS	Activate/deactivate DSSM logging function	X	X			120
SHOW-SUBSYSTEM-ATTRIBUTES	Request information on subsystem attributes	X		X		122
SHOW-SUBSYSTEM-INFO	Request information on current subsystems configu-ration	X				138

Continued ➠

Command	**Meaning**	**Required privilege**				**Page**
		SM	**O**	**STD**	**HM**	
SHOW-SUBSYSTEM-STATUS	Request information on status of subsystems	X				142
subsys=*NON-PRIV-CLASS-5				X		
subsys=*ALL / <subsys-name>			X	X		
START-LOCAL-SUBSYSTEM	Activate local subsystem in user address space			X		150
START-SUBSYSTEM	Activate subsystem	X	X			153
STOP-LOCAL-SUBSYSTEM	Deactivate local subsystem in user address space			X		159
STOP-SUBSYSTEM	Deactivate subsystem	X	X			162
UNLOAD-LOCAL-SUBSYSTEM-CATALOG	Unload local subsystem catalog			X		166
UNLOCK-SUBSYSTEM	Shift subsystem from LOCKED status to NOT-CREATED status	X				168

Table 6: DSSM commands and required privileges

SM:	SUBSYSTEM-MANAGEMENT privilege	For managing subsystems
O:	OPERATING privilege	For entry from operator terminals
STD:	STD-PROCESSING privilege	For executing user commands
HM:	HARDWARE-MAINTENANCE privilege	For calling hardware test programs

3.1 Purpose and functions of DSSM

The software product DSSM performs the following individual functions:

- DSSM sets up the entire subsystem configuration (such that it is capable of functioning).
- DSSM activates and deactivates subsystems in order to expand the subsystem configuration or adapt it to current requirements.
- DSSM checks the compatibility of individual subsystem versions when activating and deactivating subsystems.
- DSSM establishes the relations between subsystems and monitors them when activating subsystems or dissolves existing relations to other subsystems when deactivating subsystems.
- DSSM supports the linkage of tasks and programs to subsystems.
- DSSM supports the use of shared address space in memory pools.

All activities indicated below as being synchronous run in the task of the requester, while all asynchronous activities run in the holder task.

3.1.1 Subsystem declaration (SSC)

In order to manage the subsystems, DSSM requires information in the form of a declaration. This declaration determines which main components and satellites belong to a subsystem, how the subsystem can be activated, and how to set up interfaces to it. Thus, anyone creating a declaration must be familiar with the subsystem concerned. For this reason the necessary declarations for all DSSM-compatible subsystems are supplied in a declaration file.

Depending on the declaration, DSSM activates the subsystem and sets up interfaces and relations to other subsystems and to DSSM itself. Interfaces to a subsystem can be monitored by DSSM. In this way it is possible to deactivate the subsystem under the control of DSSM. If the interfaces are not monitored, the subsystem itself is responsible for deactivation.

The declarations are not given as operands in every DSSM call, as this would be too complicated and excessive. The declarations are specified using SSCM and are stored in the subsystem catalog, because

– these declarations are stable and rarely change
– these declarations are complex and extensive
– these declarations must frequently be accessed by DSSM
– not only internal subsystem information exists, but also cross-subsystem information.

The declarations of all subsystems belonging to a configuration (= declaration state) make up the SSMCAT subsystem catalog and are stored in a PAM file. This file is read in on startup and remains in memory until system shutdown. The ADD-SUBSYSTEM command can be used to extend the catalog during a session.

The LOAD-LOCAL-SUBSYSTEM-CATALOG command can be used to read the catalog and dynamically load it into the user address space of the calling task.

3.1.2 Activation and restart

Activation of a subsystem is controlled by the attributes that were defined in the subsystem catalog using the SSCM statement SET-SUBSYSTEM-ATTRIBUTES.

Activation is performed

a) automatically during the first DSSM call in the startup routine if CREATION-TIME= *BEFORE-DSSM-LOAD or *AT-DSSM-LOAD was defined; activation must have been successfully completed (*BEFORE-DSSM-LOAD means that the subsystem had already been loaded before DSSM was first called by the startup routine. After DSSM is called, this subsystem will be managed like any other. Version exchange or version coexistence with another version defined with CREATION-TIME=*AT-DSSM-LOAD is permitted. It is the responsibility of systems support to ensure that one version of the subsystem is available at all times.)

b) automatically during the second DSSM call in the startup routine if CREATION-TIME= *MANDATORY-AT-STARTUP was defined; in this case, activation must have been successfully completed.

c) automatically during the second DSSM call in the startup routine if CREATION-TIME=*BEFORE-SYSTEM-READY was defined (synchronous).

d) automatically after the second return to the startup routine if CREATION-TIME=*AFTER-SYSTEM-READY was defined (asynchronous).

e) explicitly by means of the START-SUBSYSTEM command.

f) explicitly as a local subsystem by means of the START-LOCAL-SUBSYSTEM command.

g) explicitly when the $ESMCRE interface is called.

h) implicitly after the first SVC call for a subsystem which was defined with CREATION-TIME=*AT-SUBSYSTEM-CALL(ON-ACTION=*STD or *ANY)
or implicitly after the first ISL call for a subsystem that was defined with CREATION-TIME=*AT-SUBSYSTEM-CALL(ON-ACTION=*ISL-CALL or *ANY).

With reference to the start point for subsystems see also the SSCM statement SET-SUBSYSTEM-ATTRIBUTES as of page 243.

In the same way as the other files required for DSSM initialization (e.g. SSMCAT) and the files required for activation of subsystems linked to startup, the program or object module library (OML) and the DSSM REP and NOREF files must have been created on the home pubset under the TSOS user ID, and must be available at startup.
Subsystems which are linked to startup are those whose activation points (creation points) are specified by the following values:

– *BEFORE-SYSTEM-READY
– *AFTER-SYSTEM-READY
– *BEFORE-DSSM-LOAD
– *AT-DSSM-LOAD
– *MANDATORY-AT-STARTUP

An indispensable precondition for activation is that the subsystem name is known, i.e. declared in the global or local subsystem catalog.

Activation takes place in the following steps:

1. The job is checked, in particular the dependencies on other subsystems (synchronous).

   If a subsystem has components which were declared with the attribute *INSTALLED and installed using IMON, DSSM calls IMON-GPN during the checking phase in order to find out the path names of the files.

   Depending on the availability of IMON-GPN and on the status of the installed installation unit (see the INSTALLATION-UNIT operand), DSSM acts as follows:

   a) if the installation unit exists and a file name can be linked to the specified logical ID:

      DSSM will use this file name as long as the subsystem is active (until STOP-SUBSYSTEM) or until another file name is defined by means of the MODIFY-SUBSYSTEM-PARAMETER command.

   b) In the following situations, DSSM accesses the DEFAULT-NAME defined in the catalog, if there is one:

      – IMON-GPN is not available
      – the INSTALLATION-UNIT does not exist in the IMON-GPN data
      – the INSTALLATION-UNIT exists in the IMON-GPN data but has no connection to a logical name (LOGICAL-ID)

      The message `ESM0665` explains what is happening:

      ```
      ESM0665  'DEFAULT-NAME' USED FOR FILES OF SUBSYSTEM '(&00)'
      ```

c) The INSTALLATION-UNIT exists in the IMON-GPN data with a few appropriate path names but the queried logical name does not exist. DSSM now assumes that the file does not exist. Two cases are to be discriminated:

    1. Subsystems that were defined with the *AT-DSSM-LOAD or *MANDATORY-AT-STARTUP attribute:
       During startup, a query that must be answered is issued to the operator terminal. The operator can now either specify a new, valid name for the file concerned (information file, module library or REP file with the REP-FILE-MANDATORY=*YES attribute) or stop the subsystem activation.

    2. Subsystems that were defined with the *AT-CREATION-REQUEST, *AFTER-SYSTEM-READY or *BEFORE-SYSTEM-READY attribute:
       If one of the files (module library, information file or REP file with the REP-FILE-MANDATORY=*YES attribute) is missing, the subsystem is not activated and this is indicated with a message. If the message file or REP file (with the REP-FILE-MANDATORY=*NO attribute) is missing, the subsystem is activated but a warning is still output.

2. The subsystem is loaded in a holder task (asynchronous)
   Loading of the subsystem code is initiated by the holder task.
   In addition it makes its local address space (user address space) available for the subsystems that were defined with MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED. The loading of subsystems with MEMORY-CLASS=*SYSTEM-GLOBAL is also initiated by the holder task; however, they are not loaded into the holder task's user address space but into the shareable system address space.

   When TU subsystems are activated with MEMORY-CLASS=*BY-SLICE, the shareable program area is loaded into the shareable system address space, and the non-shareable program area and/or data area into the user address space of the holder task.

   If a task establishes a connection to a subsystem of this type, only the data area that has already been loaded in the user address space of the holder task is copied to the same address in the private user address spaces of the connected tasks. This means that address-related references between the program area and data area are always possible. Performance is considerably enhanced by this method of address space distribution because no access is made to the program library or object module library when a task is connected to the subsystem, and external references do not have to be resolved.

If a subsystem was defined with MEMORY-CLASS=*BY-SLICE and is started for the first time, DSSM informs the BLSSERV subsystem that the copy of the data area in the private user address space can be accessed with the VSVI1 macro.
The VSVI1 macro informs the user about entries in the DBL tables. See the manual “BLSSERV” [4] for details on the macro.

The holder task can also be used as the work task.

3. For TPR subsystems only: subsystem activation is started in the holder task if an INIT routine has been defined (asynchronous).
4. After completion of activation, the subsystem is opened for connections of authorized users (asynchronous).

Restarting a subsystem (RESUME-SUBSYSTEM) after a HOLD-SUBSYSTEM consists of steps 1, 3 and 4.
In step 1 it is not necessary to call IMON-GPN.

If the activation or restart of a subsystem is initiated explicitly via the START-SUBSYSTEM or RESUME-SUBSYSTEM command, the synchronous processing mode can be selected instead of the asynchronous mode.

3.1.3 Interface establishment and cancelation

The interface to the job entry point of a subsystem is implemented in one of the following ways:

a) implicitly and globally by linkage (this is possible only for interfaces to subsystems that have already been loaded or between subsystems that are loaded at the same time)

b) implicitly and task-specifically via
   - subsystem-specific SVC, ISL, bISL or system exit routines
   - in the case of nonprivileged subsystems, BLS interfaces (BIND macro, START-PROGRAM, LOAD-PROGRAM and START-<product-name>, autolink)

c) explicitly and task-specifically via internal system macros ($ESMCON and $ESMCCS).

Establishing an interface to a subsystem includes:

1. generating a subtask following a corresponding internal system call ($ESMCCS)
2. attachment to the memory pool, if required

3. If a subsystem was defined with MEMORY-CLASS=*BY-SLICE, the private area is copied into the local address space (user address space). When the subsystem is started for the first time, DSSM informs the BLSSERV subsystem that the copy in the private user address space can be accessed with the VSVI1 macro.
4. transferring the address to the task or calling the subsystem code
5. setting the subsystem-specific interface marker
6. incrementing the task-specific interface counter (except when defining the subsystem with the attribute CONNECTION-SCOPE=*FREE).

It is also possible to set up an interface for job entries which the subsystem manages itself.

A relation can be **canceled** in one of these ways:

a) implicitly and task-specifically
   - by program/task termination
   - after a return from the subsystem job entry point if this was defined with the attribute MODE=*SVC/*ISL and CONNECTION-SCOPE=*CALL
   - after deactivation of subsystems with CONNECTION-SCOPE=*OPTIMAL

b) explicitly and task-specifically via an internal system macro on termination of an SVC routine ($ESMDCN)

c) implicitly and subsystem-specifically on deactivation of a subsystem which was defined with the attribute CONNECTION-SCOPE=*FREE.

Canceling a relation includes:

1. detachment from the memory pool
2. If the subsystem was defined with MEMORY-CLASS=*BY-SLICE, the part in the user address space of the connected task is unloaded.
   When the last connection is shut down, DSSM informs the BLSSERV subsystem that this private part can no longer be accessed.
3. resetting the task-specific interface marker
4. decrementing the subsystem-specific interface counter (except for subsystems defined with the attribute CONNECTION-SCOPE=*FREE)
5. the “detach” function in the case of an explicit call.

The address supplied to the task at the time of interface setup can no longer be used.

3.1.4 Subsystem Deactivation or suspension

A precondition for this function is that the subsystem is active.

A subsystem can be **deactivated** in the following ways:

a) explicitly when the STOP-SUBSYSTEM command is entered (HOLD-SUBSYSTEM suspends the subsystem)

b) explicitly when privileged macros are called at the program interface ($ESMDEL and $ESMHLD)

c) implicitly when a START-SUBSYSTEM command is entered with the operand VERSION-PARALLELISM=*EXCHANGE-MODE

d) automatically at shutdown, for all subsystems whose definitions include such a declaration (STOP-AT-SHUTDOWN=*YES)

Deactivation of a subsystem (STOP-SUBSYSTEM) takes place in the following steps:

1. The job is checked, in particular the dependencies on other subsystems (synchronous).
2. The CLOSE-CTRL routine is started, provided one is defined (asynchronous).
3. The subsystem is closed for new users (asynchronous), preventing the connection of any further tasks to the subsystem (except for those with entry points with CONNECTION-SCOPE=*FREE or SVC/ISL calls for subsystems with entry points with CREATION-TIME=*AT-SUBSYSTEM-CALL).
   From this moment on, it is no longer possible to access code that has an entry point defined with CONNECTION-SCOPE=*OPTIMAL.
4. The job termination routine (STOPCOM routine), if defined, is started (asynchronous).
5. Wait until the subsystem is jobless, i.e. the subsystem-specific connection counter is 0 and no task is accessing code that has an entry point defined with CONNECTION-SCOPE=*OPTIMAL (asynchronous). Exception: if the operand FORCED=*YES is specified in the DSSM command, deinitialization is started immediately.
6. Deinitialization, if defined, is started (asynchronous).
7. Unloading from the holder task (asynchronous).

A subsystem **hold** (HOLD-SUBSYSTEM) consists of steps 1 through 6.
If the deactivation or suspension of a subsystem is initiated explicitly via the STOP-SUBSYSTEM or HOLD-SUBSYSTEM command, the synchronous processing mode can be selected instead of the asynchronous mode.

3.1.5 Swapping subsystem versions

It is possible to swap versions of a subsystem in the following three ways:

a) Deactivate the old version (STOP-SUBSYSTEM) and activate the new version of the subsystem (START-SUBSYSTEM). Under certain circumstances, this may result in very long subsystem downtimes because the new version is not activated until the old one has been completely deactivated, i.e. when all tasks have cleared their links.

b) Activate the new version of the subsystem with START-SUBSYSTEM ...,VERSION-PARALLELISM=*EXCHANGE-MODE. From this time onward, no new links to the old version of the subsystem are set up. While the last tasks are clearing their links to the old version of the subsystem, the new version is initialized. This method substantially shortens the period of subsystem unavailability.

   Provided they have been defined, the following routines are called one after the other:
   - the STOPCOM routine of the old version,
   - the INIT routine of the new version and
   - the DEINIT routine of the old version.

   For some subsystem it is potentially a problem that the DEINIT routine of the old version is running while the new version has already been activated and is running.
   A subsystem that was defined with VERSION-EXCHANGE=*FORBIDDEN cannot be swapped in as a new version. It can, however, be deactivated (as the old version) in exchange for a new version that has been defined with VERSION-EXCHANGE=*ALLOWED.

   The old version remains in the IN-DELETE state until there is no further task connected. If the new version is in the CREATED state, activation of the old subsystem with RESET=*YES is only possible if coexistence was approved for both versions at the time of definition.

   Activation of the old version with RESET=*YES is allowed if the new version is in the IN-DELETE state and the old version was not defined with VERSION-EXCHANGE=*FORBIDDEN.

c) The CLOSE-CTRL routine can be used to switch versions without interrupting subsystem availability.
   Provided they have been defined, the following routines are called one after the other:
   - the CLOSE-CTRL routine of the old version,
   - the INIT routine of the new version,
   - the STOPCOM routine of the old version and
   - the DEINIT routine of the old version.

   If the initialization routine of the new version does not run correctly, the old version is automatically reactivated and is in the CREATED state (the result of the CLOSE-CTRL routine is reversible), thus avoiding any interruption of subsystem availability.

Version swapping is allowed if a version of the subsystem is in the CREATED state and all other versions of the subsystem that are declared in the catalog are in the NOT-CREATED or LOCKED state.
A version exchange will not be executed if all declared versions of the subsystem are in the NOT-CREATED or LOCKED state. In this case the version that is activated is the one that was specified in the START-SUBSYSTEM command.

Example

Subsystem versions SPOOL V04.2.A and V04.3.A are defined with VERSION-EXCHANGE=*ALLOWED.

```
/SHOW-SUBSYSTEM-STATUS SPOOL,*ALL
SUBSYSTEM SPOOL    /V04.2.A IS NOT CREATED
SUBSYSTEM SPOOL    /V04.3.A IS NOT CREATED

/START-SUBSYSTEM SPOOL,04.2.A,VERSION-PARAL=*EXCHANGE-MODE,SYNCH=*YES
ESM0220  FUNCTION 'CREATE' FOR SUBSYSTEM 'SPOOL /V04.3.A' COMPLETELY
         PROCESSED
ESM0400  'CREATE' OR 'RESUME' SUBSYSTEM 'SPOOL /V04.3.A' WITH
         'SYNCHRONOUS=YES' AND 'RESET=NO'
ESM0220  FUNCTION 'CREATE' FOR SUBSYSTEM 'SPOOL /V04.3.A' COMPLETELY
         PROCESSED
```

If there are one or more versions of the subsystem which are not in the NOT-CREATED or LOCKED state (apart from the version in the CREATED state that is to be exchanged), the exchange will be rejected, even if all versions allow coexistence.

Example

Subsystem versions UTM V04.0, V05.0 and V05.1 are defined with VERSION-COEXISTENCE=*ALLOWED and VERSION-EXCHANGE=*ALLOWED.

```
/SHOW-SUBSYSTEM-STATUS UTM,*ALL
SUBSYSTEM UTM    /V04.0    IS NOT RESUMED
SUBSYSTEM UTM    /V05.0    IS NOT CREATED
SUBSYSTEM UTM    /V05.1    IS CREATED

/START-SUBSYSTEM UTM,05.0,VERSION-PARALLELISM=*EXCHANGE-MODE
ESM0206  SOME ACTIONS IN PROGRESS FOR SUBSYSTEM 'UTM/V04.0'.
         NO FURTHER ACTION ON ANOTHER VERSION POSSIBLE
ESM0224  REQUESTED FUNCTION 'CREATE' FOR SUBSYSTEM 'UTM/V05.0'
         REJECTED
```

When the old version of the subsystem is replaced with a new one, the syntax file of the new version is also loaded. This means that the syntax of the new version must also recognize and execute commands and statements of the old version, i.e. it must be ensured that the syntax of the new version supports that of the old version.

It is advisable to use the CLOSE-CTRL routine to swap versions only when the new version which is to be activated is also the higher of the two.

3.1.6 Coexistence of versions

DSSM has offered the option of keeping two versions of a subsystem active, temporarily or permanently, to permit all the tasks which are linked to the old version to continue working. This coexistence mode must be specified when the subsystem is defined (using the SSCM statement SET-SUBSYSTEM-ATTRIBUTES ...,VERSION-COEXISTENCE=*ALLOWED), and must be explicitly requested in the START-SUBSYSTEM command.

If a **temporary coexistence** is in effect and version B of a subsystem is loaded whilst version A of the subsystem is active, all new callers will be connected to version B. Jobs which are connected to version A will still be processed. When all the jobs which use version A have been processed, this version will automatically be terminated.
It should be noted that, in the definition, the “old” version which is being replaced must not be dependent on the “new” version which replaces it.

If a **permanent coexistence** of different versions of a subsystem is required, then the appropriate attribute must be specified in the definition for each of the subsystems, and this mode must be explicitly requested in the START-SUBSYSTEM command.

3.1.7 Relations between subsystems

DSSM acts as the central mechanism responsible for managing the interrelations between the subsystems, allowing various relations to be explicitly monitored.

Job relations to a subsystem via explicit or implicit connections are set up and monitored by DSSM. A job relation can always be established between a task and a job entry.

Address relations via linkage editors and loaders must be specified explicitly in an operand of the SSCM statement SET-SUBSYSTEM-ATTRIBUTES (REFERENCED-SUBSYSTEMS). However, the operand should only be applied to subsystems from the same “family” (privilege level, memory class, etc.). In the case of unbundled subsystems, system administration should only select dynamic addressing mechanisms (SVC, ISL or system exit mechanisms for privileged subsystems; the BLS interface, BIND, for nonprivileged subsystems).

Address relations restrict the unloadability of a subsystem, since unloading has to take place in the reverse order from loading. Address relations are taken into account during loading and unloading, but they are not monitored. They prevent a subsystem from being unloaded, regardless of whether or not jobs exist.

Dependency relations require the availability of another subsystem. These relations can be declared by means of the RELATED-SUBSYSTEMS operand of the SSCM statement SET-SUBSYSTEM-ATTRIBUTES and are taken into account in the loading and unloading sequences.

3.1.8 Relations between subsystem satellites and subsystems

During the installation of a subsystem, the references to its satellites are created.

The **module library** is a mandatory satellite, as no subsystem can be loaded without it.

If an **information file** or a **syntax file** is specified in the definition, they will also be given the status of mandatory components, which will prevent the subsystem from being loaded if they are missing.

If a **REP file** is specified in the definition, this will only be mandatory at load time if this has been explicitly stated in the definition (REP-FILE-MANDATORY=*YES).

A **message file** can be specified in the definition, but this is not a mandatory component. In other words, loading of the subsystem can be carried out even if the message file is missing.

In following situations, DSSM sends a query to the operator terminal that must be answered by the operator:

– the module library of a mandatory subsystem is missing (*AT-DSSM-LOAD or *MANDATORY-AT-STARTUP attribute)
– the specified information is declared but not present
– the REP file with the REP-FILE-MANDATORY=*YES attribute is missing

Despite a missing file, the operator still has the option of continuing the startup process (e.g. starting the subsystem without the information file). The operator is exclusively responsible for a possible abnormal subsystem or system termination.

3.1.9 Communication between subsystems and DSSM

The exchange of information and messages is essential for subsystem-specific routines for initialization, deinitialization, job termination and the check on job termination (INIT, DEINIT, STOPCOM and CLOSE-CTRL routines).

The communications area always consists of an information area for the started routine (DSSM → subsystem) and an acknowledgment area (subsystem → DSSM).

The routine is started via

a) procedure call
during initialization (with no condition) or at deinitialization, job termination and the check on job termination if the holder task is not being used as a work task

b) the interface (bourse or FITC) transferred during initialization
if the holder task is being used as a work task

Acknowledgment to DSSM is given via a procedure return with an acknowledgment area at deinitialization, job termination and the check on job termination if the holder task is not or is no longer being used as a work task. If the holder task is being used as a work task, DSSM is notified via a NOTIFY call with the acknowledgment area serving as an input parameter.

3.1.10 Information about subsystems

Users can request information about the status of a subsystem by means of the SHOW-SUBSYSTEM-STATUS and SHOW-SUBSYSTEM-INFO commands. They can also obtain an overview of the overall subsystem configuration.

3.1.11 Status of a subsystem

NOT-CREATED
The subsystem has been declared in the current system, but has not yet been activated by a START-SUBSYSTEM command, or has been deactivated again since being activated. Tasks cannot access this subsystem until it has been activated.

IN-CREATE
The subsystem is being activated, but loading and initialization have not been completed. Tasks cannot access this subsystem yet.

CREATED
The subsystem has been loaded and initialized. Tasks can access the subsystem.

IN-DELETE
The subsystem has been deactivated by a STOP-SUBSYSTEM command. Unloading and deinitialization have not yet been completed.
Other tasks can no longer access this subsystem. Processing of tasks still connected to the subsystem will be completed.

IN-HOLD
The subsystem has been suspended by a HOLD-SUBSYSTEM command. Deinitialization has not yet been completed.
Other tasks can no longer access this subsystem. Processing of tasks still connected to the subsystem will be completed.

IN-RESUME
The subsystem is being resumed by a RESUME-SUBSYSTEM command. Reinitialization has not yet been completed. Tasks cannot yet access this subsystem.

NOT-RESUMED
The subsystem has been suspended by a HOLD-SUBSYSTEM command. Deinitialization has been completed. Tasks cannot access this subsystem until a RESUME-SUBSYSTEM command has been issued and successfully executed.

LOCKED
An unrecoverable error has occurred while the subsystem was active or was being activated, deactivated, resumed or suspended. Any further attempt to issue the corresponding commands will be rejected.

The following situations can put a subsystem into the LOCKED status:

– if the change to this status is specified in the INIT, DEINIT, STOPCOM or CLOSE-CTRL routine (applies only to privileged subsystems);
– if the subsystem’s holder task has terminated abnormally and either no restart is provided for this task, or it cannot be executed (see page 72) or
– if a problem occurs in deactivating the old version during a version swap that entails an interruption of subsystem availability; regardless of the value of the RESTART operand, the subsystem is in the LOCKED status; activation of the new version is continued.

When a subsystem is activated, deactivated, suspended or resumed, its status changes, i.e. it progresses from an initial status to a final status (for example, when a subsystem is activated, its final status is CREATED).
The initial status for a request may not always be the same; e.g. a START-SUBSYSTEM command is acceptable for a subsystem with the status NOT-CREATED or IN-DELETE if the parameter RESET=*YES has been set.

The different statuses possible for a subsystem are summarized in the following table. The changes of status caused by a DSSM command are shown on a single line. The initial status of the subsystem is indicated by a 1. The highest digit in the line denotes the final status attainable by the appropriate DSSM command. Possible intermediate statuses are also shown.

DSSM commands	**States of a subsystem**							**Operand of the DSSM command**[1]
	NOT-CREATED	IN-CREATE	CREATED	IN-DELETE	IN-HOLD	IN-RESUME	NOT-RESUMED	
START-SUBSYSTEM	1	2	3					---
			3			2	1	RESET=*YES
	1	2	3					RESET=*YES
		2	3	1				RESET=*YES
			3		1	2		RESET=*YES
			3			2	1	RESET=*YES
STOP-SUBSYSTEM	3		1	2				---
	3			2			1	---
	2				1			FORCED=*YES
	2			1				FORCED=*YES
HOLD-SUBSYSTEM			1		2		3	---
					1		2	FORCED=*YES
RESUME-SUBSYSTEM			3			2	1	---
			3			2	1	RESET=*YES
			3		1	2		RESET=*YES

[1] Operand of the DSSM command which must be set in order to change t he status

Table 7: Statuses of a subsystem

3.1.12 Subsystem monitoring with monitoring job variables

Subsystems can be monitored with monitoring job variables (MONJV). The MONJV must be specified in the START-SUBSYSTEM command, see page 153. DSSM administers and sets the MONJV during the entire subsystem runtime until it is shut down, with:

- explicit DSSM calls (/HOLD-SUBSYSTEM, /RESUME-SUBSYSTEM, /STOP-SUBSYSTEM, /UNLOCK-SUBSYSTEM)
- implicit operations (subsystem, automatic restore ...)
- SHUTDOWN

The MONJV indicates whether the subsystem is active, stopped, interrupted or locked. The MONJV can have the following contents:

Byte	Length	Contents	Values
1	3	Status	$R (running) / $A (abnormal end) / $L (loaded) / $T (terminate)
4	1	Reserved	0
5	4	TSN	???? (*four question marks*)
9	4	Home Catid	
13	4	Reserved	
17	1	Type	J / P / S
18	53	Reserved	
71	3	Session number	
74	8	Name of the subsystem	
82	7	Version of the subsystem	

89	15	Condition of the subsystem	for $R: created
			for $A: abnormal end / locked
			for $L: in create
			for $T: not created / not resumed / in delete / in resume / in hold

104	24	Unused	
128	127	Reserved for subsystem users	

Table 8: Contents of monitoring job variables

3.1.13 Overview of functions

This table provides an overview of the functions offered by DSSM/SSCM, and shows which operands may be specified for the individual subsystem classes.

Function (operands)	**TPR**	**TU**		**Sys**	**Share**	**PU**
	SAS	**SAS**	**UAS**	**exits**	**prod.**	
Subsystem declaration (SSCM) statement SET-SUBSYSTEM-ATTRIBUTES						
Identification						
SUBSYSTEM-NAME, VERSION,-DYNAMIC-CHECK-ENTRY	X	X	X	X	X	
Linking and loading						
LIBRARY, REP-FILE,LINK-ENTRY, AUTOLINK,-UNRESOLVED-EXTERNALS	X	X	X	X	X	
CREATION-TIME						
= *BEFORE-DSSM-LOAD/*AT-DSSM-LOAD/-*BEFORE-SYSTEM-READY	X			X		
= *AFTER-SYSTEM-READY/-*AT-CREATION-REQUEST	X	X	X	X	X	
= *AT-SUBSYSTEM-CALL	1					
REFERENCED-SUBSYSTEMS	2	2	2	2	2	
Address space						
MEMORY-CLASS						
= *SYSTEM-GLOBAL	X	X		X	10	
= *LOCAL-PRIVILEGED			X		10	
= *LOCAL-UNPRIVILEGED			X		10	
= *BY-SLICE		X	X		10	
SUBSYSTEM-ACCESS						
= *SYSTEM	X			X		
= *HIGH		X	X		X	
SIZE			X		X	
START-ADDRESS			3		3	
Subsystem satellites						
MESSAGE-FILE, SYNTAX-FILE	X	X	X	8	8	
SUBSYSTEM-INFO-FILE	X			8		
Starting and terminating						
INIT, STOPCOM, DEINIT, CLOSE-CTRL routine,-INTERFACE-VERSION	X			X		
Holder task for execution						
RESTART-REQUIRED	X		X			

Table 9: Overview of DSSM functions (part 1 of 2)

Function (operands)	**TPR**	**TU**		**Sys**	**Share**	**PU**
	SAS	**SAS**	**UAS**	**exits**	**prod.**	
Execution (operand SUBSYSTEM-ENTRIES=(...))						
MODE						
= *LINK	X	X	X		X	
= *SVC/*SYS-EXIT/*ISL	X			X		
CONNECTION-ACCESS						
= *SYSTEM	X			X		
= *ALL		X	X		X	
CONNECTION-SCOPE						
= *PROGRAM/*TASK/*FREE	X	X	X	X	X	
= *CALL/*OPTIMAL	9					
Subsystem configuration (SSCM) statements						
ASSIGN-HOLDER-TASK	X	4	X		8	
SET-SUBSYSTEM-ATTRIBUTES (Operand RELATED-SUBSYSTEMS)	X	X	X	X	X	
SET-SUBSYSTEM-ATTRIBUTES (Operand REFERENCED-SUBSYSTEMS)	11	11	11	11	11	
SEPARATE-ADDRESS-SPACE			5		5	
Subsystem management (DSSM) commands						
START-SUBSYSTEM, STOP-SUBSYSTEM, HOLD-SUBSYSTEM, RESUME-SUBSYSTEM	X	X	X	X	X	
ADD-/REMOVE-/UNLOCK-SUBSYSTEM, SAVE-SUBSYSTEM-CATALOG						X
MODIFY-SUBSYSTEM-PARAMETER	X	X	X	X	X	X
Subsystem information (DSSM) commands						
SHOW-SUBSYSTEM-ATTRIBUTES						X
SHOW-SUBSYSTEM-INFO						X
SHOW-SUBSYSTEM-STATUS	X	X	X	X	X	
Global subsystem management (DSSM) commands						
RELEASE-SUBSYSTEM-SPACE			6			
SET-DSSM-OPTIONS	7	7	7	7	7	7

Table 9: Overview of DSSM functions (part 2 of 2)

Key

TPR SAR	Privileged subsystems (only system address space)
TU SAS	Nonprivileged subsystems with system address space
TU UAS	Nonprivileged subsystems with user address space
Sys exits	Relevant for system exits
Share prod.	Relevant for share products
PU	Privileged user ID for systems support (user ID with SUBSYSTEM-MANAGEMENT privilege)

X	Function available
1	Only for subsystems with SVC and/or ISL connection
2	Relations only from UAS to SAS
3	only for subsystems in class 6 memory
4	Holder task is required in system address space only for execution purposes
5	Required only for subsystems in the address space strip
6	Reservation for the nonprivileged address space strip or both strips can be canceled
7	Only for diagnosis and debugging
8	Only where appropriate
9	Only in conjunction with MODE=*SVC or MODE=*ISL
10	Depending on storage location in the address space
11	Should be reserved for subsystems in the same “family” (privileges, memory class, etc.)

3.2 Storage and task concepts

Storage concept

Subsystems are loaded as follows, depending on the declaration:

- into the system address space (class 3 or class 4 memory) if MEMORY-CLASS=*SYSTEM-GLOBAL
- into the user address space (memory pool: class 5 or class 6 memory) if MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED
- into both address spaces if MEMORY-CLASS=*BY-SLICE

Interfaces to subsystems in the user address space are possible only if the allocated address area in the local task is free. The code must always be located at the same address after linkage. For generally accessible subsystems in the user address space, which can be addressed by any task at any time, a fixed area in class 5 memory is reserved, known as the “address space strip”. This address space strip is shared by all generally available subsystems, with a separate strip being available for each nonprivileged subsystem.

Problems can occur for user tasks if there is not enough class 5 memory space. The address space strip reserved for subsystems can be released by means of the DSSM command RELEASE-SUBSYSTEM-SPACE in order to obtain more class 5 memory space.

Subsystems that are not used at the same time and therefore do not call each other may be located in parallel within the address space. The more parallel subsystems there are, the smaller is the address space strip required. Care should be taken to ensure the right balance of parallel subsystems, since a high degree of parallelism, though it saves address space, may result in a deterioration of performance (the more parallelism, the more mutual preemption). The distribution of the subsystems within the address space strip can be controlled using the SSCM statement SEPARATE-ADDRESS-SPACE.

Address space housekeeping

DSSM address space housekeeping provides a means of reducing the load on the system address space (class 3 and class 4 memory) by putting subsystems in the address space of the holder task. This is only of relevance to nonprivileged subsystems, however, because all privileged subsystems are always loaded into the system address space (MEMORY-CLASS=*SYSTEM-GLOBAL).
Transfer is straightforward, provided that the subsystems do not call each other, do not depend on each other through a third program, and do not have the ability to run in class 6 memory as a main program.
Note that it is only possible to attempt to relocate nonprivileged subsystems which are currently below the 16-Mbyte boundary.
For all other nonprivileged subsystems, loading above the 16-Mbyte boundary is recommended (SUBSYSTEM-ACCESS=*HIGH).

Subsystems which are reentrant-compatible and run as main programs can also be swapped out of the system address space. They must be available in the memory pool in the class 6 memory (MEMORY-CLASS=*LOCAL-UNPRIVILEGED).

There is a strategy aimed at minimizing system address space occupancy for subsystems that consist of a shareable code (program area) and a non-shareable code (data area). This strategy works as follows:
The program area is loaded into the shareable address space (this corresponds to MEMORY-CLASS=*SYSTEM-GLOBAL). The data area is loaded into the user address space of the holder task; then, when a task is linked to the subsystem in the task’s private user address space, it is copied to the same address.
This strategy is implemented when a subsystem is defined with MEMORY-CLASS=*BY-SLICE.

The advantages of this approach are as follows:

– address-related references between the program area and data area are always possible because the copy of the data area begins at the same address as the original

– performance is considerably enhanced by this type of address space apportionment because no access is made to the program library or object module library when a task is linked to the subsystem, and external references do not need to be resolved.

The disadvantages of this approach are:

– once the address space for the data area has been defined and reserved at the time of startup, it can be expanded to only a very limited degree; optimization of memory space is not possible because of the strict apportionment of address space

– if the address space belonging to the task being linked and intended for the data area is already occupied, the subsystem code (data area and program area) is loaded in its entirety into the user address space of that task.

Relocation from the system address space to the user address space is only worthwhile in the following cases:

a) Where subsystems which are independent of each other and are not used simultaneously by user programs can be placed in parallel in the user address area. If this is not the case, relocation is pointless, since they will take up more memory in the user address space than in the system address space.

b) Where the subsystems are large enough to compensate for the loss that occurs through the use of the memory pool (minimum size of a memory pool: 1Mbyte).

c) Where the subsystem must be located below the 16-Mbyte boundary (for subsystems which may also be loaded above this boundary, the operand SUBSYSTEM-ACCESS=*HIGH can be used to counteract any overloading of the lower 16 Mbytes).

Summary:

It is only appropriate to use parallel configuration of independent subsystems in memory pools if the sum of the sizes of all the subsystems amounts to over 1Mbyte and none of the subsystems is larger than 1Mbyte in size.

Task concept - holder task

A subsystem is activated under its own task, the holder task. Depending on the type of subsystem, this task can be used as a subsystem work task or as a pure holder task. The user address space of this task can be used for relocation from the system address space.

The number of holder tasks required should be kept as small as possible. A large number of tasks does have a positive effect on parallelism, since the more tasks are created at the same time, the more subsystems can be installed. On the other hand, more tasks also require more task-specific tables.

DSSM itself minimizes the number of holder tasks. However, the distribution of subsystems can also be controlled with SSCM by means of the ASSIGN-HOLDER-TASK statement (see page 193) during generation of the subsystem catalog.
The default option is that all subsystems defined with MEMORY-CLASS=*BY-SLICE are connected to the same holder task.

If an error occurs, a restart of the holder task is automatically initiated, to avoid bringing down all the subsystems which are connected to the holder task. Provision is also made for restarting a subsystem by means of the RESTART-REQUIRED operand of the SET-SUBSYSTEM-ATTRIBUTES command. This parameter makes it possible to call the subsystem initialization again if the holder task was terminated during the execution of subsystem routines (see page 72).

3.3 Management of shared programs

DSSM supports the management of shared programs. This permits the following during a session:

– unloading of shared programs, even from class 4 memory
– relocation of shared programs from class 4 memory to class 5 or class 6 memory.

Shared programs can be declared as subsystems and as such administered by DSSM. They can be dynamically activated, deactivated, suspended and resumed, just like any other subsystems.

This requires that the shared programs be declared during generation of the subsystem catalog. Given this definition, a shared program can be activated and deactivated like a normal subsystem after DSSM has been started.

Users can access the program via the BIND macro or by means of the START-PROGRAM and LOAD-PROGRAM commands.

The entry point must always be defined by means of the SSCM statement SET-SUBSYSTEM-ATTRIBUTES,...SUBSYSTEM-ENTRIES=, even if it is identical with the LINK-ENTRY (of the reference address used to load the subsystem).

3.4 Management of the dynamic subsystem catalog

Adding to the dynamic subsystem catalog

System administration can extend the current dynamic subsystem configuration during a session (ADD-SUBSYSTEM command). The catalog for the new subsystem configuration must be generated using SSCM, and the following points must be noted:

- If required, the new catalog can be larger that the old one (i.e. it may also include all the subsystems from the old catalog), or it may be created as a “delta” catalog, containing only the new subsystem definitions.
- The “old” subsystem catalog, which was used during startup, will not automatically be updated. At the next startup it is possible
  - to use the “new” catalog, or
  - to use instead a catalog which has been completely recreated and which, for example, does not contain the versions of subsystems which are no longer being used, and in which the required changes have been made to attributes (such as CREATION-TIME).
- ASSIGN-HOLDER-TASK statement

  This statement must not be specified for old and new subsystems.

  Example

Subsystems in the old catalog:	A, B, C
Subsystems in the new catalog:	A, B, C, D and E

  Then the following is allowed:
  ```
  //ASSIGN-HOLDER-TASK TYPE=SHARED-HOLDER(BY-SUBSYSTEMS=(A,B))
  //ASSIGN-HOLDER-TASK TYPE=SHARED-HOLDER(BY-SUBSYSTEMS=(D,E))
  ```
  and the following is *not* allowed:
  ```
  //ASSIGN-HOLDER-TASK TYPE=SHARED-HOLDER(BY-SUBSYSTEMS=(A,B,E))
  ```

- SET-SUBSYSTEM-ATTRIBUTES statement

  The CREATION-TIME operand for a new subsystem must be compatible with any versions of the subsystem which already exist in the old subsystem configuration.

  Example

  If version 1 of subsystem X in the old catalog is declared with CREATION-TIME=*AT-SUBSYSTEM-CALL, then version 2 of subsystem X in the new catalog must have CREATION-TIME=*AT-CREATION-REQUEST.

  If CREATION-TIME=*BEFORE-SYSTEM-READY or *AFTER-SYSTEM-READY has been specified for a new subsystem, the subsystem will not be created because the SYSTEM READY point in time (i.e. system initialization) will not occur. A corresponding message will be issued.

  New subsystems which are being added and have the attribute MEMORY-CLASS= *LOCAL-PRIVILEGED (class 5 memory) must not alter the width of the system or user address space strip, nor the location of the old subsystems within this strip.

  No relations may be declared from the old to the new subsystems (REFERENCED-SUBSYSTEM and RELATED-SUBSYSTEM operands).

  Example

Subsystems in the old catalog:	A, B, C
Subsystems in the new catalog:	A, B, C, D and E

  X ➝ Y means that X requires Y in order to resolve external calls.

Then:	A ➝ D	not allowed
	D ➝ E	is allowed
	D ➝ B,C	is allowed.

- SEPARATE-ADDRESS-SPACE statement

  New and old subsystems may be disjunctive, i.e. their addresses may overlap.

  Example

Subsystems in the old catalog:	A, B, C
Subsystems in the new catalog:	A, B, C, D and E

  X ➝ Y means that X is disjunctive to Y.

Then:	A ➝ D	is allowed
	D ➝ E	is allowed
	D ➝ B,C	is allowed.

Changing subsystem attributes during a session

The MODIFY-SUBSYSTEM-PARAMETER command can be used to change subsystem attributes during a session, without the user having to shut down the subsystem configuration or add a new version of the subsystem. Changes in the configuration can be saved for the next startup by means of the SAVE-SUBSYSTEM-CATALOG command.

These possibilities are helpful especially in the following situations:

- In order to stop a subsystem, all the connected tasks must have cleared their links. If the subsystem was not defined with the attribute FORCED-STATE-CHANGE=*ALLOWED, there is no way of accelerating the deactivation of the subsystem. Should certain considerations make it necessary to do so, however, it is now possible to change the attribute dynamically by means of the MODIFY-SUBSYSTEM-PARAMETER command.
- If memory pool contention occurs because, for example, two subsystems are located at the same address and the task can use only one of the two, the addresses of the subsystems can be changed to avoid memory pool contention by means of the MODIFY-SUBSYSTEM-PARAMETER command.

3.5 Startup of dynamic subsystem management

Dynamic subsystem management is started during BS2000 system initialization.

All the information necessary for DSSM initialization is entered via the parameter service. This information includes the name of the static subsystem catalog and the DSSM version number. If absolutely necessary, logging of DSSM-specific data for error diagnosis may be activated at this point.

All the records processed by means of the parameter service are listed in the form of messages in the CONSLOG logging file.

The keyword for starting up subsystem management is **DSSM**.

The procedure for starting up DSSM via the parameter service is described in detail in the manual "Introductory Guide to Systems Support" [14] .

In order for DSSM V4.0 to operate, the following files must be stored on the home pubset under the TSOS user ID:

SYSLNK.DSSM.040	Library with load modules for DSSM
SYSNRF.DSSM.040	Reference file for DSSM REP file processing (NOREF file)
SYSREP.DSSM.040	REP correction file for DSSM
SYSSDF.DSSM.040	SDF syntax file
SYSMES.DSSM.040	Message file

The subsystem catalog which is to be created must likewise be placed on the home pubset and stored under the TSOS user ID. The catalog may be named as desired, and the name can be made known to the system via the parameter service.

The SSCM program is available for generating a subsystem catalog. This program is described in detail as of page 179.

Problem handling during a system run

If DSSM cannot be initialized, the reason is displayed in a message (e.g. missing static subsystem catalog) and message ESM0401 is output. The operator can specify a new subsystem catalog during the system run on the operator terminal if it was not defined in the parameter file or the standard catalog (SYS.SSD.CAT.X) is not present.

The system run is generally not continued if mandatory subsystems cannot be started up. The reason for the error during DSSM initialization must first be eliminated and the system run must then be repeated.

If subsystems with the *BEFORE-SYSTEM-READY attribute cannot be started up, DSSM continues the system run.

If one of the following files of a subsystem with the *AT-DSSM-LOAD or *MANDATORY-AT-STARTUP attribute is missing, the operator can specify a new, valid name during the system run:

– information file
– REP correction file
– module library

The system run is stopped if the operator does not input a new file name for the missing REP correction file or the module library.

If the information file is missing, the operator can

– input a new, valid file name
– ignore the message and continue the system run
– ignore the message and stop the system run.

Format of the parameter record for starting up DSSM

Format	Meaning
SSMCAT=name	Name of the static subsystem catalog
VERSION=versno	Version number of DSSM
LOGGING=ON / OFF	Controls DSSM-specific logging for error diagnostics
REPFILE=<repfile name>	Name of the REP correction file for loading DSSM

START-SUBSYSTEM commands must be issued in the BS2000 session for any subsystems that are not set up automatically during system initialization.

Excerpt from the parameter file

```
/BS2000 PARAMS
:
/BEGIN DSSM
 SSMCAT=name  ———————————————————————————————————————————————————————————  (1)
 VERSION=versno  ————————————————————————————————————————————————————————  (2)
 LOGGING=ON / OFF  ——————————————————————————————————————————————————————  (3)
 REPFILE=repfile name ———————————————————————————————————————————————————  (4)
/EOF
:
/END-PARAMS
```

(1) Control and parameter records must exist in the parameter file only if the system support wishes to set values different from the following default values:

for BS2000/OSD-BC V5.0:	SSMCAT=$TSOS.SYS.SSD.CAT.X and VERSION=040

for BS2000/OSD-BC V4.0:	SSMCAT=$TSOS.SYS.SSD.CAT.X and VERSION=040
	or
	SSMCAT=$TSOS.SYS.SSD.CAT.X and VERSION=039
	or
	SSMCAT=$TSOS.SYS.SSD.CAT.X and VERSION=038

for BS2000/OSD-BC V3.0:	SSMCAT=$TSOS.SYS.SSD.CAT.X and VERSION=040
	or
	SSMCAT=$TSOS.SYS.SSD.CAT.X and VERSION=036

(2) The version number refers to all DSSM-specific file names (e.g. SYSLNK.DSSM.040, SYSREP.DSSM.040 in the case of BS2000/OSD-BC V5.0).

(3) The statement LOGGING=OFF deactivates logging. With LOGGING=ON a log containing diagnostics data is created during DSSM startup.
Logging is not possible if only default values are specified.

(4) Name of the desired REP correction file for loading DSSM.
If the parameter is not specified, DSSM is loaded with the default name of the REP correction file (SYSREP.DSSM.*version*).

3.6 DSSM accounting records

General comments on the BS2000/OSD accounting system

The BS2000/OSD accounting system as a whole is controlled by systems support. Systems support determines the time at which the accounting system is to be started, declares the name of the accounting file, and defines the name and number of accounting records and record extensions that are to be recorded in the accounting file.
Systems support also determines the cycle and scope of periodic data entry encompassing certain accounting records and job classes.

The accounting system can be used by systems support to activate and deactivate accounting records, either entirely or in part, during a session, and to influence the extent of the individual accounting records.
Accounting records and record extensions which are not required can be deactivated by means of the MODIFY-ACCOUNTING-PARAMETERS command.

The attributes used in the representation of each data field are as follows:

Field	Consecutive number of the data field within the described part of the record		
Displace-ment	Relative distance of the data field from the start of the described part of the record		
Length	Length of the data field in bytes		
Format	A	=	alphanumeric
	B	=	binary number
	B2	=	binary representation of CPU time
	C	=	printable characters, including special characters
	F	=	file name
	X	=	non-printable characters
	Z	=	unpacked decimal number (*)
	*	=	specified for the individual record types or extension elements
	-	=	reserved for future extensions; contains either a space or binary zero
(*)	Time is shown in the form `hhmmss`, the date in the form `yymmdd`		

An accounting record consists of the following four parts:

(A) Record description	:	record identifier, time stamp, ...
(B) Identification section	:	user ID, account number, monitored resources , ...
(C) Basic information	:	default data
(D) Variable information	:	record extensions

For more details about the BS2000/OSD accounting system refer to the manual “Introductory Guide to Systems Support” [14] or “Computer Center Ready Reference, Volume 1” [41].

Subsystem record description

The record description (A) contains a record identifier, which serves the purpose of differentiating the individual accounting records, a time stamp, and details of the length of the identification section and of the basic information.

Address of the record description = left-hand end of record

Structure and contents:

Field no.	Displacement		Length	Format	Meaning
	hex.	dec.	(bytes)		
1	00	0	4	A	Record identifier [1]
2	04	4	8	B	Time stamp of time of day clock [2]
3	0C	12	2	B	Length of identification section
4	0E	14	2	B	Length of basic information
5	10	16	4	-	- reserved -

[1] The record identifier serves to differentiate between the individual record types.

[2] The time stamp is entered in the record by the system as the last item of information after creation of the record is completed or after it has been transferred to the accounting system.

Length of the record description: 20 bytes

Subsystem identification

The subsystem identification in the identification section (B) describes the subsystem to which the subsystem account data relates.

Structure and contents:

Field no.	Displacement		Length	Format	Meaning
	hex.	dec.	(bytes)		
1	00	0	8	A	Name of the subsystem
2	08	8	7	A	Version of the subsystem
3	0F	15	8	Z	Date of the call [1]
4	17	23	6	Z	Time of the call [2]

[1] Date in the form `yyyymmdd`

[2] Time in the form `hhmmss`

Length of the DSSM identification section: 29 bytes

ESMC - subsystem initialization accounting record

The accounting record is written every time that the initialization phase of a subsystem is executed.
This activation routine traverses the subsystem under the control of DSSM on execution of the START-SUBSYSTEM and RESUME-SUBSYSTEM commands.

Maximum length of the subsystem initialization accounting record: 54 bytes

(A) Record description: record identifier: “ESMC”

(B) Identification section: subsystem identifier

(C) Basic information

Field no.	Displacement		Length	Format	Meaning
	hex.	dec.	(bytes)		
1	00	0	1	B	Status indicator [1]
2	01	1	1	A	Season identifier (current) [2]
3	02	2	1	-	- reserved -

[1] There are two possible values for the status indicator:
1 subsystem is restarted after the wait state (RESUME-SUBSYSTEM command)
0 subsystem is started (START-SUBSYSTEM command)

[2] “S” for summer time; “W” for winter time

Length of the basic information: 3 bytes

(D) Variable information

The variable information of the subsystem initialization accounting record contains **no** record extension.

ESMD - subsystem termination accounting record

The accounting record is written every time that the deinitialization phase of a subsystem is executed.
This termination routine traverses the subsystem under the control of DSSM on execution of the STOP-SUBSYSTEM and HOLD-SUBSYSTEM commands.

Maximum length of the subsystem termination accounting record: 54 bytes

(A) Record description: record identifier: “ESMD”

(B) Identification section: subsystem identification

(C) Basic information

Field no.	Displacement		Length	Format	Meaning
	hex.	dec.	(bytes)		
1	00	0	1	B	Status indicator [1]
2	01	1	1	A	Season identifier (current) [2]
3	02	2	1	-	- reserved -

[1] There are two possible values for the status indicator:
1 subsystem is placed in the wait state (HOLD-SUBSYSTEM command)
0 subsystem is terminated (STOP-SUBSYSTEM command)

[2] “S” for summer time; “W” for winter time

Length of the basic information: 3 bytes

(D) Variable information

The variable information of the subsystem termination accounting record contains **no** record extension.

3.7 Error handling in DSSM

DSSM task error

1. DSSM terminates abnormally during the first step of startup if, for example, subsystems are activated that were defined with the attribute *BEFORE-DSSM-LOAD or *AT-DSSM-LOAD. An error code is sent to the startup task.
   Normally, startup is aborted. However, under certain circumstances it may be continued without DSSM initialization (depending on how startup is implemented).

   For information on the startup steps see page 211 or the manual “Introductory Guide to Systems Support” [14].

2. DSSM terminates abnormally during the second step of startup if, for example, subsystems are activated that were defined with the attribute *MANDATORY-AT-STARTUP or *BEFORE-SYSTEM-READY or while the data structures of a subsystem with the attribute *AFTER-SYSTEM-READY are being updated. If DSSM attempted to activate one of these subsystems, its status is changed to LOCKED.

   A return code is sent to the startup task when a subsystem with *MANDATORY-AT-STARTUP is put in the LOCKED status. Whether or not startup is aborted depends on how startup is implemented.

3. DSSM terminates abnormally during shutdown if the subsystem which is to be deactivated is in the LOCKED status.
   The shutdown of other subsystems continues normally.

4. Other causes that lead to abnormal DSSM termination:
   DSSM analyzes the situation, restores the integrity of its internal tables and makes the following decisions, depending on where the error occurred:

The error occurs during ...	Reaction
CREATE/DELETE/RESUME/HOLD	Subsystem LOCKED (with error message)
the swapping of subsystem versions – with interrupted availability without interrupted availability	The faulty routine is called again; if it again results in the error, – the version concerned or – both subsystem versions are placed in the LOCKED status.
ADD-/MODIFY-SUBSYSTEM-PARAMETER, REMOVE-SUBSYSTEM	The statement is aborted; parts of the catalog may already have been changed.
restoration of the holder task	Restoration is aborted.
SHOW-SUBSYSTEM-INFO, SAVE-CATALOG	The request is terminated abnormally.

   After restoring the integrity of the tables, DSSM resumes its work with the requests waiting in the DSSM bourse.

Holder task error

If problems occur in the holder task or it is terminated abnormally, DSSM analyzes the situation, automatically initiates the restart of the holder task and makes the following decisions:

The problem occurs ...	**Reaction to the holder task error when**	
	RESTART-REQUIRED=*YES	**RESTART-REQUIRED=*NO**
– when normal demands are being made on the subsystem		
Activation	Subsystem LOCKED	Subsystem LOCKED
INIT routine	INIT routine called	Subsystem LOCKED
CLOSE-CTRL routine	Subsystem CREATED	Subsystem LOCKED
STOPCOM routine	INIT routine called	Subsystem LOCKED
DEINIT routine	Deinitialization continued	Subsystem LOCKED
Deactivation	Subsystem LOCKED	Subsystem LOCKED
Subsystem session (work task)	INIT routine called	Subsystem LOCKED
– when swapping subsystem versions and interrupting availability		
Activation of V2	Subsystem V2 LOCKED and swap aborted	Subsystem V2 LOCKED and swap aborted
STOPCOM routine of V1	Subsystem V1 LOCKED and INIT routine for V2 called	Subsystem V1 LOCKED and INIT routine for V2 called
INIT routine of V2	INIT routine of V2 called and DEINIT routine of V1 continued	Subsystem V2 LOCKED and DEINIT routine of V1 continued
Deactivation of V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
DEINIT routine of V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
– when swapping subsystem versions without interrupting availability		
Activation of V2	Subsystem V2 LOCKED and swap aborted	Subsystem V2 LOCKED and swap aborted
CLOSE-CTRL routine of V1	Swap aborted (subsystem V1 CREATED)	Subsystem V1 LOCKED and unloading of V1
INIT routine of V2	INIT routine of V2 called and DEINIT routine of V1 continued	Subsystem V2 LOCKED and swap aborted
STOPCOM routine of V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
DEINIT routine of V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
Deactivation of V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED

Error log SERSLOG

DSSM writes an entry in the SERSLOG error log whenever

- a system call is faulty,
- inconsistent data is detected in the internal subsystem catalog or
- the DSSM task terminates abnormally (in this case, the SERSLOG entry contains a message describing the current situation).

The entry may consist of up to three parts:

1. the parameter list of the faulty system call or the inconsistent data (in 2K units),
2. the return code (if it is not already included in the parameter list) and
3. the address of the faulty routine and the address of the routine that called the faulty routine. This entry is useful for diagnostic purposes because DSSM uses its own central SERSLOG call.

3.8 DSSM commands

Command	Meaning	Page
ADD-SUBSYSTEM	Extend dynamic subsystem catalog	75
HOLD-SUBSYSTEM	Place subsystem in wait state	80
LOAD-LOCAL-SUBSYSTEM-CATALOG	Load local subsystem catalog	83
MODIFY-SUBSYSTEM-PARAMETER	Modify subsystem parameters	85
RELEASE-SUBSYSTEM-SPACE	Release reserved address space for subsystems	110
REMOVE-SUBSYSTEM	Remove inactive subsystem from dynamic catalog	111
RESUME-SUBSYSTEM	Cancel wait state for subsystem	113
SAVE-SUBSYSTEM-CATALOG	Save changes to dynamic subsystem catalog	116
SET-DSSM-OPTIONS	Activate/deactivate DSSM logging function	120
SHOW-SUBSYSTEM-ATTRIBUTES	Request information on subsystem attributes	122
SHOW-SUBSYSTEM-INFO	Request information on current subsystem configuration	138
SHOW-SUBSYSTEM-STATUS	Request information on status of subsystems	142
START-LOCAL-SUBSYSTEM	Activate local subsystem in user address space	150
START-SUBSYSTEM	Activate subsystem	153
STOP-LOCAL-SUBSYSTEM	Deactivate local subsystem in user address space	159
STOP-SUBSYSTEM	Deactivate subsystem	162
UNLOAD-LOCAL-SUBSYSTEM-CATALOG	Unload local subsystem catalog	166
UNLOCK-SUBSYSTEM	Shift subsystem from LOCKED status to NOT-CREATED status	168

Table 10: DSSM commands

The **SDF syntax representation** of the commands is explained in section “SDF syntax representation” on page 5.

ADD-SUBSYSTEM
Extend dynamic subsystem catalog

Domain:	SYSTEM-MANAGEMENT
Privileges:	SUBSYSTEM-MANAGEMENT

Function

Using this command, system administration can extend the current subsystem configuration during a session. The catalog specified may either be a completely new one, which includes all the entries in the previous one, or it may contain only the new subsystems which are to be added to the current catalog.

In either case, the subsystem catalog specified must have been generated using SSCM.

The (“old”) subsystem catalog used during system initialization is not automatically updated. For the **next** session, system administration can either

- use the catalog generated by means of ADD-SUBSYSTEM during system initialization, or
- generate a completely new and updated subsystem catalog and use this for system initialization. This new catalog need not build up quantitatively on an old predecessor catalog nor qualitatively support its references and attributes.

Format

ADD-SUBSYSTEM
CATALOG = <filename 1..54 without-gen-vers>
,**TYPE** = ***EXT**ENDED**-ACT**IVE**-CONF**IGURATION / ***NEW-SUBSYS**TEMS

Operands

CATALOG = <filename 1..54 without-gen-vers>
Name of the new subsystem catalog.

TYPE =
Specifies whether the current catalog is to be extended or replaced.

TYPE = *EXTENDED-ACTIVE-CONFIGURATION
A completely new catalog is to be activated, containing not only the entries from its predecessor, but also the new subsystems (refer also to the notes on page 77).

TYPE = *NEW-SUBSYSTEMS
The specified catalog contains only new subsystems, which are to be added to the old catalog.
DSSM will check the catalog to ensure that the subsystems which it contains really are new. If any subsystem is found which is also listed in the catalog which is being extended, the command will be rejected.

Restrictions:

- The subsystems specified for generation using SSCM with RELATED-SUBSYSTEM and REFERENCED-SUBSYSTEM must be cycle-free, i.e. free of mutual dependency.
- It is not permitted to define different versions of a subsystem with the start attributes AT-SUBSYSTEM-CALL, BEFORE-SYSTEM-READY, AFTER-SYSTEM-READY, BEFORE-DSSM-LOAD, AT-DSSM-LOAD and MANDATORY-AT-STARTUP
(exception: AT-SUBSYSTEM-CALL is permitted if coexistence is defined for all versions involved).

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	32	ESM0296	Abnormal termination (REQM error)
	32	ESM0350	Internal DSSM problem during processing
	64	ESM0260	File not found
	64	ESM0261	Storage space limit reached in DSSM catalog
	64	ESM0262	File is not a catalog
	64	ESM0322	Maximum number of relations exceeded
	64	ESM0325	Inconsistencies in catalog
	64	ESM0332	Incompatible version of catalog
	64	ESM0340	Reserved address-space exhausted for subsystems with MEMORY-CLASS=*BY-SLICE

Notes

There may be relationships between subsystems which are not defined in the current catalog (see the operand FORCED=*FOR-ADD-SUBSYSTEM of the SSCM statement SAVE-CATALOG). You can get round this problem by defining these subsystems in a new catalog and adding them to the old catalog by means of the /ADD-SUBSYSTEM.
Example for TYPE=*EXTENDED-ACTIVE-CONFIGURATION:

Old catalog

```
//START-CATALOG-CREATION old-cat
//SET-SUBSYSTEM-ATTRIBUTES   -
//  SUBSYSTEM-NAME=ss1,      -
//  RELATED-SUBSYSTEM=ss2
//SAVE-CATALOG FORCED=       -
//  *FOR-ADD-SUBSYSTEM
```

New catalog

```
//START-CATALOG-CREATION new-cat
//SET-SUBSYSTEM-ATTRIBUTES   -
//  SUBSYSTEM-NAME=ss1,      -
//  RELATED-SUBSYSTEM=ss2
//SET-SUBSYSTEM-ATTRIBUTES   -
//  SUBSYSTEM-NAME=ss2
//SAVE-CATALOG
```

Restrictions concerning the operand TYPE=*EXTENDED-ACTIVE-CONFIGURATION

- Subsystems with the attribute MEMORY-CLASS=*LOCAL-PRIVILEGED, which are being added to the newly created catalog, must not exceed the size of the address space strip in user or system address space, nor may their location in the address space overlap with subsystems from the old catalog.
- The CREATION-TIME operand for any new subsystem must be chosen so as to be compatible with versions of the same subsystem which are already defined in the old catalog. In making this choice, the values BEFORE-SYSTEM-READY, AFTER-SYSTEM-READY, BEFORE-DSSM-LOAD, AT-DSSM-LOAD and MANDATORY-AT-STARTUP could be used, but would have no effect since the system startup time for a session which has started will already have been passed; i.e. system administration will be given an appropriate warning, but the subsystem will not be loaded.
- When distributing subsystems to holder tasks (ASSIGN-HOLDER-TASK statement), the “stand alone” principle must be observed, i.e. subsystems from different catalogs must not be assigned to the same holder task.

Example

Subsystems in the old catalog:	A, B, C
Subsystems in the new catalog:	A, B, C, D, E

Then:

`//ASSIGN-HOLDER-TASK *SHARE-HOLDER(BY-SUB=(A,B))`	is permissible
`//ASSIGN-HOLDER-TASK *SHARE-HOLDER(BY-SUB=(D,E))`	is permissible

but:

`//ASSIGN-HOLDER-TASK *SHARE-HOLDER(BY-SUB=(A,D,C))`	is not permissible

- The new catalog must be larger than its predecessor, because it not only contains the old subsystems with their attributes (relations, dependencies, loading instructions), but must also maintain details of the new subsystems.
- Link and dependency relationships (REFERENCED-SUBSYSTEM/RELATED-SUBSYSTEM) must not violate the catalog boundaries. The catalog A must not contain any relationships of a subsystem defined in this catalog to a subsystem defined in the catalog B.
- Once REMOVE-SUBSYSTEM has been used to delete a subsystem from the catalog, TYPE=*EXTENDED-ACTIVE-CONFIGURATION can no longer be specified.

It is not permissible to define different

- subsystems with an identical combination of the attributes: SVC-NUMBER / FUNCTION-NUMBER / FUNCTION-VERSION.
- subsystems with an identical combination of the attributes: FUNCTION-NUMBER / FUNCTION-VERSION (if the value *ALLOWED is set for VERSION-COEXISTENCE or VERSION-EXCHANGE) for subsystems which are indirectly linked via System Procedure Linkage (ISL).
- versions of a subsystem with an identical combination of the attributes: SVC-NUMBER / FUNCTION-NUMBER / FUNCTION-VERSION if the value *ALLOWED is set for VERSION-COEXISTENCE or VERSION-EXCHANGE.
- versions of a subsystem with an identical combination of the attributes: FUNCTION-NUMBER / FUNCTION-VERSION / VERSION-COEXISTENCE or VERSION-EXCHANGE for subsystems which are indirectly linked via System Procedure Linkage (ISL).

Subsystem overlaps must be avoided. To this end, DSSM compares the values of the SIZE and START-ADDRESS operands in the SET-SUBSYSTEM-ATTRIBUTES statement.

Assigning a holder task (via SSCM statement) for an old and a new subsystem has no effect on the holder task allocation.

Table of incompatibilities for relations between subsystems in the old and new catalog:

x: this combination is not possible:
neither link relations (REFERENCED-SUBSYSTEM) nor any other dependencies (RELATED-SUBSYSTEM) are permitted

r: link relations (REFERENCED-SUBSYSTEM) are not permitted

Subsystem in the new catalog	**Subsystem in the old catalog**												
	MANADATORY-AT-STARTUP	BEFORE-SYSTEM-READY	AFTER-SYSTEM-READY	AT-CREATION-REQUEST	AT-SUBSYSTEM-CALL	BEFORE-DSSM-LOAD	AT-DSSM-LOAD	STOP-AT-SHUTDOWN=*YES	SUBSYSTEM-ACCESS=*LOW	SUBSYSTEM-ACCESS=*HIGH	MEMORY-CLASS=*LOCAL-PRIVILEGED	MEMORY-CLASS=*LOCAL-UNPRIVILEGED	MEMORY-CLASS=*BY-SLICE
MANDATORY-AT-STARTUP		x	x	x	x								
BEFORE-SYSTEM-READY			x	x	x								
AFTER-SYSTEM-READY				x	x								
BEFORE-DSSM-LOAD	x	x	x	x	x		x						
AT-DSSM-LOAD	x	x	x	x	x								
STOP-AT-SHUTDOWN=*NO								r					
SUBSYSTEM-ACCESS=*SYSTEM									x	x			x
MEMORY-CLASS=*SYSTEM-GLOBAL											x	x	
MEMORY-CLASS=*BY-SLICE											r	r	r

Table 11: Incompatibilities for relations between subsystems in the old and new catalog

HOLD-SUBSYSTEM Place subsystem in wait state

Domain:	SYSTEM-MANAGEMENT
Privileges:	OPERATING SUBSYSTEM-MANAGEMENT

Function

The HOLD-SUBSYSTEM command enables any given subsystem to be placed in the wait state.
No new connection is permitted to the specified subsystem; the required resources (holder task, address space) remain available. In addition, the FORCED operand can cause the command to wait until the occupying tasks have finished executing or terminate them forcibly without waiting. On completion of the deinitialization phase, the subsystem is in the wait state; this can be canceled using the RESUME-SUBSYSTEM command.

The HOLD-SUBSYSTEM command is rejected if a subsystem was defined with SUBSYSTEM-HOLD=*FORBIDDEN.

i In order to ensure a high degree of parallelism and data integrity, time-consuming administration tasks are not performed under control of the calling task; instead they are transferred to a DSSM task.
As a rule, only the check of the requested function is carried out **synchronously** (i.e. contingent upon a wait state for the calling task). DSSM carries out the actual processing **asynchronously** and independent of the calling task.

Format

HOLD-SUBSYSTEM
SUBSYSTEM-NAME = <structured-name 1..8>
,**VERSION** = ***STD** / <product-version mandatory-man-corr> / <product-version without-man-corr> / ***HIGHEST**
,**SUBSYS**TEM-**PAR**AMETER = ***NONE** / <c-string 1..254>
,**FORCED** = ***NO** / ***Y**ES
,**SYNCH**RONOUS = ***NO** / ***Y**ES

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem to be placed in the wait state.

VERSION = *STD / <product-version mandatory-man-corr> / <product-version without-man-corr> / *HIGHEST
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type "product-version" on page 13).

VERSION = *STD
If there is only **one** version of the subsystem which has been loaded, this version is selected. If there are **several** suitable versions, the required version must be specified.

VERSION = *HIGHEST
Selects the highest version of the subsystem.

SUBSYSTEM-PARAMETER = *NONE / <c-string 1..254>
Specifies whether special parameters are to be processed, which can be evaluated only by the specified subsystem.

FORCED =
Determines the mode and urgency of command processing.

FORCED = *NO
Processing and hence normal termination of all the tasks accessing this subsystem is allowed to take its normal course.

FORCED = *YES
All accessing tasks are terminated immediately. This can lead to a system dump in the case of a privileged subsystem; the program of tasks that are connected to a nonprivileged subsystem is terminated and an error handling routine of the event class ABEND is executed.

SYNCHRONOUS =
Enables synchronous or asynchronous processing to be selected.

SYNCHRONOUS = *NO
The command is to be processed asynchronously, i.e. there is no need to wait for it to execute before making another entry. After the syntax of the command has been checked the calling task receives the message `ESM0216`. Error messages relating to execution of the command are not output at the console.

SYNCHRONOUS = *YES
The command must first be executed before another entry can be made.
Accompanying error messages are output to the task.
In the event of a version swap this entry is relevant only to the new version that is to be activated. Deactivation of the other, “old” version is always executed asynchronously.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
1	0	CMD0001	No action necessary; subsystem already in wait state
	1	ESM0414	Syntax error: an invalid version was specified
	32	ESM0224	Command is not processed
	32	ESM0228	Command terminated abnormally

LOAD-LOCAL-SUBSYSTEM-CATALOG
Load local subsystem catalog

Domain:	SYSTEM-MANAGEMENT
Privileges:	STD-PROCESSING

Function

The LOAD-LOCAL-SUBSYSTEM-CATALOG command can be used by the caller to load a local subsystem catalog into the task-specific user address space (class 5 memory).

A local subsystem catalog is not permitted to contain any privileged subsystems.

Only one local subsystem catalog can be loaded at any one time. An attempt to load another local subsystem catalog without previously unloading the first catalog with the UNLOAD-LOCAL-SUBSYSTEM-CATALOG command will be rejected with an error message. The LOAD-LOCAL-SUBSYSTEM-CATALOG command is only executed if the specified catalog was defined as a local catalog, i.e. if it does not contain any privileged subsystems and was generated with SSCM V2.0 or higher.

A local subsystem catalog cannot be loaded if a program has already been loaded. The RESTART-PROGRAM command is rejected while a local subsystem catalog is loaded. Before the program can be restarted the local subsystem catalog must be explicitly unloaded.

Format

LOAD-LOCAL-**SUBSYS**TEM-**CAT**ALOG
CATALOG-**NA**ME = <filename 1..54 without-gen-vers>

Operands

CATALOG-NAME = <filename 1..54 without-gen-vers>
Name of the local subsystem catalog that is to be loaded into the task-specific user address space (class 5 memory).

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	64	ESM0255	Command not executed; a local catalog or a program is already loaded
	64	ESM0326	Processing of the command terminated; (system error, non-compatible catalog versions, DMS error, catalog contains privileged subsystems or the specified file is not a catalog)

Notes

- The size of the memory area occupied by the local subsystem catalog in the user address space is solely dependent on the size of the subsystem definitions in the catalog.
  There is therefore no need for any expansion of memory space later.
- If the local subsystem catalog contains too many subsystems or if they are too large, the user address space may become saturated. This problem can be prevented by loading the local subsystem catalog above the 16-Mbyte boundary.
- Dependency relations or other connections between subsystems are not taken into account in local subsystem management.
- If the WRCPT macro is called in a program while a local subsystem catalog is loaded, it saves the local subsystem configuration to the checkpoint file. If the program is resumed with the RESTART-PROGRAM command, the local subsystem configuration can be read back in from this file.

Example

```
/load-local-subsystem-catalog catalog-name=local.dssmcat.1 ——————————— (1)
%ESM0254 COMMAND '/LOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/load-local-subsystem-catalog catalog-name=local.dssmcat.2 ——————————— (2)
%ESM0344 A LOCAL CATALOG IS ALREADY LOADED
%ESM0255 COMMAND '/LOAD-LOCAL-SUBSYSTEM-CATALOG' NOT PROCESSED
```

(1) First the local subsystem catalog LOCAL.DSSMCAT.1 is loaded. Execution of the command is successful.

(2) Loading of another local subsystem catalog is rejected with message ESM0344: only one file can be loaded as a local subsystem catalog at any one time.

MODIFY-SUBSYSTEM-PARAMETER Modify subsystem parameters

Domain:	SYSTEM-MANAGEMENT
Privileges:	SUBSYSTEM-MANAGEMENT

Function

A user with the SUBSYSTEM-MANAGEMENT privilege can run this command to modify the parameters governing a subsystem; only the parameters explicitly specified are changed.

The command modifies only the dynamic subsystem catalog, not the static subsystem catalog. Changes made are therefore only effective during the current session but not at the next startup.
You can use the SAVE-SUBSYSTEM-CATALOG command to store the changes into a static catalog and make them permanent. You must however note that some changes may be pointless or even unfavorable at the next startup (e.g. if a message file is assigned to a subsystem with the start attribute BEFORE-DSSM-LOAD[1]).

i Only users who are very familiar with the subsystem that is being modified should use this command. This is because the command is capable of making far-reaching changes to the subsystem attributes.

The command has three different types of operand:

- operands which have their value stored in the dynamic subsystem catalog and which take immediate effect (such as VERSION-COEXISTENCE).
- operands which have their value stored in the dynamic subsystem catalog but do not take effect until the next /START-SUBSYSTEM (such as SUBSYSTEM-LIBRARY).
- operands which are accepted only if the subsystem is not currently running (such as MESSAGE-FILE).

[1] The BEFORE-DSSM-LOAD attribute can only be changed via SSCM, see chapter “SSCM” on page 179.

Format

(part 1 of 2)

```
MODIFY-SUBSYSTEM-PARAMETER

 SUBSYSTEM-NAME = <structured-name 1..8>
,VERSION = <product-version mandatory-man-corr> / <product-version without-man-corr>
,INSTALLATION-UNIT = *UNCHANGED /  *NONE / *STD / <text 1..30>
,INSTALLATION-USERID = *UNCHANGED / *NONE / <name 1..8> / *DEFAULT-USERID
,COPYRIGHT = *UNCHANGED / *NONE / <c-string 1..54>(...)
    <c-string 1..54>(...)
      |   YEAR = *YEAR-1990 / <c-string 4..4>
,SUBSYSTEM-LIBRARY = *UNCHANGED / *STD / *INSTALLED(...) / <filename 1..54 without-gen-vers>
    *INSTALLED(...)
      |   LOGICAL-ID = *UNCHANGED / *REFRESH / <filename 1..30 without-catid-userid-gen-vers>
      |  ,DEFAULT-NAME = *UNCHANGED / <filename 1..54>
,SUBSYSTEM-LOAD-MODE = *UNCHANGED / *STD / *ADVANCED
,REP-FILE = *UNCHANGED / *STD / *NO / *INSTALLED(...) / <filename 1..54 without-gen-vers>
    *INSTALLED(...)
      |   LOGICAL-ID = *UNCHANGED / *REFRESH / <filename 1..30 without-catid-userid-gen-vers>
      |  ,DEFAULT-NAME = *UNCHANGED / *NONE/ <filename 1..54>
,REP-FILE-MANDATORY = *UNCHANGED / *NO / *YES
,MESSAGE-FILE = *UNCHANGED / *NO / *INSTALLED(...) / <filename 1..54 without-gen-vers>
    *INSTALLED(...)
      |   LOGICAL-ID = *UNCHANGED / *REFRESH / <filename 1..30 without-catid-userid-gen-vers>
      |  ,DEFAULT-NAME = *UNCHANGED / *NONE/ <filename 1..54>
,SUBSYSTEM-INFO-FILE = *UNCHANGED / *NO / *INSTALLED(...) / <filename 1..54 without-gen-vers>
    *INSTALLED(...)
      |   LOGICAL-ID = *UNCHANGED / *REFRESH / <filename 1..30 without-catid-userid-gen-vers>
      |  ,DEFAULT-NAME = *UNCHANGED / *NONE/ <filename 1..54>
,SYNTAX-FILE = *UNCHANGED / *NO / *INSTALLED(...) / <filename 1..54 without-gen-vers>
    *INSTALLED(...)
      |   LOGICAL-ID = *UNCHANGED / *REFRESH / <filename 1..30 without-catid-userid-gen-vers>
      |  ,DEFAULT-NAME = *UNCHANGED / *NONE/ <filename 1..54>
,DYNAMIC-CHECK-ENTRY = *UNCHANGED / *STD / *NO / <text 1..8 without-sep>
```

Continued ➠

(part 2 of 2)

```
,CREATION-TIME = *UNCHANGED / *AT-CREATION-REQUEST / *AT-SUBSYSTEM-CALL(...) /
                     *AT-DSSM-LOAD / *MANDATORY-AT-STARTUP / *BEFORE-SYSTEM-READY /
                     *AFTER-SYSTEM-READY

   *AT-SUBSYSTEM-CALL(...)
     |   ON-ACTION = *STD / *ISL-CALL /  *ALL

,INIT-ROUTINE = *UNCHANGED / *NO / <text 1..8 without-sep>

,CLOSE-CTRL-ROUTINE = *UNCHANGED / *NO / *DYNAMIC / <text 1..8 without-sep>

,STOPCOM-ROUTINE = *UNCHANGED / *NO / *DYNAMIC / <text 1..8 without-sep>

,DEINIT-ROUTINE = *UNCHANGED / *NO / *DYNAMIC / <text 1..8 without-sep>

,STOP-AT-SHUTDOWN = *UNCHANGED / *NO / *YES

,INTERFACE-VERSION = *UNCHANGED / *NO / <text 1..8 without-sep>

,SUBSYSTEM-HOLD = *UNCHANGED / *ALLOWED / *FORBIDDEN

,STATE-CHANGE-CMDS = *UNCHANGED / *ALLOWED / *FORBIDDEN / *BY-ADMINISTRATOR-ONLY

,FORCED-STATE-CHANGE = *UNCHANGED / *ALLOWED / *FORBIDDEN

,RESET = *UNCHANGED / *ALLOWED / *FORBIDDEN

,RESTART-REQUIRED = *UNCHANGED / *NO / *YES

,VERSION-COEXISTENCE = *UNCHANGED / *FORBIDDEN / *ALLOWED

,VERSION-EXCHANGE = *UNCHANGED / *FORBIDDEN / *ALLOWED

,MEMORY-CLASS = *UNCHANGED / *SYSTEM-GLOBAL(...) / *LOCAL-UNPRIVILEGED(...) /*BY-SLICE(...)

   *SYSTEM-GLOBAL(...)
     |   SUBSYSTEM-ACCESS = *LOW / *HIGH

   *LOCAL-UNPRIVILEGED(...)
     |   SIZE = *UNCHANGED / <integer 1..32767 4Kbyte>
     |
     |  ,SUBSYSTEM-ACCESS = *UNCHANGED / *LOW / *HIGH
     |
     |  ,START-ADDRESS = *UNCHANGED / *ANY / <x-string 7..8>

   *BY-SLICE(...)
     |   SIZE = <integer 1..32767 4Kbyte>

,LINK-ENTRY = *UNCHANGED (...) / <text 1..8 without-sep>(...)

   *UNCHANGED(...)
     |   AUTOLINK = *UNCHANGED / *ALLOWED / *FORBIDDEN

   <text 1..8 without-sep>(...)
     |   AUTOLINK = *ALLOWED / *FORBIDDEN

,UNRESOLVED-EXTERNALS = *UNCHANGED / *ALLOWED / *FORBIDDEN

,CHECK-REFERENCES = *UNCHANGED / *YES / *NO

,CHANGE-STATE = *UNCHANGED / *YES / *NO
```

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Specifies the name of the subsystem for which the parameters are to be changed.

VERSION = <product-version mandatory-man-corr> /
<product-version without-man-corr>
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type "product-version" on page 13).

INSTALLATION-UNIT =
Defines the name of the installed software unit (also referred to as a release unit). A value other than *NONE must be specified for all subsystems installed with IMON-GPN, and also if the value *INSTALLED(LOGICAL-ID=...) was defined for one of the operands SUBSYSTEM-LIBRARY, REP-FILE, SUBSYSTEM-INFO-FILE, MESSAGE-FILE or SYNTAX-FILE.

INSTALLATION-UNIT = <u>*UNCHANGED</u>
The name of the installed software unit remains unchanged.

INSTALLATION-UNIT = *NONE
No name is assigned. This entry is not allowed for any subsystems installed with IMON.

INSTALLATION-UNIT = *STD
The name specified with the SUBSYSTEM-NAME operand is used as the name of the installed software unit.

INSTALLATION-UNIT = <text 1..30>
New name of the installed software unit.

INSTALLATION-USERID = <u>*UNCHANGED</u> / *NONE / <name 1..8> / *DEFAULT-USERID
Changes the default user ID of the files associated with the subsystem (operands REP-FILE, SUBSYSTEM-LIBRARY, SYNTAX-FILE and MESSAGE-FILE, SUBSYSTEM-INFO-FILE). File names specified without a user ID are assumed to belong to the new installation user ID defined here.
Any attempt to change the installation user ID is rejected if the subsystem is active and there is a message file (MESSAGE-FILE operand) or a syntax file (SYNTAX-FILE operand) assigned to it with no user ID specified.
The change takes effect immediately.

INSTALLATION-USERID = *UNCHANGED
The installation user ID is not changed.

INSTALLATION-USERID = *NONE
Removes the installation user ID.
If an installation user ID existed before the command was issued, it is stripped from all files to which it was assigned.

INSTALLATION-USERID = <name 1..8>
The user ID given here will be the new installation user ID. The name of the user ID must be given without dollar sign “$”.

INSTALLATION-USERID = *DEFAULT-USERID
Selects the system default user ID as the installation user ID (which means that files begin with “$.”).

COPYRIGHT = *UNCHANGED / *NONE / <c-string 1..54>(...)
Changes the copyright notice displayed when the subsystem is loaded.
The change takes effect as soon as the subsystem is restarted (START-SUBSYSTEM command).

COPYRIGHT = *UNCHANGED
The copyright notice is not changed.

COPYRIGHT = *NONE
No copyright notice is displayed.

COPYRIGHT = <c-string 1..54>(...)
Changes the copyright notice displayed when the subsystem is loaded.

> **YEAR = *YEAR-1990 / <c-string 4..4>**
> Defines the year displayed as the first production year in the copyright notice. The default year is 1990. Any other year must be specified explicitly. Note that the operand value is not subjected to semantic validity checking.

SUBSYSTEM-LIBRARY = *UNCHANGED / *STD / *INSTALLED(...) / <filename 1..54 without-gen-vers>
Changes the module library assignment for the specified subsystem. The module library supplies the code which is loaded for a subsystem which is not yet running.
The change takes effect as soon as the subsystem is restarted (START-SUBSYSTEM command).

SUBSYSTEM-LIBRARY = *UNCHANGED
The setting is left unchanged.

SUBSYSTEM-LIBRARY = *STD
Selects the default name SYSLNK.<subsysname>.<subsysvers#> for the library name argument.
"subsysvers#" is a three-character value composed of the "mmm" elements specified in the operand SUBSYSTEM-NAME=...(VERSION=...).

SUBSYSTEM-LIBRARY = *INSTALLED(...)
The library ID is determined by calling IMON-GPN (administration of installation paths).

LOGICAL-ID =
Specifies the logical ID of the program library or object module library under which the library is known to IMON.

LOGICAL-ID = <u>*UNCHANGED</u>
The logical name of the program library or object module library remains unchanged.

LOGICAL-ID = *REFRESH
The path name belonging to the logical name has been changed and is now to be updated in the catalog. The logical ID itself is unchanged.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
New logical ID of the program library or object module library.

DEFAULT-NAME =
Name of the library if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <u>*UNCHANGED</u>
The library name remains unchanged.

DEFAULT-NAME = <filename 1..54>
New library name.

SUBSYSTEM-LIBRARY = <filename 1..54 without-gen-vers>
The fully qualified file name specified here is defined as the new library name (see "Notes" on page 109).

SUBSYSTEM-LOAD-MODE = <u>*UNCHANGED</u> / *STD / *ADVANCED
Defines the way in which the subsystem is started.

SUBSYSTEM-LOAD-MODE = <u>*UNCHANGED</u>
The subsystem loading mode is left unchanged, i.e. the manner in which the subsystem is started is not changed.

SUBSYSTEM-LOAD-MODE = *STD
BLS is invoked in STD run mode (via the BLS-DSSM interface $PBBND1) to load the subsystem code as an object module.

SUBSYSTEM-LOAD-MODE = *ADVANCED
BLS is called in ADVANCED run mode (via the BLS-DSSM interface $PBBND1) to load the subsystem code as a link and load module.

REP-FILE = <u>*UNCHANGED</u> / *STD / *NO / *INSTALLED(...) / <filename 1..54 without-gen-vers>
Changes the REP file assignment for the specified subsystem version. REP files are designed to incorporate module updates in a subsystem which is not currently running. The change takes effect as soon as the subsystem is restarted (START-SUBSYSTEM command).

REP-FILE = <u>*UNCHANGED</u>
The REP file parameters are not changed.

REP-FILE = *STD
The name of the REP file is the default name: SYSREP.<subsysname>.<subsysvers#>. "subsysvers#" is a three-character value composed of the "mmm" elements specified in the operand SUBSYSTEM-NAME=...(VERSION=...).

REP-FILE = *NO
There is no REP file for the subsystem.

REP-FILE = *INSTALLED(...)
The name of the REP file is determined by calling IMON-GPN (administration of installation paths).

LOGICAL-ID =
Specifies the logical ID of the REP file under which the file is known to IMON.

LOGICAL-ID = <u>*UNCHANGED</u>
The logical ID of the REP file remains unchanged.

LOGICAL-ID = *REFRESH
The path name belonging to the logical ID has been changed and is now to be updated in the catalog. The logical ID itself is unchanged.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
New logical ID of the REP file.

DEFAULT-NAME =
Name of the REP file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <u>*UNCHANGED</u>
The name of the REP file remains unchanged.

DEFAULT-NAME = *NONE
No default name is assigned for the REP file.

DEFAULT-NAME = <filename 1..54>
New name of the REP file.

REP-FILE = <filename 1..54 without-gen-vers>
The fully qualified file name specified here is defined as the new REP file name (see “Notes” on page 109).

REP-FILE-MANDATORY = <u>*UNCHANGED</u> / *NO / *YES
Defines whether the subsystem is started if errors occur while the REP file is being processed. The change takes effect as soon as the subsystem is restarted (START-SUBSYSTEM command).

REP-FILE-MANDATORY = <u>*UNCHANGED</u>
The current setting is left unchanged.

REP-FILE-MANDATORY = *NO
Errors during processing of the REP file have no effect on subsystem loading.

REP-FILE-MANDATORY = *YES
Dynamic subsystem management (DSSM) inhibits subsystem loading in the following cases:

- DMS errors during REP file processing (e.g. REP file not cataloged)
- errors during REP file validation
- REP file incorrectly named
- DMS errors during NOREF file processing

MESSAGE-FILE = <u>*UNCHANGED</u> / *NO / *INSTALLED(...) / <filename 1..54 without-gen-vers>
Changes the message file definition valid for the specified subsystem version.
The subsystem version must not be running at the time.
The requirements placed by DVS on the file name are not checked.

MESSAGE-FILE = <u>*UNCHANGED</u>
The current setting is left unchanged.

MESSAGE-FILE = *NO
No subsystem-specific message file is available.
This setting is needed for cases where BEFORE-DSSM-LOAD is defined as the activation point for the subsystem, see chapter “SSCM” on page 179.

MESSAGE-FILE = *INSTALLED(...)
The name of the message file is determined by calling IMON-GPN (administration of installation paths).

LOGICAL-ID =
Specifies the logical ID of the message file under which the file is known to IMON.

LOGICAL-ID = *UNCHANGED
The logical ID of the message file remains unchanged.

LOGICAL-ID = *REFRESH
The path name belonging to the logical ID has been changed and is now to be updated in the catalog. The logical ID itself is unchanged.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
New logical ID of the message file.

DEFAULT-NAME =
Name of the message file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = *UNCHANGED
The name of the message file remains unchanged.

DEFAULT-NAME = *NONE
No default name is assigned for the message file.

DEFAULT-NAME = <filename 1..54>
New name of the message file.

MESSAGE-FILE = <filename 1..54 without-gen-vers>
The fully qualified file name specified here is defined as the new message file name (see “Notes” on page 109).

SUBSYSTEM-INFO-FILE = *UNCHANGED / *NO / *INSTALLED(...) / <filename 1..54 without-gen-vers>
Specifies which information file to use for the specified subsystem version.

SUBSYSTEM-INFO-FILE = *UNCHANGED
The current setting is left unchanged.

SUBSYSTEM-INFO-FILE = *NO
No information file is available.

SUBSYSTEM-INFO-FILE = *INSTALLED(...)
The name of the information file is determined by calling IMON-GPN (administration of installation paths).

LOGICAL-ID =
Specifies the logical ID of the information file under which the file is known to IMON.

LOGICAL-ID = <u>*UNCHANGED</u>
The logical ID of the information file remains unchanged.

LOGICAL-ID = *REFRESH
The path name belonging to the logical ID has been changed and is now to be updated in the catalog. The logical ID itself is unchanged.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
New logical ID of the information file.

DEFAULT-NAME =
Name of the information file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <u>*UNCHANGED</u>
The name of the information file remains unchanged.

DEFAULT-NAME = *NONE
No default name is assigned for the information file.

DEFAULT-NAME = <filename 1..54>
New name of the information file.

SUBSYSTEM-INFO-FILE = <filename 1..54 without-gen-vers>
The fully qualified file name specified here is defined as the new information file name (see “Notes” on page 109).

SYNTAX-FILE = <u>*UNCHANGED</u> / *NO / *INSTALLED(...) /
<filename 1..54 without-gen-vers>
Changes the syntax file definition valid for the specified subsystem version.
The syntax file contains the command and operand values valid for the subsystem version.
The subsystem version must not be loaded when the command is issued.
The requirements placed by DVS on the file name are not checked.

SYNTAX-FILE = <u>*UNCHANGED</u>
The current setting is left unchanged.

SYNTAX-FILE = *NO
No syntax file is available.
This setting is needed for cases where BEFORE-DSSM-LOAD or AT-DSSM-LOAD is defined as the activation (creation) point for the subsystem.

SYNTAX-FILE = *INSTALLED(...)
The name of the syntax file is determined by calling IMON-GPN (administration of installation paths).

LOGICAL-ID =
Specifies the logical ID of the syntax file under which the file is known to IMON.

LOGICAL-ID = <u>*UNCHANGED</u>
The logical ID of the syntax file remains unchanged.

LOGICAL-ID = *REFRESH
The path name belonging to the logical ID has been changed and is now to be updated in the catalog. The logical ID itself is unchanged.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
New logical ID of the syntax file.

DEFAULT-NAME =
Name of the syntax file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <u>*UNCHANGED</u>
The name of the syntax file remains unchanged.

DEFAULT-NAME = *NONE
No default name is assigned for the syntax file.

DEFAULT-NAME = <filename 1..54>
New name of the syntax file.

SYNTAX-FILE = <filename 1..54 without-gen-vers>
The fully qualified file name specified here is defined as the new syntax file name (see “Notes” on page 109).

DYNAMIC-CHECK-ENTRY = <u>*UNCHANGED</u> / *STD / *NO / <text 1..8 without-sep>
Changes the reference address used to check that the loaded code for the subsystem is correct.

DYNAMIC-CHECK-ENTRY = <u>*UNCHANGED</u>
The current setting is left unchanged.

DYNAMIC-CHECK-ENTRY = *STD
The reference address specified in the LINK-ENTRY operand is used as the reference address for dynamic checking.

DYNAMIC-CHECK-ENTRY = *NO
No checking is performed. This setting is not allowed if the activation point defined for the subsystem is BEFORE-DSSM-LOAD.

DYNAMIC-CHECK-ENTRY = <text 1..8 without-sep>
The address specified here is defined as the new reference address for dynamic checking.

CREATION-TIME = *UNCHANGED / *AT-CREATION-REQUEST / *AT-SUBSYSTEM-CALL(...) / *AT-DSSM-LOAD / *MANDATORY-AT-STARTUP / *BEFORE-SYSTEM-READY / *AFTER-SYSTEM-READY
Changes the creation time of a subsystem.

CREATION-TIME = *UNCHANGED
The current setting is left unchanged.

CREATION-TIME = *AT-CREATION-REQUEST
The subsystem creation time is reset to the (generation) default setting of "creation on START-SUBSYSTEM invocation".

CREATION-TIME = *AT-SUBSYSTEM-CALL(...)
The subsystem creation time is changed to the value AT-SUBSYSTEM-CALL. This means that the subsystem starts up automatically in response to the first SVC or ISL call.
This operand value is available only for subsystems called by the SVC or ISL mechanism.

The operand value can be assigned to the specified subsystem version after it has been withdrawn, if necessary, from another subsystem version. If no other version has this attribute, it is directly transferred to the specified subsystem version. If some other subsystem version has this attribute, it is withdrawn from that subsystem, either immediately (if the subsystem is not running) or after the subsystem has closed down (if it is currently running).
Subsystems with the attribute VERSION-COEXISTENCE=*ALLOWED are an exception to this rule. With them, different versions of the same subsystem can have the attribute CREATION-TIME=*AT-SUBSYSTEM-CALL at the same time.
As with SSCM, this attribute can only be set for a subsystem with CALL entry MODE=*SVC / *ISL.

ON-ACTION =
Determines what initiates automatic loading of the subsystem.

ON-ACTION = *STD
Default setting: loading begins when any SVC entry point belonging to the subsystem is called.

ON-ACTION = *ISL-CALL
Loading begins when any ISL entry point belonging to the subsystem is called.

ON-ACTION = *ANY
Loading begins when any SVC or ISL entry point belonging to the subsystem is called.

CREATION-TIME = *AT-DSSM-LOAD
The subsystem is to be loaded under the control of the DSSM task during system initialization. It must be a privileged subsystem, and any address and dependency relations it has must be with subsystems which also have this startup attribute or the BEFORE-DSSM-LOAD startup attribute.
The file name for this subsystem must be located on the home pubset under the TSOS user ID, as at startup time the user catalog is not accessible and IMPORT-PUBSET processing has not been completed.
It is not permitted to link in a syntax file for these subsystems.

CREATION-TIME = *MANDATORY-AT-STARTUP
The subsystem must be loaded during system initialization (phase 2: after DSSM has been loaded; see page 212). As with BEFORE-SYSTEM-READY, subsystem activation is initiated synchronously; but in this case, as opposed to BEFORE-SYSTEM-READY, loading of the subsystem must be **completed successfully**. Otherwise the startup routine is sent a message indicating that a mandatory subsystem could not be loaded. The startup routine then decides whether to continue or abort processing.
The subsystem must be a privileged subsystem, and any address and dependency relations it has must be with subsystems which have the same start attribute or one of the start attributes BEFORE-DSSM-LOAD or AT-DSSM-LOAD. The file name for this subsystem must be cataloged on the home pubset.

CREATION-TIME = *BEFORE-SYSTEM-READY
The subsystem is to be loaded during system initialization (phase 2; see page 212). Activation is initiated synchronously; control is not returned to the startup routine until after completion of loading (or a loading error). Once the startup routine has regained control, it can report “SYSTEM READY”.
The subsystem must be a privileged subsystem, and any address and dependency relations it has must be with subsystems which have the same start attribute or one of the start attributes BEFORE-DSSM-LOAD, AT-DSSM-LOAD or MANDATORY-AT-STARTUP.
The file name for this subsystem must be cataloged on the home pubset.

CREATION-TIME = *AFTER-SYSTEM-READY
Loading of the subsystem is initiated during system initialization (phase 2; see page 212). Loading is not synchronized with the startup routine, which can report “SYSTEM READY” before loading of this subsystem has been completed.
Any address and dependency relations the subsystem has must be with subsystems which have the same start attribute or one of the start attributes BEFORE-DSSM-LOAD, AT-DSSM-LOAD, MANDATORY-AT-STARTUP or BEFORE-SYSTEM-READY.
The files for this subsystem must be on the home pubset.

INIT-ROUTINE = *UNCHANGED / *NO / <text 1..8 without-sep>
Changes the subsystem initialization routine (see page 214), provided that this does not affect the manner in which the subsystem functions.
The change takes effect immediately to allow the subsystem to be reconstructed if necessary.

INIT-ROUTINE = *UNCHANGED
The current setting is left unchanged.

INIT-ROUTINE = *NO
No initialization routine is carried out.

INIT-ROUTINE = <text 1..8 without-sep>
The name specified here is defined as the new reference address name for the subsystem (see "Notes" on page 109).

CLOSE-CTRL-ROUTINE = *UNCHANGED / *NO / *DYNAMIC / <text 1..8 without-sep>
Changes the subsystem's CLOSE-CTRL routine (see page 215 or page 257), provided that this does not affect the manner in which the subsystem functions. The change takes effect immediately.

CLOSE-CTRL-ROUTINE = *UNCHANGED
The current setting is left unchanged.

CLOSE-CTRL-ROUTINE = *NO
DSSM processes the STOP-SUBSYSTEM and HOLD-SUBSYSTEM commands without calling a CLOSE-CTRL routine.

CLOSE-CTRL-ROUTINE = *DYNAMIC
The CLOSE-CTRL routine is called by the subsystem dynamically at the end of the initialization (INIT) routine. A reference address name must be defined for the routine
When the CLOSE-CTRL routine is called, the holder task of the subsystem must be a work task (ASSIGN-HOLDER-TASK statement, page 193).

CLOSE-CTRL-ROUTINE = <text 1..8 without-sep>
The name specified here is defined as the new reference address name for the subsystem (see "Notes" on page 109).

STOPCOM-ROUTINE = *UNCHANGED / *NO / *DYNAMIC / <text 1..8 without-sep>
Changes the subsystem's STOPCOM routine (see page 215 or page 257), provided that this does not affect the manner in which the subsystem functions. The change takes effect immediately.

STOPCOM-ROUTINE = *UNCHANGED
The current setting is left unchanged.

STOPCOM-ROUTINE = *NO
DSSM processes the STOP-SUBSYSTEM and HOLD-SUBSYSTEM commands without calling a STOPCOM routine.

STOPCOM-ROUTINE = *DYNAMIC
The STOPCOM routine is called dynamically by the subsystem at the end of the CLOSE-CTRL routine or, if this has not been defined, at the end of the initialization (INIT) routine. When the STOPCOM routine is called, the holder task of the subsystem must be a work task (ASSIGN-HOLDER-TASK statement, page 193).

STOPCOM-ROUTINE = <text 1..8 without-sep>
The name specified here is defined as the new reference address name for the STOPCOM routine (see “Notes” on page 109).

DEINIT-ROUTINE = *UNCHANGED / *NO / *DYNAMIC / <text 1..8 without-sep>
Changes the subsystem’s DEINIT routine (see page 216 or page 259), provided that this does not affect the manner in which the subsystem functions. The change takes effect immediately.

DEINIT-ROUTINE = <u>*UNCHANGED</u>
The current setting is left unchanged.

DEINIT-ROUTINE = *NO
DSSM processes the STOP-SUBSYSTEM and HOLD-SUBSYSTEM commands without invoking a DEINIT routine.

DEINIT-ROUTINE = *DYNAMIC
The DEINIT routine is called dynamically by the subsystem at the end of the STOPCOM routine or, if this has not been defined, at the end of the CLOSE-CTRL routine. If this is also not defined, the DEINIT routine is called at the end of the initialization (INIT) routine
When the DEINIT routine is called, the holder task of the subsystem must be used as a work task (ASSIGN-HOLDER-TASK statement, page 193).

DEINIT-ROUTINE = <text 1..8 without-sep>
The name specified here is defined as the new reference address name for the DEINIT routine (see “Notes” on page 109).

STOP-AT-SHUTDOWN = <u>*UNCHANGED</u> / *NO / *YES
Causes DSSM to close down the subsystem automatically as soon as the SHUTDOWN command (terminate session) is issued. The change takes effect immediately.

STOP-AT-SHUTDOWN = <u>*UNCHANGED</u>
The current setting is left unchanged.

STOP-AT-SHUTDOWN = *NO
DSSM ignores the subsystem when the SHUTDOWN command is issued.

STOP-AT-SHUTDOWN = *YES
DSSM closes down the subsystem as soon as the SHUTDOWN command is issued (as with /STOP-SUBSYSTEM).

INTERFACE-VERSION = <u>*UNCHANGED</u> / *NO / <text 1..8 without-sep>
Designates the entry point via which DSSM can access the interface version used for decoupled calling of the INIT, CLOSE-CTRL, STOPCOM and DEINIT routines.
The change takes effect immediately.

INTERFACE-VERSION = <u>*UNCHANGED</u>
The current setting is left unchanged.

INTERFACE-VERSION = *NO
None of the following entry points is available:
INIT-ROUTINE, DEINIT-ROUTINE, STOPCOM-ROUTINE, CLOSE-CTRL-ROUTINE.

INTERFACE-VERSION = <text 1..8 without-sep>
The name specified here is defined as the new entry point.

SUBSYSTEM-HOLD = <u>*UNCHANGED</u> / *ALLOWED / *FORBIDDEN
Defines whether a command or macro can be used to halt or unload the subsystem.
The change takes effect immediately.

SUBSYSTEM-HOLD = <u>*UNCHANGED</u>
The current setting is left unchanged.

SUBSYSTEM-HOLD = *ALLOWED
A command or macro can be used to halt or unload the subsystem.

SUBSYSTEM-HOLD = *FORBIDDEN
As with the STOP-AT-SHUTDOWN operand value, the subsystem cannot be unloaded until the BS2000 system is closed down by means of the SHUTDOWN command.

STATE-CHANGE-CMDS = <u>*UNCHANGED</u> / *ALLOWED / *FORBIDDEN / *BY-ADMINISTRATOR-ONLY
Defines whether the DSSM commands START-SUBSYSTEM, RESUME-SUBSYSTEM, STOP-SUBSYSTEM and HOLD-SUBSYSTEM are allowed for the subsystem.
The change takes effect immediately.
In the event of a version swap this entry is only of relevance for the new version that is to be activated. Deactivation of the other, “old” version is always carried out.

STATE-CHANGE-CMDS = <u>*UNCHANGED</u>
The current setting is left unchanged.

STATE-CHANGE-CMDS = *ALLOWED
The listed commands can be issued from the console or under a user ID with the SUBSYSTEM-MANAGEMENT privilege.

STATE-CHANGE-CMDS = *FORBIDDEN
The listed commands cannot be issued either from the console or under a user ID with the SUBSYSTEM-MANAGEMENT privilege.

STATE-CHANGE-CMDS = *BY-ADMINISTRATOR-ONLY
The listed commands can be issued under a user ID with the SUBSYSTEM-MANAGEMENT privilege but not from the console.

FORCED-STATE-CHANGE = <u>*UNCHANGED</u> / *ALLOWED / *FORBIDDEN
Defines whether the FORCED operand of the DSSM commands STOP-SUBSYSTEM and HOLD-SUBSYSTEM is allowed for the subsystem.
The change takes effect immediately.

FORCED-STATE-CHANGE = <u>*UNCHANGED</u>
The current setting is left unchanged.

FORCED-STATE-CHANGE = *ALLOWED
Use of the FORCED operand in these commands is allowed.

FORCED-STATE-CHANGE = *FORBIDDEN
Use of the FORCED operand is forbidden.

RESET = <u>*UNCHANGED</u> / *ALLOWED / *FORBIDDEN
Defines whether the operand RESET=*YES of the DSSM commands START-SUBSYSTEM and RESUME-SUBSYSTEM is allowed for the subsystem.
The change takes effect immediately.

RESET = <u>*UNCHANGED</u>
The current setting is left unchanged.

RESET = *ALLOWED
The DSSM commands START-SUBSYSTEM and RESUME-SUBSYSTEM are accepted if issued with the operand RESET=*YES.

RESET = *FORBIDDEN
The DSSM commands START-SUBSYSTEM and RESUME-SUBSYSTEM are rejected if issued with the operand RESET=*YES.

RESTART-REQUIRED = *UNCHANGED / *NO / *YES
Defines whether the initialization (INIT) routine is invoked to restart the subsystem in the event of abnormal holder task termination.
The initialization routine is invoked during reconstruction of the holder task.
The change takes effect immediately.

RESTART-REQUIRED = *UNCHANGED
The current setting is left unchanged.

RESTART-REQUIRED = *NO
In the event of abnormal holder task termination, the subsystem is locked during reconstruction of the holder task.

RESTART-REQUIRED = *YES
The initialization routine is invoked to restart the system.

VERSION-COEXISTENCE = *UNCHANGED / *FORBIDDEN / *ALLOWED
Defines whether different versions of the subsystem can be active at the same time.
The change takes effect immediately.

VERSION-COEXISTENCE = *UNCHANGED
The current setting is left unchanged.

VERSION-COEXISTENCE = *FORBIDDEN
Only one version of the subsystem can be active.

VERSION-COEXISTENCE = *ALLOWED
Different versions of the subsystem can be active at the same time.

VERSION-EXCHANGE = *UNCHANGED / *FORBIDDEN / *ALLOWED
Defines whether a new subsystem version can be activated without having to delete the old version. The change takes effect immediately.

VERSION-EXCHANGE = *UNCHANGED
The current setting is left unchanged.

VERSION-EXCHANGE = *FORBIDDEN
A new subsystem version cannot be activated unless the old version has been completely deleted.

VERSION-EXCHANGE = *ALLOWED
A subsystem version can be activated without having to delete the other version.

MEMORY-CLASS = <u>*UNCHANGED</u> / *SYSTEM-GLOBAL(...) / *LOCAL-UNPRIVILEGED(...) / *BY-SLICE
Changes the subsystem’s memory class or defines the subsystem’s position in main memory (above or below 16 Mbytes). Note the following:

- A privileged subsystem cannot be changed to a nonprivileged subsystem.
- If the memory class is changed, all suboperands must be specified.
- A subsystem cannot be made LOCAL-UNPRIVILEGED if there is an address overlap between two LOCAL-UNPRIVILEGED subsystems sharing the same holder task.

The change takes effect as soon as the subsystem is restarted (START-SUBSYSTEM command).

MEMORY-CLASS = <u>*UNCHANGED</u>
The current setting is left unchanged.

MEMORY-CLASS = *SYSTEM-GLOBAL(...)
The memory class of the subsystem is changed to class 3 or class 4 memory.

SUBSYSTEM-ACCESS = <u>*LOW</u> / *HIGH
Defines the rights of access to the requested space and the location of the requested space in the address space.

SUBSYSTEM-ACCESS = <u>*LOW</u>
Nonprivileged address space is requested. The main memory location is below 16 Mbytes.

SUBSYSTEM-ACCESS = *HIGH
Nonprivileged address space up to 2 Gbytes is requested.

MEMORY-CLASS = *LOCAL-UNPRIVILEGED(...)
The memory pool is set up as class 6 memory (only for subsystems which are to be executed in the same way as programs).

SIZE = <u>*UNCHANGED</u> / <integer 1..32767 *4Kbyte*>
Defines the size of the address space required for the memory pool in 4K pages. The defined value must be large enough to hold the subsystem and all the selectable units and load units dynamically loaded by the subsystem.

SIZE = <u>*UNCHANGED</u>
The current setting is left unchanged.

SIZE = <integer 1..32767 *4Kbyte*>
The address space specified here defines the size of the memory pool.

SUBSYSTEM-ACCESS = *UNCHANGED / *LOW / *HIGH
Defines the rights of access to the requested space and the location of the requested space in the address space.

SUBSYSTEM-ACCESS = *UNCHANGED
The current setting is left unchanged.

SUBSYSTEM-ACCESS = *SYSTEM
Privileged address space is requested. The load address is above 16 Mbytes.

SUBSYSTEM-ACCESS = *LOW
Nonprivileged address space is requested. The load address is below 16 Mbytes.

SUBSYSTEM-ACCESS = *HIGH
Nonprivileged address space up to 2 gigabytes is requested.

START-ADDRESS = *UNCHANGED / *ANY / <x-string 7..8>
Defines the load address of the subsystem. This must be a multiple of X'100000'. It is the responsibility of the user to specify an address pointing to class 6 memory.

START-ADDRESS = *UNCHANGED
The current setting is left unchanged.

START-ADDRESS = *ANY
The location of the subsystem in class 6 memory is defined by DSSM.

START-ADDRESS = <x-string 7..8>
Address in the segment raster at which the base address of the subsystem is to be located. The address must be a multiple of x'100000'.

MEMORY-CLASS = *BY-SLICE(...)
The specified subsystem is a nonprivileged subsystem and consists of an LLM, which in turn consists of a shareable code (program area) and a non-shareable code (data area). The program area is loaded into the shareable address space (this corresponds to MEMORY-CLASS=*SYSTEM-GLOBAL). The data area is loaded into the user address space of the holder task and is copied into the private user address spaces of the connected tasks at the same address.
If the subsystem is defined with *BY-SLICE, the following points must be borne in mind:

– If a reserved address space for the data area already exists, the command is executed only if this address space actually has enough free space to accommodate the modified subsystem.

– If no address space has been reserved to accommodate the data area, then address space is created. Tasks connected to the subsystem when reserved address space is created cannot use the data area.

When a task is first connected to a subsystem that was defined with *BY-SLICE, DSSM informs the BLSSERV subsystem that the copy of the data area in the private user address space can be accessed with the VSVI1 macro.
The VSVI1 macro informs the user about entries in the DBL tables. See the manual "BLSSERV" [4] for details on the macro.
When the last connection is shut down, DSSM informs the BLSSERV subsystem that this private area can no longer be accessed.

DSSM only accepts an address space change in the new *BY-SLICE attribute if MODE=*LINK was specified for the type of specified incoming job for the subsystem and CONNECTION-SCOPE=*TASK / *PROGRAM was specified for all subsystem entries.

SIZE = <integer 1..32767 *4Kbyte*>
Specifies the size of the requested memory space for the data area in 4K pages.

LINK-ENTRY = <u>*UNCHANGED</u>(...) / <text 1..8 without-sep>(...)
Changes the reference address used for subsystem loading. It is also possible to specify whether automatic linking of modules to form load modules (AUTOLINK) is allowed. The change takes effect as soon as the subsystem is restarted (START-SUBSYSTEM command).

LINK-ENTRY = <u>*UNCHANGED</u>(...)
The current setting is left unchanged.

AUTOLINK = <u>*UNCHANGED</u> / *ALLOWED / *FORBIDDEN
Defines whether automatic linking of modules to form load modules (AUTOLINK) is allowed.

AUTOLINK = <u>*UNCHANGED</u>
The current setting is left unchanged.

AUTOLINK = *ALLOWED
AUTOLINK is allowed.

AUTOLINK = *FORBIDDEN
AUTOLINK is not allowed.

LINK-ENTRY = <text 1..8 without-sep>(...)
The address specified here is used as the new reference address for subsystem loading.

AUTOLINK = *ALLOWED / *FORBIDDEN
Defines whether automatic linking of modules to form load modules (AUTOLINK) is allowed.

AUTOLINK = *ALLOWED
AUTOLINK is allowed.

AUTOLINK = *FORBIDDEN
AUTOLINK is not allowed.

UNRESOLVED-EXTERNALS = *UNCHANGED / *ALLOWED / *FORBIDDEN
Defines whether unresolved external references prevent subsystem startup.

UNRESOLVED-EXTERNALS = *UNCHANGED
The current setting is left unchanged.

UNRESOLVED-EXTERNALS = *ALLOWED
Unresolved external reference do not prevent subsystem startup.
This setting is intended for debugging purposes only.

UNRESOLVED-EXTERNALS = *FORBIDDEN
Unresolved external reference prevent subsystem startup.

CHECK-REFERENCES = *UNCHANGED / *YES / *NO
Defines whether DSSM is to check the status of subsystems with which there is a dependency relation. The status of these subsystems may determine whether loading or unloading of the subsystem is allowed.
The change takes effect immediately.

CHECK-REFERENCES = *UNCHANGED
The current setting is left unchanged.

CHECK-REFERENCES = *YES
DSSM checks the status of subsystems with which there is a dependency relation. Depending on the status of these subsystems, DSSM decides whether the subsystem referred to by this command can be loaded or unloaded.

CHECK-REFERENCES = *NO
If the subsystem referred to by this command has a dependency relation with another subsystem, DSSM checks whether the latter subsystem has already been loaded. If it has, the first subsystem can be loaded, even if the other one is not yet executable (the reference is held to be resolved).
The RESUME-SUBSYSTEM, STOP-SUBSYSTEM and HOLD-SUBSYSTEM commands are executed regardless of any dependency relations which may exist.

CHANGE-STATE = <u>*UNCHANGED</u> / *YES / *NO
Restricts the use of subsystem control commands, or cancels a restriction currently in force. This operand is particularly significant in relation to a malfunctioning subsystem, as it can be used to prevent loading, activation and deactivation of the subsystem for as long as it takes to eliminate the malfunction. Corrections can thus be performed without risk.

The change takes effect immediately. It is applicable only to the current session (it is not stored in the catalog referenced by the SAVE-SUBSYSTEM-CATALOG command). The next time the subsystem is started up, the operand value is set to *NO.

CHANGE-STATE = <u>*UNCHANGED</u>
The current setting is left unchanged.

CHANGE-STATE = *YES
The commands locked by an operand value of CHANGE-STATE=*NO in an earlier MODIFY-SUBSYSTEM-PARAMETER command are released for use, thus restoring full control of the subsystem.

CHANGE-STATE = *NO
Prevents loading, activation, deactivation, suspension, restarting and unlocking of the subsystem. The following commands are locked until the command MODIFY-SUBSYSTEM-PARAMETER CHANGE-STATE=*YES is next entered:

– START-SUBSYSTEM
– RESUME-SUBSYSTEM
– STOP-SUBSYSTEM
– HOLD-SUBSYSTEM
– REMOVE-SUBSYSTEM
– UNLOCK-SUBSYSTEM

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors Guaranteed message: ESM0254
	0	CMD0001	Command executed with warnings Guaranteed message: ESM0647
1	0	CMD0001	Command executed with warnings (no logical ID found or message file (de)activated)
	1	ESM0414	Syntax error: an invalid version was specified
	1	ESM0415	Syntax error: an invalid INSTALLATION-UNIT name was specified
	1	ESM0653	Syntax error: an invalid job entry point was specified
	32	ESM0646	Internal DSSM problem during processing
	32	ESM0646	Internal error
	64	ESM0201	Subsystem not found
	64	ESM0269	Subsystem without SVC or ISL entry
	64	ESM0280	Command not executed in order to avoid inconsistencies in the subsystem catalog
	64	ESM0340	Reserved address-space exhausted for subsystems with MEMORY-CLASS=*BY-SLICE
	64	ESM0613	Changes to the message or syntax file or the installation user ID not allowed. The subsystem must first be removed by means of STOP-SUBSYSTEM.
	64	ESM0617	Memory class changed; all parameters must be specified

Notes

- Message `ESM0647` is output if attributes that it already had are entered for the subsystem. The message indicates successful changes. This change message output can be avoided by using default values instead of existing ones (generally default value *UNCHANGED).
- The CREATION-TIME operand is provided for changing subsystem parameters that were added via the ADD SUBSYSTEM command.
- If file names are specified without a user ID in the operands LIBRARY, MESSAGE-FILE, SYNTAX-FILE, REP-FILE and SUBSYSTEM-INFO-FILE, the subsystem's installation user ID is searched for the files.
- Changes to the operands INIT-ROUTINE, CLOSE-CTRL-ROUTINE, STOPCOM-ROUTINE and DEINIT-ROUTINE are accepted only if they do not impair the functioning of the subsystem.
- If a change cannot be implemented, a message to this effect is sent to SYSOUT. Messages relating to accepted changes are written to the CONSLOG file.

Example

The ARCHIVE subsystem, Version 2.8, is to be started automatically as soon as the first SVC call is issued:

```
/MODIFY-SUBSYSTEM-PARAMETER SUBSYSTEM-NAME=ARCHIVE,VERSION='02.8', -
          CREATION-TIME=*AT-SUBSYSTEM-CALL
```

RELEASE-SUBSYSTEM-SPACE Release reserved address space for subsystems

Domain:	SYSTEM-MANAGEMENT
Privileges:	STD-PROCESSING HARDWARE-MAINTENANCE SUBSYSTEM-MANAGEMENT

Function

The RELEASE-SUBSYSTEM-SPACE command enables the user to specify a subsystem group for which space in class 5 memory has been reserved by means of SCOPE=*GLOBAL and deselect it for the duration of the task. In other words, the address space reserved for this group is released and can be used for other purposes.

Format

RELEASE**-SUBSYS**TEM**-SPACE**

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	32	ESM0423	Problem with storage space management
	32	ESM0424	Internal DSSM error

REMOVE-SUBSYSTEM Remove inactive subsystem from dynamic catalog

Domain: SYSTEM-MANAGEMENT

Privileges: SUBSYSTEM-MANAGEMENT

Function

Using this command, system administration can remove an inactive subsystem from the current dynamic subsystem catalog during the current session. Removal is logical only, which means that the number of subsystems and CALL entries that can be added to the current subsystem catalog by means of the ADD-SUBSYSTEM command after this command (a maximum of 1000 subsystems and 16000 CALL entries) is not affected.

The REMOVE-SUBSYSTEM command is rejected if:

– the subsystem to be removed is active
– cross-references or dependency relations with regard to another subsystem exist

Format

```
REMOVE-SUBSYSTEM

 SUBSYSTEM-NAME = <structured-name 1..8>
,VERSION = <product-version mandatory-man-corr> / <product-version without-man-corr>
```

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem to be removed.

VERSION = <product-version mandatory-man-corr> / <product-version without-man-corr>
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type “product-version” on page 13).

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	1	ESM0414	Syntax error: an invalid version was specified
	32	ESM0646	Command processing terminated abnormally; an error occurred when updating the preliminary routine or calculating the standard version.
	64	ESM0642	Command not executed

Notes

- As soon as the subsystem has been removed, its references to and dependency relations with other subsystems cease to exist.
- If the newest version of a nonprivileged subsystem is removed, all references and dependency relations affecting the subsystem are no longer applicable.
- As soon as one subsystem has been removed using REMOVE-SUBSYSTEM, dynamic extension of the current subsystem catalog by means of the ADD-SUBSYSTEM command with the operand TYPE=*EXTENDED-ACTIVE-CONFIGURATION is no longer possible. However, dynamic extension is possible if TYPE=*NEW-SUBSYSTEMS is specified in the ADD-SUBSYSTEM command.

Example

The subsystem DAB Version 6.0 is to be removed:

```
/REMOVE-SUBSYSTEM SUBSYSTEM-NAME=DAB,VERSION='06.0'
```

RESUME-SUBSYSTEM Cancel wait state for subsystem

Domain:	SYSTEM-MANAGEMENT
Privileges:	OPERATING SUBSYSTEM-MANAGEMENT
Routing code:	R

Function

By means of the RESUME-SUBSYSTEM command system administration can cancel the wait state for any given subsystem.
Once the command has been executed, it is once again possible to set up connections to the specified subsystem, provided the subsystem was previously placed in a defined wait state by means of a HOLD-SUBSYSTEM command. This ensures that all necessary resources (holder task, address space) are still available and that the initialization routine is executable.

Format

```
RESUME-SUBSYSTEM

 SUBSYSTEM-NAME = <structured-name 1..8>
,VERSION = *STD / <product-version mandatory-man-corr> / <product-version without-man-corr> / *HIGHEST
,SUBSYSTEM-PARAMETER = *NONE / <c-string 1..254>
,RESET = *NO / *YES
,SYNCHRONOUS = *NO / *YES
```

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem whose wait state is to be canceled.

VERSION = *STD / <product-version mandatory-man-corr> / <product-version without-man-corr> / *HIGHEST
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type “product-version” on page 13).

VERSION = *STD
If there is only **one** version of the subsystem in the wait state, the default value for this version applies.
If there are **two or more** versions in the wait state, the appropriate version has to be specified.

VERSION = *HIGHEST
Selects the highest version of the subsystem.

SUBSYSTEM-PARAMETER = *NONE / <c-string 1..254>
Specifies whether special parameters are to be processed, which can be evaluated only by the specified subsystem.

RESET =
Influences the mode and urgency of command processing.

RESET = *NO
If the relevant subsystem is not yet in a defined wait state, the command is rejected until this is the case.

RESET = *YES
The command is accepted regardless of any outstanding deactivation process and the subsystem or selected components is/are initialized immediately (see “notes” below).

SYNCHRONOUS =
Allows you to choose between synchronous and asynchronous processing.

SYNCHRONOUS = *NO
The command is to be processed asynchronously, i.e. it is not necessary to wait for command execution. Error messages concerning execution of the command are not output at the console.

SYNCHRONOUS = *YES
The command must first be executed before another entry can be made.
Appropriate error messages concerning execution are output to the task.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
1	0	CMD0001	No action necessary; subsystem already in wait state
	1	ESM0414	Syntax error: an invalid version was specified
	32	ESM0224	Command will not be processed
	32	ESM0228	Command terminated abnormally

Notes

- In order to ensure a high degree of parallelism and data integrity, time-consuming administration tasks are not executed under control of the calling task; instead they are transferred to a DSSM task.
  As a rule, only the check of the requested function is carried out **synchronously** (i.e. contingent upon a wait state for the calling task). DSSM carries out the actual processing **asynchronously** and independent of the calling task.
- A RESUME-SUBSYSTEM command after a HOLD-SUBSYSTEM command will be rejected if DSSM has not yet completed the “hold subsystem” operation. However, by specifying RESET=*YES system administration can enforce the unconditional cancelation of the wait state for the subsystem; it is not necessary to wait until the HOLD-SUBSYSTEM command has been completely processed.
  In this case the initialization routine is started; the relevant subsystem is informed of the RESET and can define the scope of the initialization routine itself (complete initialization, partial initialization, no initialization).

  If two versions of a subsystem are to be exchanged, the following points should be borne in mind when using the operand RESET=*YES:

  – if version A has the status IN-DELETE and version B has the status CREATED, RESET=*YES can only be specified for A if coexistence was stipulated in the definition with SSCM of both versions (see page 47)

  – if both versions have the status IN-DELETE, RESET=*YES is permissible for one of these versions if it was defined with RESET=*ALLOWED, VERSION-EXCHANGE=*ALLOWED.

SAVE-SUBSYSTEM-CATALOG
Save changes to dynamic subsystem catalog

Domain:	SYSTEM-MANAGEMENT
Privileges:	SUBSYSTEM-MANAGEMENT

Function

With the aid of this command, users with the SUBSYSTEM-MANAGEMENT privilege can take changes made to the dynamic subsystem catalog and save them to a static subsystem catalog.

Changes made by means of the ADD-SUBSYSTEM, REMOVE-SUBSYSTEM or MODIFY-SUBSYSTEM-PARAMETER command always refer to the dynamic subsystem catalog, not the static subsystem catalog.

The changes can be saved to a static catalog with the SAVE-SUBSYSTEM-CATALOG command. They are then also effective at the next startup.
You must however note that some changes may be pointless or even unfavorable at the next startup (e.g. if a message file is assigned to a subsystem with the start attribute BEFORE-DSSM-LOAD[1]).

Format

SAVE-SUBSYSTEM-CATALOG
CATALOG-NAME = *STD / *STARTUP-CATALOG / <filename 1..54 without-gen-vers>
,FORCED = *NO / *YES

[1] The BEFORE-DSSM-LOAD attribute can only be changed via SSCM, see chapter “SSCM” on page 179.

Operands

CATALOG-NAME = *STD / *STARTUP-CATALOG / <filename 1..54 without-gen-vers>
Defines the name of the file to which the dynamic catalog is to be saved.

CATALOG-NAME = *STD
The dynamic catalog is saved under the default file name $.SYS.SSD.CAT.X.

CATALOG-NAME = *STARTUP-CATALOG
The dynamic catalog is saved under the name of the catalog used at startup.

CATALOG-NAME = <filename 1..54 without gen-vers>
The file name specified here is defined for the static catalog.

FORCED = *NO / *YES
Defines whether, despite errors, the dynamic catalog is saved to the static catalog.

FORCED = *NO
The errored dynamic catalog is not saved to the static catalog.

FORCED = *YES
The dynamic catalog is saved to the static catalog even though errors were detected in it.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	32	ESM0288	DSSM bourse not available
	32	ESM0296	Request for memory space (REQM) not executed
	32	ESM0350	Internal DSSM error; DSSM task is started again
	32	ESM0360	Insufficient memory space
	32	ESM0409	DSSM not initialized
	32	ESM0643	Internal error during save operation
	64	ESM0648	Command not executed

Notes

- The dynamic catalog to be saved may be inconsistent for a number of reasons. For example, the catalog required by DSSM may have been saved using the operand FORCED=*YES, in which case there will be inconsistencies between the subsystems. Another possibility is that changes made by means of /MODIFY-SUBSYSTEM-PARAMETER will be inacceptable at the next BS2000 startup, although they will be accepted in the current BS2000 session. Because of the danger of such inconsistencies, the catalog must first be subjected to a variety of checks before it can be saved. Any errors detected in the course of these checks are reported, and a corresponding message is output via SYSOUT.
- Even if the dynamic catalog was saved without inconsistencies being detected, it cannot be taken for granted that the next startup carried out with this catalog will be successful. For example, if the start time (CREATION-TIME) of a subsystem has been changed by means of an earlier /MODIFY-SUBSYSTEM-PARAMETER so that it is no longer started automatically during BS2000 system startup, this may lead to serious problems for other subsystems.
- If a DMS error relating to the catalog file occurs during saving of the catalog, message `ESM1806` is output. The result of the save operation must be checked accordingly. If the same message is output in relation to one of the subsystems involved, it should merely be interpreted as a warning; it has no influence on the result of /SAVE-SUBSYSTEM-CATALOG.
- If the specified catalog name is the same as the name of an existing file, a message is displayed inquiring whether the user wishes to overwrite this file.
- If certain functions cannot be correctly processed, appropriate messages are output via SYSOUT.

Example

If the dynamic catalog does not contain any errors, it is to be saved as a static catalog under the file name `NEW.STATIC.CAT`:

```
/SAVE-SUBSYSTEM-CATALOG CATALOG-NAME=COPY.DSSMCAT,FORCED=*NO
CHECK REPORT:
**** NO ERROR ****
CHECK OF LINK REFERENCES:
VERSION RANGE CHECK:
**** NO ERROR ****
LINK RELATION CHECK:
**** NO ERROR ****
CHECK OF FUNCTIONAL DEPENDENCE:
VERSION RANGE CHECK:
**** NO ERROR ****
DEPENDENCE RELATION CHECK:
**** NO ERROR ****
CYCLE CHECK:
**** NO ERROR ****
CHECK OF RELATED FILES:
********************************************************************
*    2 * SUBSYSTEM NAME:      ACS      VERSION:    14.0            *
********************************************************************
**** NO ERROR ****
********************************************************************
*    3 * SUBSYSTEM NAME:     ADAM      VERSION: 14.0A00            *
********************************************************************
**** NO ERROR ****
 .
 .
 .
 .
********************************************************************
*  176 * SUBSYSTEM NAME: CRTEBASR      VERSION:    01.3            *
********************************************************************
**** NO ERROR ****
%  ESM1200 CATALOG ':CAM1:$TSOS.COPY.DSSMCAT' GENERATED
%  ESM0254 COMMAND 'SAVE-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
```

SET-DSSM-OPTIONS Activate/deactivate DSSM logging function

Domain:	SYSTEM-MANAGEMENT
Privileges:	OPERATING SUBSYSTEM-MANAGEMENT
Routing code:	R

Function

This command can be used to control the DSSM logging function. Logging to the DSSMLOG file has a negative effect on performance. For this reason, this function should only be used when errors actually occur.

The command can be issued regardless of the status of subsystem management. By default, logging is deactivated at system startup (default value LOGGING=*OFF), but can be activated by means of the startup parameter LOGGING=*ON (see page 64).

Format

SET-DSSM-OPTIONS
LOGGING = ***OFF** / ***ON**
,**TITLE** = ***NONE** / <c-string 1..100>

Operands

LOGGING =
Determines whether DSSM-specific logging is to be carried out for error diagnosis.

LOGGING = *OFF
No DSSM-specific logging takes place.

LOGGING = *ON
All DSSM-specific data relevant for error diagnosis is logged in the file `DSSMLOG.<date>.<time>`

TITLE =
Defines a header line which appears in the logging file.

TITLE = *NONE
No additional header text is to be included in the logging file.

TITLE = <c-string 1..100>
The specified text is the first record to be written to the logging file.
If the logging file is already open, the text is added at the current position, i.e. no new file is created.
This operand is ignored if the logging function is deactivated.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	32	ESM0432	Command not executed

SHOW-SUBSYSTEM-ATTRIBUTES Request information on subsystem attributes

Domain:	SYSTEM-MANAGEMENT
Privileges:	STD-PROCESSING SUBSYSTEM-MANAGEMENT

Function

The command informs the user about the attributes of global and local subsystems.

The following table shows what information on which subsystem type (global or local) is output as a function of the user privileges.

Privilege	**Information**	**Subsystem type**
STD-PROCESSING	Attributes of nonprivileged subsystems	Local and global in class 5 memory
SUBSYSTEM-MANAGEMENT	Attributes of all subsystems	Global

The command supports output in S variables, see also the manual “Commands Volume 6” [20] and page 128.

Format

```
SHOW-SUBSYSTEM-ATTRIBUTES

 SUBSYSTEM-NAME = *ALL / <structured-name 1..8>
,VERSION = *ALL / <product-version mandatory-man-corr> / <product-version without-man-corr>
,INFORMATION = *MINIMUM / *ALL-ATTRIBUTES / *PARAMETERS(...)
    *PARAMETERS(...)
      |  GENERAL-ATTRIBUTES = *NO / *YES
      | ,INTERNAL-ENTRIES = *NO / *YES
      | ,MEMORY-ATTRIBUTES = *NO / *YES
      | ,RELATED-FILES = *NO / *YES
      | ,LINK-ATTRIBUTES = *NO / *YES
      | ,REFERENCE-RELATION = *NO / *YES
      | ,DEPENDENCE-RELATION = *NO / *YES
      | ,HOLDER-TASK-INFO = *NO / *YES
      | ,SUBSYSTEM-ENTRIES = *NO / *YES
,OUTPUT = *SYSOUT / *SYSLST(...) / *NONE
    *SYSLST(...)
      |  SYSLST-NUMBER = *STD / <integer 1..99>
      | ,LINES-PER-PAGE = <integer 1..99>
```

Operands

SUBSYSTEM-NAME = *ALL / <structured-name 1..8>
Specifies the subsystems about which the information is desired.

SUBSYSTEM-NAME = *ALL
Information is to be requested on all subsystems which are listed in the catalog (depending on privilege).

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem about which SSCM provides information from the catalog.

VERSION = *ALL / <product-version mandatory-man-corr> / <product-version without-man-corr>
Specifies the version of the selected subsystem.

VERSION = _*ALL_
Information on all versions of the subsystem stored in the catalog should be included in the output. The output scope is dependent on the privileges possessed by the user.

VERSION = <product-version mandatory-man-corr> / <product-version without-man-corr>
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type “product-version” on page 13).

INFORMATION = _*MINIMUM_ / *ALL-ATTRIBUTES / *PARAMETERS(...)
Declares the extent of the information output.

INFORMATION = _*MINIMUM_
Only the name, version and status of the subsystem are output.

INFORMATION = *ALL-ATTRIBUTES
All the information on the subsystem is output.

INFORMATION = *PARAMETERS(...)
Specifies which information is required.

GENERAL-ATTRIBUTES = _*NO_ / *YES
Specifies whether the following attributes of the named subsystems should be read from the catalog (*YES) or not (*NO):

– When should the subsystem be started after system initialization? (CREATION-TIME)
– In which load mode should the subsystem be loaded? (SUBSYSTEM-LOAD-MODE)
– Should the subsystem be automatically unloaded on shutdown? (STOP-AT-SHUTDOWN)
– Can the loaded subsystem be stopped or unloaded? (SUBSYSTEM-HOLD)
– Can the subsystem control commands be used? (STATE-CHANGE-CMDS)
– Is the FORCE option permitted? (FORCED-STATE-CHANGE)
– Is the RESET option permitted? (RESET)
– Does the initialization routine have to be repeated if the holder task is terminated abnormally? (RESTART-REQUIRED)
– Can more than one version of the subsystem be active simultaneously? (VERSION-COEXISTENCE)
– Can two versions of a subsystem be dynamically exchanged? (VERSION-EXCHANGE)
– What is the name of the subsystem’s INSTALLATION-UNIT? (INSTALLATION-UNIT)

– What is the text of the copyright (text and date) of the subsystem? (COPYRIGHT)

INTERNAL-ENTRIES = <u>*NO</u> / *YES
Specifies whether the following information on the entry points of the specified subsystems should be provided by SSCM (*YES) or not (*NO):

– the names of the entry points for the subsystem routines INIT, STOPCOM, DEINIT, and CLOSE-CRTL
– the name of the entry point which is used for dynamic identity checks (DYNAMIC-CHECK-ENTRY)
– the name of the interface version used to call the routines INIT, STOPCOM, DEINIT, or CLOSE-CTRL (INTERFACE-VERSION)

MEMORY-ATTRIBUTES = <u>*NO</u> / *YES
Specifies whether the following memory-related information, which is stored for each subsystem in the catalog, should be output (*YES) or not (*NO):

Specifies whether the subsystem memory attributes should be output. The memory attributes are:

– memory class (MEMORY-CLASS)
– size of the required address space (SIZE)
– start address of the subsystem code (START-ADDRESS)
– privileges and access authorizations relating to the address space (SUBSYSTEM-ACCESS)

RELATED-FILES = <u>*NO</u> / *YES
Specifies whether information on related components of the subsystem is to be supplied (*YES) or not (*NO). The output also specifies whether the use of a REP file for the subsystem is mandatory (REP-FILE-MANDATORY) as well as the user ID under which the related components are cataloged (INSTALLATION-USERID).

The term "related components" comprises:
– the subsystem’s object module file (LIBRARY)
– the message file (MESSAGE-FILE)
– the syntax file (SYNTAX-FILE)
– the subsystem’s information file (SUBSYSTEM-INFO-FILE)
– the REP file (REP-FILE)

LINK-ATTRIBUTES = <u>*NO</u> / *YES
Specifies whether stored information relating to the binding and loading of the subsystem is to be read from the catalog (*YES) or not (*NO):

- the name of the object module required for loading /ENTRY/CSECT (LINK-ENTRY)
- integration of the function (AUTOLINK)
- information concerning behavior in the event of unresolved external references (UNRESOLVED)
- Integration of the check run for reference subsystems (CHECK-REFERENCE)

REFERENCE-RELATION = <u>*NO</u> / *YES
Specifies whether the list of subsystems to which there are address references should be taken into account on the output of catalog information (*YES) or not (*NO).

DEPENDENCE-RELATION = <u>*NO</u> / *YES
Specifies whether the list of subsystems with which there are dependence relations should be taken into account on the output of catalog information (*YES) or not (*NO).

HOLDER-TASK-INFO = <u>*NO</u> / *YES
Specifies whether the identification of the holder task and the list of subsystems which are to be located in a shared holder task should be taken into account on the output of catalog information (*YES) or not (*NO)

SUBSYSTEM-ENTRIES = <u>*NO</u> / *YES
Specifies whether the list of incoming jobs specified on subsystem definition, together with the associated attributes listed below, should be read from the catalog (*YES) or not (*NO):

- type of specified incoming job (MODE)
- number of the routine (for *SVC or *SYSTEM-EXIT) (NUMBER)
- the function number of the entry point (FUNCTION-NUMBER)
- the version of the function number (FUNCTION-VERSION)
- information about calls to system exit routines (CALL-BY-SYSTEM-EXIT)
- privileges and access authorizations relating to entry points (CONNECTION-ACCESS and CONNECTION-SCOPE)

OUTPUT = <u>*SYSOUT</u> / *SYSLST(...) / *NONE
Specifies the system file to which the information should be output.

OUTPUT = <u>*SYSOUT</u>
The information is output to SYSOUT.

OUTPUT = *SYSLST(...)
The information is output to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Specifies the number of the SYSLST file to which the information is to be output.

LINES-PER-PAGE = <integer 1..99>
Specifies the number of lines in a SYSLST page.

OUTPUT = *NONE
No output is sent to either SYSOUT or SYSLST. Only S variables are generated (see below). For more information on S variables, refer to the manual “Commands Volume 6” [20].

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	ESM0414	Syntax error: invalid version specified
	32	CMD2009	System error: on writing S variable area
	32	ESM0360	System error: XVT or TCB unavailable
	32	ESM0602	System error: memory management problems
	32	ESM0611	System error: no connection tables for this task
	32	ESM0671	System error: on writing to SYSOUT
	64	ESM0600	Operand error: no version with *ALL
	64	ESM0601	Specified subsystem not found
	64	ESM0608	Subsystem version not found
	64	OPS0002	Command interrupted
	130	OPS0001	Command not executed because of insufficient memory space. Repeat command later

Output in S variables

The information output in the S variables corresponds to the output to SYSOUT or SYSLST (see the description of operands above). In the following table, the S variables are gathered together in groups which correspond to their assignment to the suboperands of INFORMATION=*PARAMETERS(...).

Command notation	**Abbreviated notation in table**
INFORMATION=*PARAMETERS(GENERAL-ATTRIBUTES=*NO/*YES)	GEN-ATT
INFORMATION=*PARAMETERS(INTERNAL-ENTRIES=*NO/*YES)	INT-ENT
INFORMATION=*PARAMETERS(MEMORY-ATTRIBUTES=*NO/*YES)	MEM-ATT
INFORMATION=*PARAMETERS(RELATED-FILES=*NO/*YES)	REL-FIL
INFORMATION=*PARAMETERS(LINK-ATTRIBUTES=*NO/*YES)	LINK-ATT
INFORMATION=*PARAMETERS(REFERENCE-RELATION=*NO/*YES)	REF-REL
INFORMATION=*PARAMETERS(DEPENDENCE-RELATION=*NO/*YES)	DEP-REL
INFORMATION=*PARAMETERS(HOLDER-TASK-INFO=*NO/*YES)	HOL-TASK
INFORMATION=*PARAMETERS(SUBSYSTEM-ENTRIES=*NO/*YES)	SUB-ENT

(part 1 of 6)

Output information	**Name of the S variable**	**T**	**Contents**	**Condition**
Integration of the Autolink function	var(*LIST).AUTOLINK	S	*ALLOW *FORBID	LINK-ENT
Integration of the check run for reference subsystems	var(*LIST).CHECK-REF	S	*YES *NO	LINK-ENT
Address of the name of the entry point for the subsystem routine CLOSE-CTRL-ROUTINE (if CRE)	var(*LIST).CLOSE-CTRL-ROUT.ADDR	S	<text 1..8>	INT-ENT
Name of the entry point for the subsystem routine CLOSE-CTRL-ROUTINE	var(*LIST).CLOSE-CTRL-ROUT.NAME	S	<text 1..8> *NO *DYN	INT-ENT
Copyright (text and date) of the subsystem	var(*LIST).COPYRIGHT	S	<string 1..54> *NONE	GEN-ATT

(part 2 of 6)

Output information	Name of the S variable	T	Contents	Condition
Start time of the subsystem following system initialization	var(*LIST).CRE-TIME	S	*BEFORE-DSSM-LOAD *AT-DSSM-LOAD *MANDATORY-AT-STARTUP *BEFORE-SYS-READY *AFTER-SYS-READY *AT-CRE-REQ *AT-SUBSYS-CALL	GEN-ATT
Name of the subsystem	var(*LIST).DATA(*LIST).SUBSYS-NAME	S	<structured-name 1..8>	
Version of the subsystem	var(*LIST).DATA(*LIST).SUBSYS-VERSION	S	<product-version>	
Address of the name of the entry point for the subsystem routine DEINIT (if CRE)	var(*LIST).DEINIT-ROUT.ADDR	S	<text 1..8>	INT-ENT
Name of the entry point for the subsystem routine DEINIT	var(*LIST).DEINIT-ROUT.NAME	S	<text 1..8> *NO *DYN	INT-ENT
Address of the name of the entry point used for dynamic identity checks (if CRE)	var(*LIST).DYN-CHECK-ENTRY-NAME.ADDR	S	<text 1..8>	INT-ENT
Name of the entry point used for dynamic identity checks	var(*LIST).DYN-CHECK-ENTRY-NAME.NAME	S	<text 1..8> *NO	INT-ENT
Permission to use FORCE option	var(*LIST).FORCED-STATE-CHA	S	*ALLOW *FORBID	GEN-ATT
Subsystem name in the shared holder task	var(*LIST).HOLDER-TASK.SHARED-WITH-SUBSYS(*LIST).SUBSYS-NAME	S	<structured-name 1..8> *WORK-TASK	HOL-TASK
Product version of the subsystem in the shared holder task	var(*LIST).HOLDER-TASK.SHARED-WITH-SUBSYS(*LIST).SUBSYS-VERSION	S	<product-version>	HOL-TASK
TID of the holder task (if CREATED)	var(*LIST).HOLDER-TASK.TID	S	<text 8>	HOL-TASK
TSN of the holder task	var(*LIST).HOLDER-TASK.TSN	S	<text 4>	HOL-TASK
Address of the name of the entry point for the subsystem routine INIT (if CRE)	var(*LIST).INIT-ROUT.ADDR	S	<text 1..8>	INT-ENT
Name of the entry point for the subsystem routine INIT	var(*LIST).INIT-ROUT.NAME	S	<text 1..8> *NO	INT-ENT
Name of the subsystem's INSTALLATION-UNIT	var(*LIST).INSTALL-UNIT	S	<text 1..30> *NONE	GEN-ATT

(part 3 of 6)

Output information	Name of the S variable	T	Contents	Condition
User ID under which the related components are cataloged	var(*LIST).INSTALL-USERID	S	<name 1..8> *NONE *DEF	REL-FIL
Address of the name of the interface version used to call the routines INIT-/STOPCOM-/DEINIT-/CLOSE-CTRL (if CRE)	var(*LIST).INTERF-VERSION.ADDR	S	<text 1..8>	INT-ENT
Name of the interface version used to call the routines INIT-/STOPCOM-/DEINIT-/CLOSE-CTRL	var(*LIST).INTERF-VERSION.NAME	S	<text 1..8> *NO	INT-ENT
Address of the name of the object module /ENTRY/CSECT required for loading (if CRE)	var(*LIST).LINK-ENTRY.ADDR	S	<text 1..8>	LINK-ENT
Name of the object module /ENTRY/CSECT required for loading	var(*LIST).LINK-ENTRY.NAME	S	<text 1..8>	LINK-ENT
Memory class	var(*LIST).MEM-CLASS	S	*SYS-GBL *LOC-PRIVIL *LOC-UNPRIVIL *BY-SLICE	MEM-ATT
Monitoring job variable	var(*LIST).MONJV	S	*YES *NO	GEN-ATT
Default name of the message file (for *INSTALL)	var(*LIST).MSG-F.DEF-NAME	S	<filename 1..54> *NONE	REL-FIL
Logic ID of the message file (for *INSTALL)	var(*LIST).MSG-F.LOGIC-ID	S	<filename 1..30>	REL-FIL
Name of the message file	var(*LIST).MSG-F.NAME	S	<filename 1..54> *INSTALL *NO	REL-FIL
Criterion for automatic loading if CREATION-TIME=*AT-SUBS-CALL is set at start time: on the first call to an associated SVC, ISL or other interface	var(*LIST).ON-ACTION	S	*STD *ISL-CALL *ANY	GEN-ATT
Product version of the subsystem for which address references exist (*HIGH=highest product version)	var(*LIST).REF-SUBSYS(*LIST). HIGH-VERSION	S	<product-version> *HIGH	REF-REL
Product version of the subsystem for which address references exist (*LOW=lowest product version)	var(*LIST).REF-SUBSYS(*LIST). LOW-VERSION	S	<product-version> *LOW	REF-REL

(part 4 of 6)

Output information	Name of the S variable	T	Contents	Condition
Name of the subsystem for which address references exist	var(*LIST).REF-SUBSYS(*LIST). SUBSYS-NAME	S	<structured-name 1..8>	REF-REL
Product version of the subsystem for which dependence relations exist (*HIGH=highest product version)	var(*LIST).RELATED-SUBSYS(*LIST). HIGH-VERSION	S	<product-version> *HIGH	DEP-REL
Product version of the subsystem for which dependence relations exist (*LOW=lowest product version)	var(*LIST).RELATED-SUBSYS(*LIST). LOW-VERSION	S	<product-version> *LOW	DEP-REL
Name of the subsystem for which dependence relations exist	var(*LIST).RELATED-SUBSYS(*LIST). SUBSYS-NAME	S	<structured-name 1..8>	DEP-REL
Default name of the REP file (for *INSTALL)	var(*LIST).REP-F.DEF-NAME	S	<filename 1..54> *NONE	REL-FIL
Logic ID of the REP file (for *INSTALL)	var(*LIST).REP-F.LOGIC-ID	S	<filename 1..30>	REL-FIL
Use of a REP file is mandatory for this subsystem	var(*LIST).REP-F.MANDATORY	S	*YES *NO	REL-FIL
Name of the REP file	var(*LIST).REP-F.NAME	S	<filename 1..54> *INSTALL *NO	REL-FIL
Permit RESET option	var(*LIST).RESET	S	*ALLOW *FORBID	GEN-ATT
Repeat initialization routine on abnormal termination of the holder task	var(*LIST).RESTART-REQ	S	*YES *NO	GEN-ATT
Size of the required address space (for *LOC-PRIVIL and *BY-SLICE)	var(*LIST).SIZE	I	<integer 1..32767>	MEM-ATT
Start address of subsystem code (for *LOC-UNPRIVIL)	var(*LIST).START-ADDR	S	<text 1..8>	MEM-ATT
Use of subsystem control commands	var(*LIST).STATE-CHA-CMDS	S	*ALLOW *FORBID *BY-ADM-ONLY	GEN-ATT
Should the subsystem be automatically unloaded on shutdown?	var(*LIST).STOP-AT-SHUTDOWN	S	*YES *NO	GEN-ATT
Address of the name of the entry point for the subsystem routine STOPCOM (if CRE)	var(*LIST).STOPCOM-ROUT.ADDR	S	<text 1..8>	INT-ENT
Name of the entry point for the subsystem routine STOPCOM	var(*LIST).STOPCOM-ROUT.NAME	S	<text 1..8> *NO *DYN	INT-ENT

(part 5 of 6)

Output information	Name of the S variable	T	Contents	Condition
Privileges and access authorizations relating to the address space (for *SYS-GBL and *LOC-UNPRIVIL)	var(*LIST).SUBSYS-ACCESS	S	*LOW *SYS *HIGH	MEM-ATT
Addresses of the entry points (if CREATED)	var(*LIST).SUBSYS-ENTRIES(*LIST).ADDR	S	<text 1..8>	SUB-ENT
Privileges relating to the entry points	var(*LIST).SUBSYS-ENTRIES(*LIST).CONN-ACCESS	S	*ALL *SYS *SIH	SUB-ENT
Access authorizations relating to the entry points	var(*LIST).SUBSYS-ENTRIES(*LIST).CONN-SCOPE	S	*TASK *PROG *FREE *CALL *OPTIM	SUB-ENT
Permit entry points on initial connection	var(*LIST).SUBSYS-ENTRIES(*LIST).FIRST-CONN	S	*ALLOW *FORBID	SUB-ENT
Function number of entry point (for ISL or SVC)	var(*LIST).SUBSYS-ENTRIES(*LIST).FUNC-NUM	I	<integer 0..255>	SUB-ENT
Version of the function number (for ISL or SVC)	var(*LIST).SUBSYS-ENTRIES(*LIST).FUNC-VERSION	I	<integer 1..255>	SUB-ENT
Type of specified incoming job	var(*LIST).SUBSYS-ENTRIES(*LIST).MODE	S	*LINK *ISL *SVC *SYS-EXIT	SUB-ENT
Name of entry point	var(*LIST).SUBSYS-ENTRIES(*LIST).NAME	S	<text 1..8>	SUB-ENT
Stop or unload the loaded subsystem	var(*LIST).SUBSYS-HOLD	S	*ALLOW *FORBID	GEN-ATT
Default name of the subsystem’s information file (for *INSTALL)	var(*LIST).SUBSYS-INFO-F.DEF-NAME	S	<filename 1..54> *NONE	REL-FIL
Logic ID of the subsystem’s information file (for *INSTALL)	var(*LIST).SUBSYS-INFO-F.LOGIC-ID	S	<filename 1..30>	REL-FIL
Name of the subsystem’s information file	var(*LIST).SUBSYS-INFO-F.NAME	S	<filename 1..54> *INSTALL *NO	REL-FIL
Internal status	var(*LIST).SUBSYS-INT-STA	S	INSTALLED INITIALIZED CONNECTABLE WAIT-CLS-CTRL WAIT-DISCON WAIT-DEINIT WAIT-STOP-COM	GEN-ATT
Default name of the subsystem’s object module file (for *INSTALL)	var(*LIST).SUBSYS-LIB.DEF-NAME	S	<filename 1..54>	REL-FIL

(part 6 of 6)

Output information	Name of the S variable	T	Contents	Condition
Logic ID of the subsystem's object module file (for *INSTALL)	var(*LIST).SUBSYS-LIB.LOGIC-ID	S	<filename 1..30>	REL-FIL
Name of the subsystem's object module file	var(*LIST).SUBSYS-LIB.NAME	S	<filename 1..54> *INSTALL *CPLINK	REL-FIL
Load mode in which the subsystem is loaded	var(*LIST).SUBSYS-LOAD-MODE	S	*STD *ADV *ANY	GEN-ATT
Status of the subsystem	var(*LIST).SUBSYS-STA	S	*NOT-CRE *IN-CRE *IN-HOLD *IN-DEL *IN-RESUME *NOT-RESUMED *CRE *LOCK	GEN-ATT
Default name of the syntax file (for *INSTALL)	var(*LIST).SYNTAX-F.DEF-NAME	S	<filename 1..54> *NONE	REL-FIL
Logic ID of the syntax file (for *INSTALL)	var(*LIST).SYNTAX-F.LOGIC-ID	S	<filename 1..30>	REL-FIL
Name of the syntax file	var(*LIST).SYNTAX-F.NAME	S	<filename 1..54> *INSTALL *NO	REL-FIL
Information on behavior in the event of unresolved external references	var(*LIST).UNRESOLVED-EXTERNAL	S	*ALLOW *FORBID	LINK-ENT
More than one version of the subsystem active simultaneously	var(*LIST).VERSION-COEXIST	S	*ALLOW *FORBID	GEN-ATT
Dynamic exchange of two versions of the subsystem	var(*LIST).VERSION-EXCHA	S	*ALLOW *FORBID	GEN-ATT
Year specification (if COPYRIGHT)	var(*LIST).YEAR	S	<string 4>	GEN-ATT

Notes

- /SHOW-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=*ALL displays the same information as /SHOW-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=*ALL, VERSION=*ALL.
- /SHOW-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=*ALL,VERSION=<version> is not supported.
- If a subsystem is found in the local subsystem catalog, the word “Subsystem” will be replaced in the output information by “LOCAL SUBSYSTEM”.
- If a user only has the *STD-PROCESSING privilege, he/she receives the attributes of the local subsystem (if present) and the attributes of the nonprivileged global subsystems (i.e. those where SUBSYSTEM-ACCESS is not *SYSTEM).
- Should an error occur when writing the information to SYSOUT or SYSLST (`ESM0671`), the S variables are written as normal.
- Should an error occur in writing the S variables (`OPS0001` or `CMD2009`), the writing of the information to SYSOUT or SYSLST will be continued normally.
- Should the interrupt key $\boxed{\text{K2}}$ be pressed during the input request (`PLEASE ACKNOWLEDGE`) the output to SYSOUT as well as the output in S variables will be interrupted immediately. Return code `OPS0002` is set.
- The complete matching of the respective outputs to SYSOUT/SYSLST can only be guaranteed if the procedure which issued the command has forbidden interruption of the output with the operand INTERRUPTION-ALLOWED=*NO.

Examples

Example 1

Output of the attributes of non-privileged subsystems
(users with the *STD-PROCESSING privilege)

```
/show-subsystem-attributes subsystem-name=*all

%********************************************************************
%*    3 * SUBSYSTEM NAME:  ASSEMBH      VERSION:    01.2            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : NOT CREATED
%********************************************************************
%*    4 * SUBSYSTEM NAME:  ASSTRAN      VERSION:    01.7            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : NOT CREATED
%********************************************************************
%*    5 * SUBSYSTEM NAME:  ASSGENH      VERSION:    01.2            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : NOT CREATED
%********************************************************************
%*    6 * SUBSYSTEM NAME:  BUILDER      VERSION:    01.0            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : CONNECTABLE
%********************************************************************
%*    7 * SUBSYSTEM NAME: CRTEBASR      VERSION:    01.3            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : NOT CREATED
%********************************************************************
%*    8 * SUBSYSTEM NAME: CRTEBASY      VERSION:    01.3            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : CONNECTABLE
%********************************************************************
 .
 .
 .
 .
%********************************************************************
%*  192 * SUBSYSTEM NAME:   MONSYS      VERSION: 03.0B20            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : NOT CREATED
%********************************************************************
%*  193 * SUBSYSTEM NAME:   VAS-TU      VERSION:    02.1            *
%********************************************************************
% STATUS OF THE SUBSYSTEM : NOT CREATED
%  ESM0608 SUBSYSTEM VERSION NOT FOUND. FUNCTION ABORTED
%  ESM0255 'SHOW-SUBSYSTEM-ATTRIBUTES' COMMAND NOT PROCESSED
```

Example 2

Output of the attributes of non-privileged subsystems
(users with the *STD-PROCESSING privilege)

```
/show-subsystem-attributes subsystem-name=assembh,information=*all-attributes

%*********************************************************************
%*    3 * SUBSYSTEM NAME:  ASSEMBH      VERSION:    01.2             *
%*********************************************************************
%GENERAL ATTRIBUTES:
% INSTALLATION-UNIT       : **** NOT SPECIFIED ****
% COPYRIGHT               : FUJITSU SIEMENS COMPUTERS GMBH
% YEAR                    : 2000
% SUBSYSTEM-LOAD-MODE     : ADVANCED
% CREATION-TIME           : AT-CREATION-REQUEST
% STOP-AT-SHUTDOWN        : NO
% SUBSYSTEM-HOLD          : ALLOWED
% STATE-CHANGE-CMDS       : ALLOWED
% FORCED-STATE-CHANGE     : FORBIDDEN
% RESET                   : FORBIDDEN
% RESTART-REQUIRED        : NO
% VERSION-COEXISTENCE     : FORBIDDEN
% VERSION-EXCHANGE        : FORBIDDEN
% STATUS OF THE SUBSYSTEM : NOT CREATED
% MONITORING JOB VARIABLE : NO
%INTERNAL ENTRIES:
% INIT-ROUTINE         : **** NOT SPECIFIED ****
% CLOSE-CTRL-ROUTINE   : **** NOT SPECIFIED ****
% STOPCOM-ROUTINE      : **** NOT SPECIFIED ****
% DEINIT-ROUTINE       : **** NOT SPECIFIED ****
% DYNAMIC-CHECK-ENTRY  : **** NOT SPECIFIED ****
% INTERFACE-VERSION    : **** NOT SPECIFIED ****
%MEMORY ATTRIBUTES:
% MEMORY-CLASS: SYSTEM GLOBAL    SUBSYSTEM-ACCESS: HIGH
%RELATED FILES:
% INSTALLATION-USERID : *DEFAULT-USERID
% LIBRARY             : $TSOS.SYSLNK.ASSEMBH.012
% MESSAGE-FILE        : **** NOT SPECIFIED ****
% SYNTAX-FILE         : **** NOT SPECIFIED ****
% SUBSYSTEM-INFO-FILE : **** NOT SPECIFIED ****
% REP-FILE            : **** NOT SPECIFIED ****
% REP-FILE-MANDATORY  : NO
%LINK ATTRIBUTES:
% LINK-ENTRY: IARC000
% AUTOLINK  : ALLOWED     UNRESOLVED: FORBIDDEN     CHECK-REFERENCE: YES
%REFERENCED SUBSYSTEMS:
%         NAME    LOWEST VERSION    HIGHEST VERSION
%         ----    --------------    ---------------
%         **** NONE ****
%FUNCTIONAL DEPENDENCE WITH SUBSYSTEMS:
%         NAME    LOWEST VERSION    HIGHEST VERSION
%         ----    --------------    ---------------
%         **** NONE ****
```

Continued ➠

```
%SHARED-HOLDER-TASK:        WITH SUBSYSTEMS:
%          NAME    VERSION           NAME    VERSION
%          ----    -------           ----    -------
%           SMI       01.0        ASSTRAN       01.7
%       ASSGENH       01.2        BUILDER       01.0
%      CRTEBASR       01.3       CRTEBASY       01.3
%       PAMCONV       12.0        COBPARR       02.3
%      CRTEPARR       02.3        DCE-RTS       01.0
%       DRIVE21       02.1          DRIVE       02.1
%        DRTS21       02.1            EDT       16.6
%        EDTCON       16.6        ESQLCOB    02.0A00
%       FOR1LZS    02.2C20         C5-MIG       01.0
%         LEASY       06.0          MAREN       08.1
%           OSS       04.0           PLI1       04.2
%       PLI1RTS       04.2        PLI1IOS       04.2
%       RPG3RTS    04.0B00         SESSQL       02.2
%         UDS-D    02.0B70        UDS-SQL    02.0B70
%       UXBASIC    03.0B20         UXCOMP    03.0B20
%         UXRUN    03.0B20        VM2-MON       07.0
%      SESKOMMD       03.0         SESDBH       03.0
 .
 .
 .
 .
 .
%      HSMS-API       06.0        HSMS-SV       06.0
%      BCAM-CMD       16.0       BCAM-COS       16.0
%      BCAM-SM2       16.0         CMX-TU       01.3
%        CMX-TP       01.3         CMX-11       01.3
%      DCAM-COS       13.1       DCM-DIAG       01.0
%        SOC-TP       02.0        SOCKETS       02.0
%          VTSU       13.1       VTSUTRAC       13.1
%      XHCS-SYS       01.5            RFA       14.0
%           SM2       14.0           TIAM       13.1
%      VTSU-X29       01.4           SORT       07.8
%        BLSSEC       14.0        FHS-TPR       08.2
%           FHS       08.2         FHS-DM       08.2
%      FHS-PRIV       08.2             JV       13.0
%      GUARDCOO       04.0       GUARDDEF       04.0
%         SATCP       04.0        SRPMOPT       04.0
%        PERCON    02.7A10         LLMAID       01.0
%         CRTEC       02.3         BINDER       02.1
%          SSCM       02.3            ACO    02.2A00
%          EDOR       08.3       EXIT#010    04.0A01
%      EXIT#015    04.0A01       EXIT#020    04.0A01
%      EXIT#051    04.0A01       EXSP0090    04.0A01
%      EXSP0091    04.0A01       EXSP0092    04.0A01
%      TASKDATE    14.0A00           TPME    04.0A01
%          DCAM       13.1       SDFPBASY    02.1C30
%         SDF-P    02.1C30
%SUBSYSTEM-ENTRIES:
%  CE-BY-PROGRAM      : NO
% NAME: IARC000
%  MODE               : LINK
%  DUMMY-ENTRY        : NO
%  CONNECTION-ACCESS  : ALL
%  CONNECTION-SCOPE   : PROGRAM
%  FIRST-CONNECTION   : ALLOWED
```

SHOW-SUBSYSTEM-INFO
Request information on current subsystems configuration

Domain:	SYSTEM-MANAGEMENT
Privileges:	SUBSYSTEM-MANAGEMENT

Function

The SHOW-SUBSYSTEM-INFO command enables system administration to request information on the current subsystems configuration. Output is exclusively via SYSLST and comprises the complete description of all known subsystems, including the following dynamic features:

– current status
– load addresses of started subsystems
– number of connections registered since startup
– size of the subsystem (size of the memory space it occupies, including subsystem code and metadata)

Format

SHOW-SUBSYSTEM-**INFO**
SUBSYSTEM-NAME = ***ALL**

Operands

SUBSYSTEM-NAME = *ALL
Output is directed to SYSLST and comprises the entire dynamic subsystem configuration.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	32	ESM0288	Problems with bourse communication
	32	ESM0298	Problems with memory management
	32	ESM0350	Internal DSSM error during processing
	32	ESM0670	Error when writing to the SYSLST file

The following abbreviations are used in the output:

for CREATION-TIME

ACR	:	*AT-CREATION-REQUEST
ASC	:	*AT-SUBSYSTEM-CALL
ADL	:	*AT-DSSM-LOAD
BDL	:	*BEFORE-DSSM-LOAD
MAS	:	*MANDATORY-AT-STARTUP
BSR	:	*BEFORE-SYSTEM-READY
ASR	:	*AFTER-SYSTEM-READY

for MEMORY-CLASS

S	:	*SYSTEM-GLOBAL
P	:	*LOCAL-PRIVILEGED
U	:	*LOCAL-UNPRIVILEGED
B	:	*BY-SLICE

for SUBSYSTEM-ACCESS

SYS	:	*SYSTEM
ALL	:	*LOW / *HIGH

for INTERNAL-ENTRIES

DYN	:	*DYNAMIC
YES	:	name
NO	:	*NO

for CONNECTION-ACCESS

SYS	:	*SYSTEM

for STATE-CHANGE-CMDS

ADM	:	*BY-ADMINISTRATOR-ONLY

for REP-FILE

MAN	:	REP-FILE = *STD / file name and REP-FILE-MANDATORY = *YES

for GENERAL-ATTRIBUTES

YES	:	*ALLOWED
NO	:	*FORBIDDEN

for RELATED-FILES

YES	:	*STD / file name
NO	:	*NO
IMO	:	*INSTALLED

Example

```
/show-subsystem-info
%  ESM0254 COMMAND 'SHOW-SUBSYSTEM-INFO' COMPLETELY PROCESSED
```

The output is sent preformatted to SYSLST (extract of output):

```
A  Fujitsu Siemens Computers GmbH 2000        DYNAMIC SUBSYSTEM MANAGEMENT           DATE: 2001-10-29   TIME: 15-30-58     PAGE:    1
?
?  CATALOG-LAYOUT       : VERSION 309                                                PRINT-LAYOUT           : VERSION 03.6
?
?  STARTUP CATALOG-NAME : :T053:$TSOS.SYS.SSD.CAT.X                                  FORCED GENERATION      : NO
?
?  DSSM BOURSE ID       : 80A1002F
?
?  CATALOG ADDRESS IN CLASS 4             : X'7FB32060'                              CATALOG ADDRESS IN CLASS 5 : X'70A00060'
?
?  ADDRESS-STRIPE FOR LOCAL-PRIVILEGED SS : X'00100000' BYTES LENGTH                 START ADDRESS              : X'00B00000'
?  ADDRESS-STRIPE FOR PRIVATE SLICES OF SS: X'00A00000' BYTES LENGTH                 START ADDRESS              : X'70500000'
?
A  Fujitsu Siemens Computers GmbH 2000        DYNAMIC SUBSYSTEM MANAGEMENT           DATE: 2001-10-29   TIME: 15-30-58     PAGE:    2
?
?                          |           GENERAL ATTRIBUTES            |  INTERNAL ENTRIES   | SUBSYST | MEMORY|  RELATED FILES  |
?                          |  C   S                                  |      C           D  | ENTRIES |       |              I  |  H
?                          |  R   H       C           C   C   X   R  |      L   S       Y  |         |       |  M   R   S   N  |  O
?                          |  E   U       O   F       H   O   C   E  |      O   T   D   N  |         |    A  |  S   E   Y   F  |  L
? NR  SS-NAME  VERSION STD |  A   T       M   O   R   K   E   H   S  |      S   O   E   C  |      S  | C  C  |  G   P   N   O  |  D
?                          |  T   D   H   M   R   E   L   X   A   T  |  I   E   P   I   H  |      Y  | L  C  |  F   F   F   F  |  E
?                          |  I   O   O   A   C   S   I   I   N   A  |  N   C   C   N   E  |  S   S  | A  E  |  I   I   I   I  |  R
?                          |  M   W   L   N   E   E   N   S   G   R  |  I   T   O   I   C  |  V   E  | S  S  |  L   L   L   L  |  I
?                          |  E   N   D   D   D   T   K   T   E   T  |  T   L   M   T   K  |  C   X  | S  S  |  E   E   E   E  |  D
?--------------------------|-----------------------------------------|---------------------|---------|-------|-----------------|-----
?   1 CP                   |                                         |                     |         |       |                 |
?   2 ACS      14.0    YES | ACR YES YES YES NO  NO  YES NO  NO  NO  | YES NO  YES YES YES | NO  NO  | S SYS | NO  IMO IMO NO  |   44
?   3 ADAM     14.0A00 YES | ASC YES YES YES NO  NO  YES NO  YES NO  | YES NO  NO  YES YES | YES NO  | S SYS | NO  IMO NO  NO  |   44
?   4 AID      03.0    YES | ACR NO  YES YES NO  NO  YES NO  NO  NO  | YES NO  NO  YES NO  | NO  NO  | S SYS | NO  IMO NO  NO  |   44
?   5 SMI      01.0    YES | ASR NO  YES YES NO  NO  YES NO  YES NO  | NO  NO  NO  NO      | NO  NO  | S SYS | NO  NO  NO  NO  |   44
?   6 LLMAID   01.0    YES | ASR NO  YES YES YES NO  YES YES YES NO  | NO  NO  NO  NO  NO  | NO  NO  | S SYS | NO  IMO NO  NO  |   44
?   7 AIDSYS   14.0    YES | ASC NO  YES YES NO  NO  YES NO  NO  NO  | NO  NO  NO  NO  NO  | NO  NO  | S SYS | NO  IMO NO  NO  |   44
?   8 AIDSYSA  14.0    YES | MAS NO  NO  NO  NO  NO  YES NO  NO  NO  | YES NO  NO  YES NO  | YES NO  | S SYS | NO  IMO NO  NO  |   44
?   9 ANITA    14.0    YES | ASC NO  YES YES NO  NO  YES NO  NO  NO  | YES NO  NO  YES     | YES NO  | S SYS | NO  NO  NO  NO  |   44
  .
  .
  .
  .
  .
? 175 UTM      05.1    YES | ACR NO  YES YES NO  NO  NO  YES YES NO  | YES NO  NO  NO  NO  | NO  NO  | S SYS | NO  IMO NO  NO  |   44
? 176 UTM-SM2  14.0    YES | ACR NO  YES YES NO  NO  NO  NO  YES NO  | YES NO  YES YES NO  | NO  NO  | S SYS | NO  IMO NO  NO  |   44
? 177 VAS-TU   02.1    YES | ACR NO  YES YES NO  NO  YES YES YES NO  | NO  NO  NO  NO  NO  | NO  NO  | P LOW | NO  YES NO  NO  |   49
? 178 VM2-MON  07.0    YES | ACR NO  NO  NO  NO  NO  NO  NO  NO  NO  | YES NO  NO  NO  YES | NO  NO  | S SYS | IMO IMO IMO NO  |   44
? 179 VOLIN    14.0    YES | ASC NO  YES NO  NO  NO  YES NO  NO  NO  | YES NO  NO  NO  YES | YES NO  | S SYS | NO  IMO NO  NO  |   44
? 180 XHCS-SYS 01.5    YES | BSR NO  NO  YES NO  NO  YES NO  NO  NO  | YES NO  NO  NO  YES | YES NO  | S SYS | NO  IMO NO  NO  |   44
? 181 VTSU-X29 01.4    YES | ACR NO  YES YES NO  NO  YES NO  NO  NO  | YES NO  YES NO  YES | NO  NO  | S SYS | NO  IMO NO  NO  |   44
? 182 VTSUTRAC 13.1    YES | ACR NO  YES YES NO  NO  YES NO  NO  NO  | YES NO  NO  NO  YES | NO  YES | S SYS | NO  IMO NO  NO  |   44
? 183 WARTOPT  14.0    YES | ASC NO  YES YES NO  NO  YES NO  NO  NO  | NO  NO  NO  NO  YES | YES NO  | S SYS | IMO IMO NO  NO  |   44
? 184 XCS-TIME 14.0A00 YES | ACR NO  YES NO  NO  NO  NO  NO  NO  YES | YES NO  NO  DYN NO  | NO  NO  | S SYS | NO  IMO NO  NO  |   43
? 185 EXPRESSO 01.0    YES | ACR NO  YES YES NO  NO  YES NO  NO  NO  | NO  NO  NO  NO  NO  | NO  YES | S SYS | NO  YES NO  NO  |   44
A  Fujitsu Siemens Computers GmbH 2000        DYNAMIC SUBSYSTEM MANAGEMENT           DATE: 2001-10-29   TIME: 15-30-58     PAGE:    6
?
?   1          SS NAME  : CP                                                             STATUS: CREATED
?
   Fujitsu Siemens Computers GmbH 2000        DYNAMIC SUBSYSTEM MANAGEMENT           DATE: 2001-10-29   TIME: 15-30-58     PAGE:    7
?
?   2          SS NAME  : ACS           VERSION: 14.0         STANDARD VERSION: YES      STATUS: NOT CREATED
?
?    LINK ENTRY NAME /@: DACSUB   / X'7FFFFFFF'         LIBRARY DEFAULT      : $TSOS.SYSLNK.ACS.140
?    INSTALLATION UNIT : ACS                            LIBRARY LOGID        : SYSLNK
?
?    INTERFACE ENTRY   : DACDSMV                        REP FILE DEFAULT     : $TSOS.SYSREP.ACS.140
?                                                       REP FILE LOGID       : SYSREP
?                                                    USED REP FILE NAME   : *NO
?
?    CHECK ID NAME   /@: DACDSMC  / X'7FFFFFFF'         REP FILE MANDATORY   : *NO                  SUBSYSTEM LOAD MODE: *ADVANCED
?
?    CREATION TIME     : AT CREATION REQUEST            MESSAGE FILE NAME    : *NO
?
?    AUTO LINK         : YES                            INFO FILE NAME       : *NO
?
?    CONTINUE (IGNORE UNRESOLVED EXTERNALS) : NO        SYNTAX FILE DEFAULT  : $TSOS.SYSSDF.ACS.140
?                                                    SYNTAX FILE LOGID    : SYSSDF
?
?     ROUTINES :                         NAME       TYPE     ADDRESS
?                  INITIALISATION   :  DACDSMI    BLS      X'7FFFFFFF'
?                  CLOSE CONTROL    :  *NO        NONE         --
?                  STOP COMMISSION  :  DACDSMS    BLS      X'7FFFFFFF'
?                  DEINITIALISATION :  DACDSMD    BLS      X'63E2D6E2'
?
?     CALL ENTRIES:  INDEX   NAME       TYPE    NUMBER  FNUMBER  FVERS  ACCESS  CONNECTION SCOPE     ADDRESS   A-MODE FIRST CONNECT
?                        1   DACSUB     ISL        -       -       -     SYS          TASK              --       -    *ALLOWED
?                        2   DACCMD     ISL        -       -       -     SYS          TASK              --       -    *ALLOWED
?                        3   DACDSMI    BLS        -       -       -     SYS          TASK              --       -    *ALLOWED
?                        4   DACDSMD    BLS        -       -       -     SYS          TASK              --       -    *ALLOWED
?                        5   DACDSMS    BLS        -       -       -     SYS          TASK              --       -    *ALLOWED
?                        6   DACDSMV    BLS        -       -       -     SYS          TASK              --       -    *ALLOWED
?                        7   DACDSMC    BLS        -       -       -     SYS          TASK              --       -    *ALLOWED
?
?     SUBSYSTEM RELATIONSHIPS :
?                  REFERENCED SUBSYSTEM(S):     1
?                  RELATED    SUBSYSTEM(S):  NONE
?                  DISJOINTED SUBSYSTEM(S):  NONE
?
?     ADDRESS SPACE :
?                  ACCESS        : SYS
?                  MEMORY CLASS  : 3/4
?                  MEMORY POOL   : NO
?                  MEMORY SIZE   :
?
?    HOLDERTASK IDENTIFICATION :   44    TSN : HT2C
 .
 .
 .
 .
 .
 .
```

SHOW-SUBSYSTEM-STATUS Request information on status of subsystems

Domain:	SYSTEM-MANAGEMENT
Privileges:	STD-PROCESSING OPERATING SUBSYSTEM-MANAGEMENT
Routing code:	R

Function

The SHOW-SUBSYSTEM-STATUS command informs the user of the status of global subsystems which are made available to all users in class 5 memory (nonprivileged subsystems) and, if a local subsystem catalog has been loaded, of the status of the user's local subsystems.
Errors which occur during execution of the SHOW-SUBSYSTEM-STATUS command do not result in spin-offs (see /SET-JOB-STEP the "Commands Volume 1-5" manual [19]).

The command SHOW-SUBSYSTEM-STATUS outputs its information not only to SYSOUT. It also supports output in S variables, see page 145. Users are then able to evaluate and process the information in their own S procedures. For detailed information on working with S variables see the manual "Commands Volume 6" [20].
The output information varies depending on the privileges of the caller (see the table on the next page).

Privileged functions (SUBSYSTEM-MANAGEMENT or OPERATING privilege):

Using the SHOW-SUBSYSTEM-STATUS command, systems support can request information on the status of global (privileged and nonprivileged) subsystems. In addition to the type (global), name, version and status of the global subsystem, the following information is displayed:

– which tasks have a connection to the specified subsystem (TSN and TID)
– the number of connections set up to a specified subsystem since startup

While the command is being processed, other jobs can set up or clear down a connection to the subsystem; consequently, the list of displayed jobs may not reflect the current status. Privileged users cannot request information on local subsystems belonging to other tasks.

Nonprivileged functions (STD-PROCESSING privilege)

The type (global or local), name, version and status of nonprivileged global or local subsystems are displayed.

A user with all three privileges is able to obtain information on all subsystems, no matter whether they are local or global, privileged or nonprivileged.
The following information is output depending on the caller's privileges:

STD-PROCESSING	STD-PROCESSING and SUBSYSTEM-MGMT	OPERATING or SUBSYSTEM-MGMT
The requested information relates to locally defined subsystems only:*		
a) all local subsystems b) if none of them is in the CREATED state, then all local NOT-CREATED subsystems, but without indication of the version	a) all local subsystems b) if none of them is in the CREATED state, then all local NOT-CREATED subsystems, but without indication of the version	no output
The requested information relates to locally defined and globally defined subsystems:*		
a) all local subsystems and all global, nonprivileged subsystems b) if all of them are in the NOT-CREATED state, then as under a), but without indication of the version	a) all local subsystems and all global subsystems with the connected tasks, provided they are in the CREATED, IN-HOLD or IN-DELETE state b) if all subsystems are in the NOT-CREATED state, then as under a), but without indication of the version	a) no local subsystems, but all global subsystems with the connected tasks, provided they are in the CREATED, IN-HOLD or IN-DELETE state b) if all of them are in the NOT-CREATED state, then as under a), but without indication of the version
The requested information relates to globally defined subsystems only:		
a) all global, nonprivileged subsystems b) if all of them are in the NOT-CREATED state, then as under a), but without indication of the version	a) all global subsystems with the connected tasks, provided they are in the CREATED, IN-HOLD or IN-DELETE state b) if all global subsystems are in the NOT-CREATED state, then as under a), but without indication of the version	a) all global subsystems with the connected tasks, provided they are in the CREATED, IN-HOLD or IN-DELETE state b) if all of them are in the NOT-CREATED state, then as under a), but without indication of the version

* Only the local subsystems whose owner is the calling user task are displayed.

Format

SHOW-SUBSYSTEM-**STA**TUS
SUBSYSTEM-NAME = ***AL**L / ***NON-PRIVIL**EGED-**CLASS-5** / <structured-name 1..8>
,**VERSION** = ***STD** / ***ALL** / <product-version mandatory-man-corr> / <product-version without-man-corr>

Operands

SUBSYSTEM-NAME =
Specifies the name of the subsystem on which information is requested.

SUBSYSTEM-NAME = *ALL
Outputs information on all subsystems.
This operand value must not be specified together with VERSION=<product-version ...>.

SUBSYSTEM-NAME = *NON-PRIVILEGED-CLASS-5
Outputs information on all global subsystems occupying nonprivileged class 5 memory pages (subsystems with the attribute MEMORY-CLASS=*LOCAL-PRIVILEGED).

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem on which information is requested.

VERSION =
Defines the version number.

VERSION = *STD
If the version is not specified of if *STD is specified explicitly, the following sequence applies:

1. The information supplied refers to the subsystem which does not have the status NOT-CREATED.
2. If there is more than one version with a status other than NOT-CREATED, all of these versions are included in the output information.

If all subsystem versions have the status NOT-CREATED, there is no version specified in the output text. The information is output both for global subsystem versions and for local subsystem versions - provided a local subsystem catalog is loaded.

VERSION = *ALL
Information is to be supplied on all available versions of the relevant subsystem.
This entry is not permitted in conjunction with SUBSYSTEM-NAME=*NON-PRIVILEGED-CLASS-5.

VERSION = <product-version mandatory-man-corr> /
<product-version without-man-corr>
Version number of this subsystem. This number must be identical to the format specified in the definition of the subsystem. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type "product-version" on page 13).

This operand value must not be specified together with SUBSYSTEM-NAME=*ALL/*NON-PRIVILEGED-CLASS-5.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	1	ESM0414	Syntax error: invalid version specified
	32	CMD2009	Error when writing the S variable area
	32	ESM0360	System error: XVT or TCB not accessible
	32	ESM0602	Problems with storage management
	32	ESM0611	No connection tables for this task
	32	ESM0671	Error when writing to SYSOUT
	64	ESM0600	No version with *ALL
	64	ESM0601	Specified subsystem not found
	64	ESM0604	No version with *NON-PRIV-CLASS-5
	64	ESM0608	Subsystem version not found
	64	ESM0610	No subsystem with *NON-PRIV-CLASS-5
	64	OPS0002	Command interrupted
	130	OPS0001	Command not executed due to lack of memory space: repeat command later

Output in S variables

If an error occurs during output to SYSOUT (message `ESM0671`), output in S variables continues normally (assuming, of course, it has been requested).
If an error occurs in the course of output in S variables (message `ESM0672` with return code `CMD2009`), output to SYSOUT continues normally.

The information given here corresponds in essence to that output via SYSOUT. If a version was specified, the S variable SUBSYS-TYPE additionally describes whether the relevant subsystem has been defined globally or locally.
If all versions of a subsystem in the subsystem catalog have the status NOT-CREATED and if output of the subsystem is requested with VERSION=*STD, only the subsystem name and the subsystem status are output.

Output information	Name of the S variable	T	Contents	Condition
Number of connected tasks since the start of the subsystem	var(*LIST).CONN-NUM-SINCE-START	I	<integer 0..2^{31}-1>	
Information on the user address space for nonprivileged subsystems *RESERVED: the subsystem is in class 5 memory *UNRESERVED: a privileged user has called the RELEASE-SUBSYSTEM-SPACE command	var(*LIST).SUBSYS-ADDR-SPACE	S	*RESERVED *UNRESERVED	
Internal status, see table 12 on page 147	var(*LIST).SUBSYS-INT-STA	S	INSTALLED INITIALIZED CONNECTABLE WAIT-CLS-CTRL WAIT-STOP-COM WAIT-DISCON WAIT-DEINIT NONE	
Name of the subsystem	var(*LIST).SUBSYS-NAME	S	<structured-name 1..8>	
Status of the subsystem	var(*LIST).SUBSYS-STA	S	*CREATED *IN-CREATE *IN-DELETE *IN-HOLD *IN-RESUME *LOCKED *NOT-CREATED *NOT-RESUMED	
Is the relevant subsystem defined locally or globally? Only evaluated if a value is assigned to the VERSION operand.	var(*LIST).SUBSYS-TYPE	S	*GLB *LOC	
Version of the subsystem	var(*LIST)..SUBSYS-VERSION	S	<product-version>	
TID of the task which is currently connected to the subsystem	var(*LIST).USED-TASK-LIST(*LIST).TID	S	<text 8>	
TSN of the task which is currently connected to the subsystem	var(*LIST).USED-TASK-LIST(*LIST).TSN	S	<text 4>	

For details of working with S variables refer to Manual “Commands Volume 6” [20] (basic principles, structure, declaration, assignment, access and further use).

For an example with output in S variables see page 173ff.

The contents of the S variable var(*LIST).SUBSYS-INT-STA is dependent on the status of the subsystem (S variable var(*LIST).SUBSYS-STA):

Status / var(*LIST).SUBSYS-STA	**Sub-Status var(*LIST).SUBSYS-INT-STA**	**Meaning**
IN-CREATE/ IN-RESUME	INSTALLED INITIALIZED CONNECTABLE	The subsystem is loaded. The 'Init' routine has been executed. The subsystem is ready.
IN-DELETE / IN-HOLD	WAIT-CLS-CTRL WAIT-DISCON WAIT-STOP-COM WAIT-DEINIT INSTALLED	The 'Close Control' routine has been called. DSSM is waiting for it to terminate. DSSM is waiting for the last disconnection (forced cancellation possible). No further task is connected and DSSM is waiting for the 'Stop Commission' routine to terminate. No further task is connected, the 'Deinit' routine has been called and DSSM is waiting for it to terminate. The subsystem has been loaded but not initialized.
NOT-CREATED / NOT-RESUMED / LOCKED / CREATED	NONE	The subsystem is in the displayed state. Since this is not a transition state, there is no further information.

Table 12: Contents of var(*LIST).SUBSYS-INT-STA depending on var(*LIST).SUBSYS-STA)

Notes

- A combination of the operands SUBSYSTEM-NAME=*ALL and VERSION=*ALL supplies the same amount of information as SUBSYSTEM-NAME=*ALL.
- If a subsystem is found in the local subsystem catalog, LOCAL is written in the output before the remaining information.
- SHOW-SUBSYSTEM-STATUS does not supply any information for subsystems declared with the attributes CONNECTION-SCOPE=*FREE and MEMORY-CLASS=*SYSTEM-GLOBAL.
  By contrast, subsystems with the attributes CONNECTION-SCOPE=*FREE and MEMORY-CLASS=*LOCAL-PRIVILEGED or *LOCAL-UNPRIVILEGED are included in the output.

- If different versions of a subsystem are loaded (reloading in coexistence or exchange mode), the privileged user can issue the SHOW-SUBSYSTEM-STATUS command without specifying a version (operand VERSION=*STD / *ALL) in order to request information on the status of all versions of the subsystem with a status unequal NOT-CREATED.
- Although tasks which are connected to a global subsystem with CONNECTION-SCOPE=*OPTIMAL are counted in the sum of all connected tasks, they are not explicitly listed with their TID and TSN.
- If an error occurs during output to SYSOUT (message `ESM0671`), output is continued in the normal way in S variables, provided this has been requested.
  If an error occurs during output in S variables (message `ESM0672` or `CMD2009`), output to SYSOUT is continued in the normal way.
- An interruption of command processing by pressing [K2] has the effect of interrupting both output to SYSOUT and output in S variables. The return code `OPS0002` is set.
- If the interrupt key [K2] is pressed during an input request (`PLEASE ACKNOWLEDGE`), output to SYSOUT is aborted immediately. Output in S variables is continued in the normal way, provided this has been requested. No message is output.
- It is only possible to guarantee that the outputs to SYSOUT and in S variables will fully match if the procedure issuing the command has prohibited interruption of the output by means of the operand INTERRUPT-ALLOWED=*NO.
- If a privileged user requires information about a specific subsystem (SUBSYSTEM-NAME=<structured-name 1...8>) and output in S variables, the USED-TASK-LIST field may be blank, even though it has been generated.
  The number of list elements can be determined with the builtin function SIZE().
- If the subsystem is in a transition state, further information on the actual processing state (sub-status) is output, see table 12 on page 147.

Examples

Example 1

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*NON-PRIVILEGED-CLASS-5 ————————— (1)
%SUBSYSTEM VAS-TU  /V02.1    IS NOT CREATED  IN CL5
%  :                   :            :         (all other subsystems in CL5)
```

(1) SYSOUT output of information on all subsystems occupying nonprivileged class 5 memory. Nonprivileged users receive the same information as privileged users.

Example 2

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=EDT ————————————————————————————— (1)
%SUBSYSTEM EDT      /V16.6 IS USED BY   8 TASKS ——————————————————————— (2)
%  TASKID  00070011  00020026  0002002A  00010038  0001003A  0001003B
%  TSN         01QB      01LX      XAAJ      XAAD      01LH      01LG
%  TASKID  0001003E  0001003F
%  TSN         01LF      01LE
%       38 CONNECTIONS SINCE STARTUP
%SUBSYSTEM EDT      /V16.5 IS USED BY   2 TASKS ——————————————————————— (3)
%  TASKID  0001004F  00010051
%  TSN         01LQ      01LR
%                    IS IN DELETE / WAIT-DISCON
%      252 CONNECTIONS SINCE STARTUP
```

(1) SYSOUT output of information on the subsystem EDT without specification of a version. Only privileged users receive the information reproduced above. Nonprivileged users receive only the following output line:

```
%SUBSYSTEM EDT    /V16.6 IS CREATED
%SUBSYSTEM EDT    /V16.5 IS IN DELETE / WAIT-DISCON
```

(2) The output indicates that two EDT subsystems (V16.6 and V16.5) are active. The subsystem EDT V16.6 was started in exchange mode and is to replace the subsystem EDT V16.5.

(3) The subsystem EDT V16.5 has the status IN-DELETE since it is still being used by two other tasks.

i For an example with output in S variables see section “Output in an S variable” on page 173.

START-LOCAL-SUBSYSTEM Activate local subsystem in user address space

Domain:	SYSTEM-MANAGEMENT
Privileges:	STD-PROCESSING

Function

The START-LOCAL-SUBSYSTEM command can be used to activate a local subsystem by loading its code into nonprivileged address space local to the task (user address space). If a syntax file and message file are defined for the subsystem, they will also be activated.

The command is aborted and a message is issued if

– the specified subsystem is already activated
– the specified subsystem is not contained in the local subsystem catalog
– the associated syntax file or message file cannot be activated
– the subsystem code cannot be loaded

A local subsystem cannot be activated if a program is loaded.

Local subsystems are always loaded above the 16MB boundary.
As local subsystems are loaded into the user address space (nonprivileged class 5 memory), starting several local subsystems at (almost) the same time may result in memory saturation and, as a consequence, the task may crash!

Format

START-**LOC**AL-**SUBSYS**TEM
SUBSYSTEM-NAME = <structured-name 1..8>
,**VERSION** = ***STD** / <product-version mandatory-man-corr> / <product-version without-man-corr> / ***HIGHEST**

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the local subsystem that is being activated.

VERSION = *STD / <product-version mandatory-man-corr> / <product-version without-man-corr> / *HIGHEST
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type "product-version" on page 13).

VERSION = *STD
If there is more than one version of the specified subsystem and if no version is specified or *STD is not explicitly specified, the subsystem with the lowest version number available in the local subsystem catalog is selected.

VERSION = *HIGHEST
Selects the highest version of the subsystem.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed, subsystem successfully started or Command executed, with warning
1	0	CMD0001	No action necessary; the subsystem is already active
	1	ESM0414	Syntax error: invalid version specified
	64	ESM0255	Command not executed (a program is loaded, the specified subsystem does not exist in the local subsystem catalog, or no local subsystem catalog is loaded)
	64	ESM0326	Command abnormally terminated

Example

```
/LOAD-LOCAL-SUBSYSTEM-CATALOG CATALOG-NAME=MY.LOCAL ——————————————————— (1)
%ESM0254 COMMAND 'LOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/START-LOCAL-SUBSYSTEM SUBSYSTEM-NAME=LOC-SUBS,VERSION='01.0'
%ESM0220 FUNCTION 'CREATE' FOR SUBSYSTEM 'LOC-SUBS/01.0' COMPLETELY PROCESSED

/START-LOCAL-SUBSYSTEM SUBSYSTEM-NAME=NO-NAME ————————————————————————— (2)
%ESM0201 SUBSYSTEM 'NO-NAME' NOT FOUND
%ESM0255 'START-LOCAL-SUBSYSTEM' COMMAND NOT PROCESSED
```

(1) First a local subsystem catalog is loaded, then a local subsystem is activated. Both commands have been successfully executed.

(2) The second local subsystem cannot be activated because it is not known in the current local subsystem catalog.

START-SUBSYSTEM
Activate subsystem

Domain:	SYSTEM-MANAGEMENT
Privileges:	OPERATING SUBSYSTEM-MANAGEMENT
Routing code:	R

Function

This command enables system administration to activate any desired subsystem.
The following information from the dynamic subsystem catalog is used to activate the subsystem:

– information on loading and linking the subsystem
– information on initialization/deinitialization and on terminating the job relations
– information on calling points, subcomponents and operating dependencies (see corresponding SSCM statements in chapter “SSCM” on page 179)

This command is rejected if:

– the subsystem cannot be found in the dynamic subsystem catalog
– another version of the subsystem exists and coexistence is not permitted (see the VERSION-PARALLELISM operand on page 156)
– subsystems upon which the subsystem to be activated is dependent have not been loaded
– a necessary file is missing (e.g. message file, library)

The operator or system administration receives a message confirming that the command has been accepted or rejected. By means of the operand RESET=*YES, it is possible to enforce initialization of subsystems which are in the process of being deactivated. Any number of START-SUBSYSTEM commands can be issued in different tasks under the privileged user ID of the system administrator, unless this is expressly prohibited by the parameters defined during subsystem definition.

Depending on the subsystem definition (SSCM statement SET-/MODIFY-SUBSYSTEM-ATTRIBUTES with the SUBSYSTEM-LOAD-MODE operand), it can be activated in three different ways:

– SUBSYSTEM-LOAD-MODE=*STD
  The BLS is called in STD run mode and loads the subsystem as an object module.
– SUBSYSTEM-LOAD-MODE=*ADVANCED
  The BLS is called in ADVANCED run mode and loads the subsystem as a link and load module (LLM).
– SUBSYSTEM-LOAD-MODE=*ANY
  The BLS is called in STD run mode and loads the subsystem as an object module. If an error occurs during loading, the BLS is called a second time. This call is in ADVANCED run mode and the subsystem is loaded as a link and load module (LLM).
  If the first BLS call is not successful, a BLS error messages is output at the console.

Format

START-SUBSYSTEM
SUBSYSTEM-NAME = <structured-name 1..8>
,**VERSION** = ***STD** / <product-version mandatory-man-corr> / <product-version without-man-corr> / ***HIGHEST**
,**SUBSYS**TEM-**PAR**AMETER = ***NONE** / <c-string 1..254>
,**RESET** = ***NO** / ***Y**ES
,**SYNCH**RONOUS = ***NO** / ***Y**ES
,**VERSION-PARALLELISM** = ***NONE** / ***EXCHANGE-MODE**(...) / ***COEXIST**ENCE**-MODE**
***EXCHANGE-MODE**(...)
⏐ **SUBSYS**TEM-**PAR**AMETER = ***NONE** / <c-string 1..254>
,**MONJV** = ***NONE** / <filename 1..54 without-gen-vers>

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem to be activated.

VERSION = *STD / <product-version mandatory-man-corr> / <product-version without-man-corr> / *HIGHEST
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type "product-version" on page 13).

VERSION = *STD
If there is more than one version of the specified subsystem and if none of these versions is specified or if *STD is specified explicitly, the subsystem declared with the start attribute CREATION-TIME=*AT-SUBSYSTEM-CALL (see the SSCM statement SET-SUBSYSTEM-ATTRIBUTES on page 243) is loaded. If this condition does not apply, the lowest version number for this subsystem contained in the static subsystem catalog is selected.

Exception

If a version of a subsystem is to be activated automatically with the first SVC call, this version is the default version.

VERSION = *HIGHEST
Selects the highest version of the subsystem.

SUBSYSTEM-PARAMETER = *NONE / <c-string 1..254>
Specifies whether special parameters are to be processed, which can be evaluated only by the specified subsystem.

RESET =
Influences the mode and urgency of command processing.

RESET = *NO
If the relevant subsystem is in the process of being deactivated, the command is rejected until this disconnection operation has been completed.

RESET = *YES
The command is accepted regardless of any outstanding deactivation process and the subsystem or selected components is/are initialized (see notes below).
The version parameter is mandatory for this operand.

SYNCHRONOUS =
Allows you to choose between synchronous and asynchronous processing.

SYNCHRONOUS = *NO
The command is to be processed asynchronously, i.e. without having to wait for it to be executed before making a new entry. After the syntax of the command has been checked, the calling task receives message `ESM0216`. Error messages concerning execution of the command are output at the console.

SYNCHRONOUS = *YES
The command must first be executed before another entry can be made.
Appropriate error messages concerning execution are output to the task.
In the event of a version swap this entry is only of relevance for the new version that is to be activated. Deactivation of the other, “old” version is always carried out asynchronously.

VERSION-PARALLELISM =
Specifies whether different versions of the same subsystem may be active at the same time.

VERSION-PARALLELISM = *NONE
The coexistence of different versions of the same subsystem - regardless of what was specified at definition time - is not to be permitted. If a version has a status other than “NOT-CREATED”, activation is rejected.

VERSION-PARALLELISM = *EXCHANGE-MODE(...)
The temporary coexistence of two versions of the same subsystem is to be permitted.
Activation is permitted only if no version of the subsystem, or only one, has the status CREATED. If two versions already have this status, implicit deactivation of the most recently started version is initiated.
If a version of a subsystem has the status LOCKED, DSSM treats it as if it had the status NOT-CREATED.
In the following cases the command is rejected if specified with this operand:

– the version to be replaced has been defined with HOLD=*NO, but without a CLOSE-CTRL routine
– the command MODIFY-SUBSYSTEM-PARAMETER CHANGE-STATE=*NO was used for the version to be replaced
– the operand RESET=*NO was specified at the same time
– the version status is not CREATED, NOT-CREATED or LOCKED

SUBSYSTEM-PARAMETER = *NONE / <c-string 1..254>
Specifies whether special parameters are to be processed, which can be evaluated only by the specified subsystem.

VERSION-PARALLELISM = *COEXISTENCE-MODE
The coexistence of two or more versions of the same subsystem, without any restrictions, is to be permitted. Prerequisite: this has been authorized for all relevant versions in the SSCM statement SET-SUBSYSTEM-ATTRIBUTES.

MONJV = *NONE / <filename 1..54 without-gen-vers>
Defines the name of a monitoring job variable. The monitoring job variable indicates whether the subsystem is active, stopped, interrupted or locked. The contents of the monitoring job variable are listed on page 52.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
1	0	CMD0001	No action necessary; subsystem already active
	1	ESM0414	Syntax error: invalid version specified
	32	ESM0224	Command is not processed
	32	ESM0228	Command terminated abnormally

Notes

- Subsystems generally have a variety of dependency relations and load relations to other subsystems.
  Due account must be taken of these relationships in order to ensure the proper functioning of the individual subsystems. DSSM attempts to avoid potential conflicts caused by user requests by rejecting any commands which would result in such a conflict. Actions such as the installation of subsystems that do not exist or the unloading of dependent subsystems are not executed.
  However, if the user also generates complex subsystems with the statement CHECK-REFERENCE=*NO (see the SSCM statement SET-SUBSYSTEM-ATTRIBUTES on page 243), DSSM carries out the requested functions **despite** the risk of conflicts: the START-SUBSYSTEM command loads the specified subsystem even if a subsystem to which it has a defined relation has not yet been fully loaded.
- In order to ensure a high degree of parallelism and data integrity, time-consuming administration tasks are not executed under control of the calling task; instead they are transferred to a DSSM task.
  As a rule, only the check of the requested function is carried out **synchronously** (i.e. contingent upon a wait state for the calling task). DSSM carries out the actual processing **asynchronously** and independent of the calling task.

- A START-SUBSYSTEM command after a STOP-SUBSYSTEM command will be rejected if DSSM has not yet completed the “unload subsystem” operation. However, by specifying RESET=*YES system administration can enforce the unconditional loading of the subsystem; it is not necessary to wait until the STOP-SUBSYSTEM command has been completely processed.
  In this case the initialization routine is started and the relevant subsystem is informed of the RESET and can define the scope of the INIT routine itself (complete initialization, partial initialization or no initialization).

  Exception

  If the relevant subsystem still has the status IN-DELETE even though it has been deinitialized, the “unload subsystem” operation is not interrupted, despite the specification RESET=*YES. The START-SUBSYSTEM command is rejected until the subsystem has the status NOT-CREATED and has released all resources.

- If two versions of a subsystem are to be exchanged, the following points should be borne in mind when using the operand RESET=*YES:
  - if version A has the status IN-DELETE and version B has the status CREATED, RESET=*YES can only be specified for A if coexistence was stipulated in the SSCM-definition of both versions (see page 47)
  - if both versions have the status IN-DELETE, RESET=*YES is permissible for one of these versions if it was defined with RESET=*ALLOWED, VERSION-EXCHANGE=*ALLOWED.
- A restart (i.e. invocation of the INIT routine for subsystems defined with RESTART-REQUIRED=*YES) is prohibited, as it might otherwise lead to illegal coexistence situations.

STOP-LOCAL-SUBSYSTEM
Deactivate local subsystem in user address space

Domain:	SYSTEM-MANAGEMENT
Privileges:	STD-PROCESSING

Function

The STOP-LOCAL-SUBSYSTEM command can be used to deactivate a local subsystem. If a syntax file and a message file are defined for the subsystem, they will also be deactivated. The subsystem code is unloaded.

A local subsystem cannot be deactivated unless it was previously activated with the START-LOCAL-SUBSYSTEM command or if a program is still loaded. A message is output in both of these cases.

Deactivation of the specified local subsystem is also terminated if it is not possible to deactivate the associated syntax file and/or message file without error.

Format

STOP-LOCAL-SUBSYSTEM
SUBSYSTEM-NAME = <structured-name 1..8>
,VERSION = ***STD** / <product-version mandatory-man-corr> / <product-version without-man-corr> / ***HIGHEST**

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the local subsystem that is to be deactivated.

VERSION = *STD / <product-version mandatory-man-corr> / <product-version without-man-corr> / *HIGHEST
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type "product-version" on page 13).

VERSION = *STD
If there is more than one version of the specified subsystem, the currently active subsystem is selected.

VERSION = *HIGHEST
Selects the highest version of the subsystem.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
1	0	CMD0001	No action necessary; subsystem already active
	1	ESM0414	Syntax error: invalid version specified
	32	ESM0255	Command not executed (a program is loaded, the specified subsystem does not exist in the local subsystem catalog, or no local subsystem catalog is loaded)
	32	ESM0326	Command abnormally terminated

Example

```
/load-local-subsystem-catalog catalog-name=local.dssmcat.1 ————————————  (1)
%ESM0254 COMMAND '/LOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/load-local-subsystem-catalog catalog-name=local.dssmcat.11 ———————————  (1)
%ESM0254 COMMAND 'LOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/start-local-subsystem subsystem-name=loc-subs,version='01.0'
%ESM0220 FUNCTION 'CREATE' FOR SUBSYSTEM 'LOC-SUBS/01.0' COMPLETELY PROCESSED
/stop-local-subsystem subsystem-name=loc-subs,version='01.0' ——————————  (2)
%ESM0220 FUNCTION 'DELETE' FOR SUBSYSTEM 'LOC-SUBS/01.0' COMPLETELY PROCESSED
/unload-local-subsystem-catalog
%ESM0254 COMMAND 'UNLOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/stop-local-subsystem subsystem-name=loc-subs,version= '01.0' —————————  (3)
%ESM0341 NO LOCAL SUBSYSTEM CATALOG LOADED
%ESM0255 'STOP-LOCAL-SUBSYSTEM' COMMAND NOT PROCESSED
```

(1) First a local subsystem catalog is loaded, then a local subsystem is activated. Both commands were successfully executed.

(2) The local subsystem is deactivated. The current local subsystem catalog is then unloaded. Both commands were successfully executed.

(3) The local subsystem cannot be deactivated (for the second time), because there is no current local subsystem catalog.

STOP-SUBSYSTEM Deactivate subsystem

Domain:	SYSTEM-MANAGEMENT
Privileges:	OPERATING SUBSYSTEM-MANAGEMENT

Function

This command enables system administration to deactivate any desired subsystem. The function and execution of the command can be described as follows:

1. The relevant subsystem is closed for all new callers.
2. The subsystem is deactivated as soon as all jobs accessing it have been terminated normally. If the operand FORCED=*YES was specified in the STOP-SUBSYSTEM command, all jobs accessing the subsystem are aborted immediately. (However, the operand FORCED=*YES is only accepted if a previous STOP-SUBSYSTEM command with the operand FORCED=*NO failed to terminate the jobs.)
3. The subsystem is unloaded.
4. All resources in use are released.

The STOP-SUBSYSTEM command is rejected if:

– the subsystem cannot be found in the dynamic subsystem catalog
– already activated subsystems or subsystems in the process of being activated are dependent on the subsystem to be deactivated
– already activated subsystems or subsystems in the process of being activated have link relations with the subsystem to be deactivated
– the operand FORCED=*YES was specified without a previous attempt to terminate the jobs by means of FORCED=*NO

Format

STOP-SUBSYSTEM
SUBSYSTEM-NAME = <structured-name 1..8> ,VERSION = ***STD** / <product-version mandatory-man-corr> / <product-version without-man-corr> / ***HIGHEST** ,SUBSYSTEM-PARAMETER = ***NONE** / <c-string 1..254> ,FORCED = ***NO** / ***Y**ES ,SYNCHRONOUS = ***NO** / ***Y**ES

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem to be deactivated.

VERSION = *STD / <product-version mandatory-man-corr> / <product-version without-man-corr> / *HIGHEST
Specifies the version number.
When specifying a version number, the format entered must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type “product-version” on page 13).

VERSION = *STD
If there is only **one** version of the subsystem loaded, this version is selected.
If there are **two or more** loaded versions, the appropriate version must be specified.

VERSION = *HIGHEST
Selects the highest version of the subsystem.

SUBSYSTEM-PARAMETER = *NONE / <c-string 1..254>
Specifies whether special parameters are to be processed, which can be evaluated only by the specified subsystem.

FORCED =
Defines the mode and urgency of command processing.

FORCED = *NO
The processing and thus normal termination of all tasks accessing this subsystem must be completed before the subsystem is deactivated.

FORCED = *YES
The immediate termination of all processes is initiated.
In the case of a privileged subsystem, this may result in a system dump. Tasks connected to a nonprivileged subsystem can exit via the STXIT error routine provided by DSSM.

SYNCHRONOUS =
Allows you to choose between synchronous and asynchronous processing.

SYNCHRONOUS = *NO
The command is to be processed asynchronously, i.e. without having to wait for it to be executed before making a new entry. After the syntax of the command has been checked, the calling task receives message `ESM0216`. Error messages concerning execution of the command are output at the console.

SYNCHRONOUS = *YES
The command must first be executed before another entry can be made.
Appropriate error messages concerning execution are output to the task.
In the event of a version swap this entry is only of relevance for the new version that is to be activated. Deactivation of the other, “old” version is always carried out asynchronously.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
1	0	CMD0001	No action necessary; subsystem is no longer active
	1	ESM0414	Syntax error: invalid version specified
	32	ESM0224	Command is not processed
	32	ESM0228	Command terminated abnormally

Notes

- In order to ensure a high degree of parallelism and data integrity, time-consuming administration tasks are not performed under control of the calling task; instead they are transferred to a DSSM task.
  As a rule, only the check of the requested function is carried out **synchronously** (i.e. contingent upon a wait state for the calling task). DSSM carries out the actual processing **asynchronously** and independent of the calling task.
- STOP-SUBSYSTEM with the operand FORCED=*YES is accepted only if preceded by FORCED=*NO and if the subsystem is simply waiting for deactivation of the task accessing it.
  The FORCED function cannot guarantee the “normal” behavior of tasks which are connected to a privileged subsystem.
  Tasks connected to a nonprivileged subsystem can run an error routine which enables them to continue processing.
  The FORCED function is implemented by running a contingency routine for every task connected to the subsystem. Deactivation of the task is completed when this contingency routine has been executed.
  However, since DSSM does not wait for this routine to be executed, these tasks can be registered as still connected to the subsystem after an intermediate START-SUBSYSTEM command.

UNLOAD-LOCAL-SUBSYSTEM-CATALOG Unload local subsystem catalog

Domain:	SYSTEM-MANAGEMENT
Privileges:	STD-PROCESSING

Function

The UNLOAD-LOCAL-SUBSYSTEM-CATALOG command can be used by callers to unload a local subsystem catalog from their own task-specific user address space (class 5 memory).

If there are any subsystems still active at this time, they are automatically deactivated and execution of the UNLOAD-LOCAL-SUBSYSTEM-CATALOG command is resumed.

The command is rejected with an error message if a local subsystem catalog has not previously been loaded into the user address space with the LOAD-LOCAL-SUBSYSTEM-CATALOG command.
A local subsystem catalog cannot be unloaded if a program is loaded at the same time.

Format

UNLOAD**-LOC**AL**-SUBSYS**TEM**-CAT**ALOG

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command successfully executed
	32	ESM0255	Command not executed; there is no local catalog loaded or a program is loaded
	32	ESM0326	Command processing aborted (system error)

Example

```
/load-local-subsystem-catalog catalog-name=local.dssmcat.1 ────────────  (1)
%ESM0254 COMMAND 'LOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/unload-local-subsystem-catalog ──────────────────────────────────────  (2)
%ESM0254 COMMAND 'UNLOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/unload-local-subsystem-catalog ──────────────────────────────────────  (3)
%ESM0341 NO LOCAL SUBSYSTEM CATALOG LOADED
%ESM0255 'UNLOAD-LOCAL-SUBSYSTEM-CATALOG' COMMAND NOT PROCESSED
```

(1) First a local subsystem catalog is loaded. Execution of the command is successful.

(2) Unloading of the local subsystem catalog is also successful.

(3) The second attempt to unload the local subsystem catalog is rejected with message `ESM0341`: only a local subsystem catalog that has previously been loaded can be unloaded.

UNLOCK-SUBSYSTEM Shift subsystem from LOCKED status to NOT-CREATED status

Domain:	SYSTEM-MANAGEMENT
Privileges:	SUBSYSTEM-MANAGEMENT

Function

This command can be used by systems support to convert a locked subsystem (subsystem with the status LOCKED) back into a declared but not activated status (NOT-CREATED status). In other words, the subsystem is unlocked during the current session. UNLOCK-SUBSYSTEM therefore contributes to ensuring interrupt-free operation of BS2000/OSD.

By means of its INIT, DEINIT, STOPCOM or CLOSE-CTRL routines a subsystem can be placed in the LOCKED status. These routines either request DSSM to lock the subsystem themselves or they trigger a dump and the restart of the holder task, which can then no longer be executed without errors (RESTART-REQUIRED=*NO or - if *YES applies - the maximum number of permissible attempts is exceeded).
If a subsystem is locked during the activation phase (INIT routine), this would mean that, without the UNLOCK-SUBSSYSTEM command, the subsystem is not available until shutdown and the subsequent RESTART command.

It is worth bearing in mind that not all subsystems can be unlocked without problems and that restarting is not always possible (cf. the notes on the following page).

Format

UNLOCK-**SUBSYS**TEM
SUBSYSTEM-NAME = <structured-name 1..8> ,**VER**SION = <product-version mandatory-man-corr> / <product-version without-man-corr>

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem that is to be unlocked.

VERSION = <product-version mandatory-man-corr> /
<product-version without-man-corr>
Version number of this subsystem. The format specified here must match the format used when the subsystem was defined. Release and correction levels either must be specified or may not be specified (see statements SET-SUBSYSTEM-ATTRIBUTES on page 243 and SDF data type "product-version" on page 13).

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
	1	ESM0414	Syntax error: invalid version specified
	32	ESM0228	Command abnormally terminated
	64	ESM0224	Command not processed

Notes

The subsystem that is to be unlocked must have the LOCKED status.
Use of the UNLOCK-SUBSYSTEM command is subject to the following restrictions:

- Some subsystems cannot be terminated normally and restarted within the same session. It must be possible to execute the STOP-SUBSYSTEM and HOLD-SUBSYSTEM commands for the subsystem to be unlocked, i.e. the subsystem must not have been defined with the attribute SUBSYSTEM-HOLD=*FORBIDDEN (cf. the restrictions of the command STOP-SUBSYSTEM described on page 162).
- Dependencies on other subsystems may mean that the UNLOCK-SUBSYSTEM command leads to inconsistencies in the subsystem catalog.
  If this situation is to be avoided, all subsystems that relate to this subsystem must first be stopped with /STOP-SUBSYSTEM. They must then be restarted together with the unlocked subsystem with /START-SUBSYSTEM.
- Even if the subsystem can be restarted from every status and if all dependencies to other subsystems - if there are any - have been taken into account (see above), it is not absolutely certain that the subsystem will be active following execution of the UNLOCK-SUBSYSTEM command and when the next START-SUBSYSTEM command is issued. This restriction applies in particular if problems occur which are caused by the subsystem itself.

3.9 Examples

3.9.1 Use of local subsystems

```
/START-SSCM ----------------------------------------------------------- (1)
% BLS0517 MODULE 'SSCM' LOADED
% SCM5000 PROGRAM SSCM VERSION V02.3B10 STARTED

//START-CATALOG-CREATION CATALOG-NAME=MEIN.KATALOG  }
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSC.SDF-A.040      }-------------- (2)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSC.INIT.130       }
//CHECK-CATALOG -------------------------------------------------------- (3)
CHECK REPORT:
**** NO ERROR ****
CHECK OF LINK REFERENCES:
VERSION RANGE CHECK:
**** NO ERROR ****
LINK RELATION CHECK:
**** NO ERROR ****
CHECK OF FUNCTIONAL DEPENDENCE:
VERSION RANGE CHECK:
**** NO ERROR ****
DEPENDENCE RELATION CHECK:
**** NO ERROR ****
CYCLE CHECK:
**** NO ERROR ****
//SAVE-CATALOG --------------------------------------------------------- (4)
% SCM0200 CATALOG 'MEIN.KATALOG' GENERATED

//END
% SCM0405 SSCM NORMALLY TERMINATED


/LOAD-LOCAL-SUBSYSTEM-CATALOG CATALOG-NAME=MEIN.KATALOG --------------- (5)
% ESM0254 COMMAND 'LOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL ---------------------------- (6)
% LOCAL SUBSYSTEM INIT    /V13.0    IS NOT CREATED
% LOCAL SUBSYSTEM SDF-A   /V04.0    IS NOT CREATED
% SUBSYSTEM INIT    /V14.0    IS CREATED
% SUBSYSTEM SSCM    /V02.3    IS CREATED
% SUBSYSTEM SDF-A   /V04.1    IS CREATED
/START-SDF-A ---------------------------------------------------------- (7)
% BLS0517 MODULE 'SDF-A' LOADED
% SDA0001 'SDF-A' VERSION '04.1E10' STARTED
//END
```

```
/START-LOCAL-SUBSYSTEM SUBSYSTEM-NAME=SDF-A,VERSION=04.0 ————————————— (8)
% ESM0220 FUNCTION 'CREATE' FOR SUBSYSTEM 'SDF-A /04.0' COMPLETELY
          PROCESSED

/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL —————————————————————————— (9)
LOCAL SUBSYSTEM INIT    /V13.0    IS NOT CREATED
LOCAL SUBSYSTEM SDF-A   /V04.0    IS CREATED
% SUBSYSTEM INIT    /V14.0    IS CREATED
% SUBSYSTEM SSCM    /V02.3    IS CREATED
% SUBSYSTEM SDF-A   /V04.1    IS CREATED
/START-SDF-A ———————————————————————————————————————————————————————— (10)
% BLS0517 MODULE 'SDF-A' LOADED
% SDA0001 'SDF-A' VERSION '04.0A00' STARTED
//END
/SELECT-PRODUCT-VERSION PRODUCT-NAME=SDF-A,VERSION=04.1,-
       SCOPE=*PROGRAM ——————————————————————————————————————————————— (11)
/START-SDF-A ———————————————————————————————————————————————————————— (12)
% BLS0517 MODULE 'SDF-A' LOADED
% SCM5000 PROGRAM SDF-A VERSION V04.1E10 STARTED
//END
% SCM0405 SDF-A NORMALLY TERMINATED
/STOP-LOCAL-SUBSYSTEM SUBSYSTEM-NAME=SDF-A,VERSION=04.0 —————————————— (13)
% ESM0220 FUNCTION 'DELETE' FOR SUBSYSTEM 'SDF-A /04.0' COMPLETELY
          PROCESSED
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL —————————————————————————— (14)
% LOCAL SUBSYSTEM INIT    /V13.0    IS NOT CREATED
% LOCAL SUBSYSTEM SDF-A   /V04.0    IS NOT CREATED
% SUBSYSTEM INIT    /V14.0    IS CREATED
% SUBSYSTEM SSCM    /V02.3    IS CREATED
% SUBSYSTEM SDF-A   /V04.1    IS CREATED
/START-SDF-A ———————————————————————————————————————————————————————— (15)
%  BLS0517 MODULE 'SDF-A' LOADED
%  SDA0001 'SDF-A' VERSION '04.1E10' STARTED
//END
/SELECT-PRODUCT-VERSION PRODUCT-NAME=SDF-A,VERSION=04.0, -
       SCOPE=*PROGRAM ——————————————————————————————————————————————— (16)
/START-SDF-A ———————————————————————————————————————————————————————— (17)
%  BLS0334 SYMBOL 'SDF-A' CANNOT BE FOUND. LOADING ABORTED
%  NRTT101 ABNORMAL JOBSTEP TERMINATION BLS0532
/UNLOAD-LOCAL-SUBSYSTEM-CATALOG —————————————————————————————————————— (18)
% ESM0254 COMMAND 'UNLOAD-LOCAL-SUBSYSTEM-CATALOG' COMPLETELY PROCESSED
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL —————————————————————————— (19)
% SUBSYSTEM INIT    /V14.0    IS CREATED
% SUBSYSTEM SSCM    /V02.3    IS CREATED
% SUBSYSTEM SDF-A   /V04.1    IS CREATED
```

(1) The SSCM catalog manager is called.

(2) A local subsystem catalog with the name MY.CATALOG is created.
The subsystems SDF-A V04.0 and INIT V13.0 are added as local subsystems.

(3) The local catalog is then checked to confirm the compatibility and consistency of the two local subsystems. The check reveals no errors.

(4) The local catalog MY.CATALOG is stored and SSCM is terminated.

(5) The local catalog MY.CATALOG is loaded in the user address space.

(6) From this point on, the local subsystems (as well as the global subsystems) can be displayed and/or - if requested - output in an S variable.
The two local subsystems SDF-A V4.0 and INIT V13.0 are stored in the local catalog but have the status NOT-CREATED because they have not yet been activated.
The three subsystems listed are currently available in the global subsystem catalog.

(7) For example, if a START-SDF-A command is issued, it calls the global subsystem SDF-A V4.1 and not the local subsystem with V4.0.
The subsystem SDF-A V4.1 is terminated without executing any statements.

(8) The local subsystem SDF-A V4.0 is activated.

(9) The display shows that the subsystem now has the status CREATED.

(10) When START-SDF-A is called, the local subsystem version has priority over the global subsystem.

(11) The priority of the different subsystems is modified in favor of the global subsystem SDF-A V4.1.

(12) The START-SDF-A command again calls the global subsystem SDF-A V4.1.

(13) The local subsystem SDF-A is deactivated.

(14) Both local subsystems are deactivated.

(15) When START-SDF-A is called, only the global subsystem can be started.

(16) The priority of the different subsystems is modified in favor of the local subsystem SDF-A V4.0.

(17) hen START-SDF-A is called, this version cannot be found (it does not have the status CREATED).

(18) The local subsystem catalog is unloaded.

(19) The display no longer shows any local subsystems.

3.9.2 Output in an S variable

This example illustrates output in S variables as a function of the privileges of the caller. It does not represent a situation lifted from actual practice; it is rather an artificial, simplified example designed specially to show the principles involved.

The following privileges exist:

a) The caller has only the privilege STD-PROCESSING

b) The caller has the privileges STD-PROCESSING and SUBSYSTEM-MANAGEMENT or OPERATING

c) The caller does not have the privilege STD-PROCESSING, but does have at least one of the privileges SUBSYSTEM-MANAGEMENT and OPERATING

The following configuration exists:

– the local subsystem AA V04.5 has the status CREATED

– the global nonprivileged subsystem AA V04.6 has the status CREATED; one task is connected (TID=00010054, TSN=0123)

– the global, privileged subsystem XX V01.0 has the status CREATED: one task is connected (TID=0002006F, TSN=0BFC)

– the global, privileged subsystem XX V02.0 has the status IN-DELETE: two tasks are connected (TID=00070015, TSN=0CMM and TID=00010057, TSN=00AP)

A compound variable of the type “list” with the name DATA is declared and assigned to the S variable stream SYSINF.

```
/DECLARE-VARIABLE VAR-NAME=DATA(TYPE=*STRUCTURE),
MULTIPLE-ELEMENTS=*LIST
/ASSIGN-STREAM STREAM-NAME=SYSINF,TO=*VAT(VAR-NAME=DATA)
/SHOW-STREAM-ASSIGNMENT
```

```
       STREAM-NAME   = SYSINF
        ASSIGN-LEVEL = 0
        DESTINATION  = *VARIABLE
          VARIABLE-NAME = DATA
               VAR-MODE = *EXTEND
          RETURN-VARIABLE-NAME = *NONE
          CONTROL-VAR-NAME = *NONE
          RET-CONTROL-VAR-NAME = *NONE
    STREAM-NAME   = SYSMSG
    :
```

a) The caller has only the privilege **STD-PROCESSING**

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL ————————————————————————— (1)
% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED
% SUBSYSTEM AA     /V04.6    IS CREATED
/SHOW-VAR DATA ——————————————————————————————————————————————————— (2)
   :
   DATA(*LIST).SUBSYS-TYPE='*LOC'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.5'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.6'
   DATA(*LIST).SUBSYS-STA='*CREATED'

/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=AA ——————————————————————————— (3)
% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED
% SUBSYSTEM AA      /V04.6    IS CREATED
/SHOW-VAR DATA ——————————————————————————————————————————————————— (4)
   :
   DATA(*LIST).SUBSYS-TYPE='*LOC'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.5'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.6'
   DATA(*LIST).SUBSYS-STA='*CREATED'
```

(1) The names, versions and statuses of all global, nonprivileged subsystems and of all local subsystems are output to SYSOUT.
In our specially constructed example, only the global subsystem AA is nonprivileged. It is output - together with the local subsystem AA.

(2) Output in the S variable DATA explicitly contains the subsystem type “global” or “local”.

(3) The restriction concerning output to subsystem AA does not result in any changes in the output to SYSOUT, in contrast to (1).

(4) The restriction concerning output to subsystem AA does not result in any changes in the output to the S variable DATA in contrast to (2).

b) The caller has the privileges **STD-PROCESSING** and **SUBSYSTEM-MANAGEMENT** and/or **OPERATING**

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL ────────────────────────  (5)
% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED
% SUBSYSTEM AA     /V04.6    IS CREATED
% SUBSYSTEM XX     /V01.0    IS CREATED
% SUBSYSTEM XX     /V02.0    IS IN DELETE / WAIT-DISCON
/SHOW-VAR DATA ────────────────────────────────────────────────────  (6)
   :
   DATA(*LIST).SUBSYS-TYPE='*LOC'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.5'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.6'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='01.0'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='02.0'
   DATA(*LIST).SUBSYS-STA='*IN-DELETE'
   DATA(*LIST).SUBSYS-INT-STA='WAIT-DISCON'

/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=AA ──────────────────────────  (7)
% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED
% SUBSYSTEM AA     /V04.6    IS USED BY 1 TASK
% TASKID 00010054
% TSN        0123
%   4 CONNECTIONS SINCE STARTUP
/SHOW-VAR DATA ────────────────────────────────────────────────────  (8)
   :
   DATA(*LIST).SUBSYS-TYPE='*LOC'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.5'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.6'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).CONN-NUM-SINCE-START=4
   DATA(*LIST).USED-TASK-LIST(*LIST).TID='00010054'
   DATA(*LIST).USED-TASK-LIST(*LIST).TSN='0123'
```

(5) The names, versions and statuses of all global and local subsystems are output to SYSOUT. In other words, information is provided on all subsystems loaded in the current system.
In our specially constructed example, this refers to the global subsystems XX (privileged, two versions) and AA (nonprivileged) and to the local subsystem AA.

(6) Output in the S variable DATA explicitly contains the subsystem type “global” or “local”.

(7) The restriction affecting the output to subsystem AA does not result in any changes as compared with (1): any tasks connected to a subsystem are shown together with their TSN and TID. The total number of tasks connected to this subsystem since startup is likewise output to SYSOUT.

(8) Output from the subsystems AA in the S variable DATA contains the same information as in the output to SYSOUT, as well as additional information concerning the subsystem type “global” or “local”.

c) The caller does not have the privilege STD-PROCESSING but does have one of the privileges **SUBSYSTEM-MANAGEMENT** or **OPERATING**

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL ───────────────────────── (9)
% SUBSYSTEM AA     /V04.6    IS CREATED
% SUBSYSTEM XX     /V01.0    IS CREATED
% SUBSYSTEM XX     /V02.0    IS IN DELETE / WAIT-CLS-CTRL
/SHOW-VAR DATA ──────────────────────────────────────────────────── (10)
   :
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.6'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='01.0'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='02.0'
   DATA(*LIST).SUBSYS-STA='*IN-DELETE'
   DATA(*LIST).SUBSYS-INT-STA='WAIT-CLS-CTRL'
```

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=XX ─────────────────────────────── (11)
% SUBSYSTEM XX     /V0.0    IS USED BY 1 TASK
% TASKID 0002006F
% TSN       0BFC
%   7 CONNECTIONS SINCE STARTUP
% SUBSYSTEM XX     /V02.0    IS USED BY 2 TASKS
% TASKID 00070015 00010057
% TSN       0CMM     00AP
%                         IS IN DELETE / WAIT-DISCON
% 12 CONNECTIONS SINCE STARTUP
/SHOW-VAR DATA ────────────────────────────────────────────────────────── (12)
   :
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='01.0'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).CONN-NUM-SINCE-START=7
   DATA(*LIST).USED-TASK-LIST(1).TID='0002006F'
   DATA(*LIST).USED-TASK-LIST(1).TSN='0BFC'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='02.0'
   DATA(*LIST).SUBSYS-STA='*IN-DELETE'
   DATA(*LIST).SUBSYS-INT-STA='WAIT-DISCON'
   DATA(*LIST).CONN-NUM-SINCE-START=12
   DATA(*LIST).USED-TASK-LIST(*LIST).TID='00070015'
   DATA(*LIST).USED-TASK-LIST(*LIST).TSN='0CMM'
   DATA(*LIST).USED-TASK-LIST(*LIST).TID='00010057'
   DATA(*LIST).USED-TASK-LIST(*LIST).TSN='00AP'
```

(9) The names, versions and statuses of all global subsystems are output to SYSOUT. In our specially constructed example, this refers to the global subsystems XX (privileged, two versions) and AA (nonprivileged).

(10) Output in the S variable DATA contains the same information as in output to SYSOUT.

(11) The restriction concerning output to the subsystem XX produces the following changes as compared with (9): any tasks connected to a subsystem are displayed with their TSN and TID. The total number of tasks connected to this subsystem since startup is likewise output to SYSOUT.

(12) Output from the subsystems XX in the S variable DATA contains the same information as in output to SYSOUT, with the addition of information concerning the subsystem type “global”.

4 SSCM

The subsystem SSCM (Static Subsystem Catalog Manager) ensures flexible, user-friendly management of the static subsystem catalogs (SSMCAT).

SSCM V2.3 is available as of BS2000/OSD-BC V3.0 and for working as of DSSM V3.8.

For the procedure for installing SSCM V2.3, see page 289; for information on the coexistence of SSCM versions, see page 289, and for information on version dependencies between SSCM, DSSM and BS2000/OSD, see page 28.

4.1 Generating a subsystem catalog

The subsystem catalog which is to be created must be placed on the home pubset and stored under the TSOS user ID. The catalog may be given any name desired, and the name can be declared to the system by means of the parameter service (see page 65).

The program provided for generating a subsystem catalog is SSCM.

The diagram on the following page shows the procedure for generating a subsystem catalog.

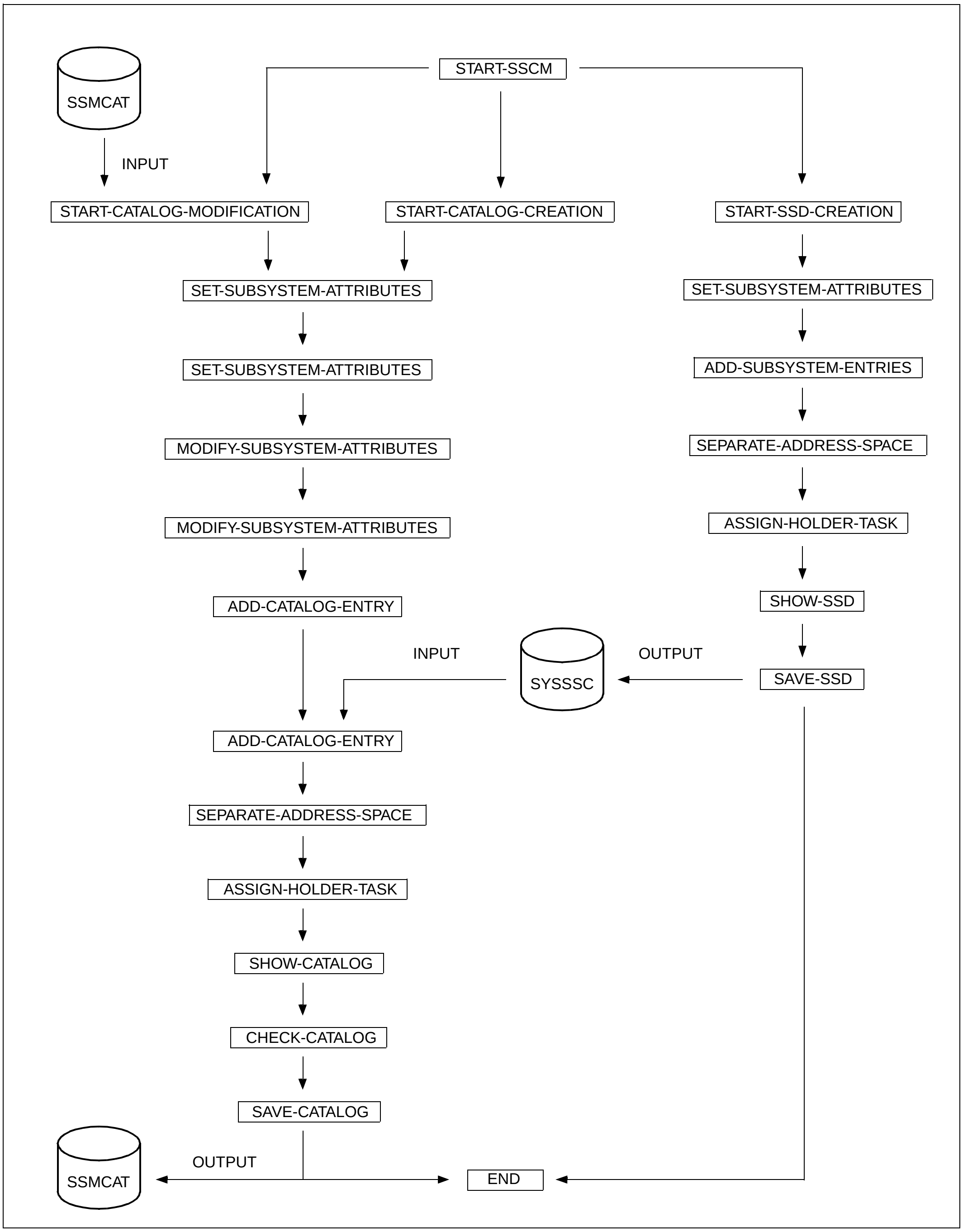

Figure 1: Generating a subsystem catalog

4.2 Starting and terminating SSCM

To start the SSCM program, use the START-SSCM command.

START-**SSCM**
MONJV = ***NONE** / <filename 1..54 without-gen-vers>
,**CPU-**LIMIT = ***JOB-REST** / <integer 1..32767>

You can also call up SSCM using the abbreviation **SSCM**.

The message file for SSCM ($TSOS.SYSMES.SSCM.023) is activated and the SSCM link load module in the SYSLNK.SSCM.023 module library is loaded (message `BLS0517`).

To terminate SSCM, use the **END** statement.

Monitoring programs by means of monitoring job variables

A job variable which is to be used to monitor a program must be declared in the START-SSCM command with MONJV=<jv-name>.

SSCM can set the monitoring variable to the following values:

`‘$T0000‘`	The program terminated normally.
`‘$T1010‘`	Statement rejected; execution of the program will be continued.
`‘$A2010‘`	Statement rejected; the program has been terminated.
`‘$A2015‘`	Unexpected EOF on SYSDTA; the program has been terminated.
`‘$A3020‘`	SSCM-internal error; the program is being terminated.

For details of program monitoring by means of monitoring job variables refer also to the “Commands Volume 1-5” [19] and “Job Variables” [18] manuals.

Monitoring by means of task switches

If a statement is abnormally terminated by SSCM, task switch 31 is activated.

If task switch 1 is activated, SDF will merely execute a syntax check.

4.3 The SSCM statements

SSCM is offered with an SDF interface for the user interface. This gives the SSCM user all the functions and advantages offered by SDF - guided dialog, help texts for operands, the use of default values.

The **SDF syntax representation** of the commands is explained in section “SDF syntax representation” on page 5.

The following SSCM statements are available to (sub)system administration:

Command	Meaning	Page
ADD-CATALOG-ENTRY	Add subsystem definition(s) to subsystem catalog	183
ADD-SUBSYSTEM-ENTRIES	Define additional job entry points	186
ASSIGN-HOLDER-TASK	Distribute subsystems to holder tasks	193
CHECK-CATALOG	Check subsystem definition(s) for consistency	196
GENERATE-CATALOG-SOURCE	Create SSCM statement list for generation	198
MODIFY-SUBSYSTEM-ATTRIBUTES	Modify subsystem attributes	200
MODIFY-WORK-TASK-ATTRIBUTE	Modify work task parameters	233
REMOVE-ADDR-SPACE-SEPARATION	Revoke disjunctive distribution of subsystems in class 5 memory	235
REMOVE-CATALOG-ENTRY	Logically delete definition of subsystem from subsystem catalog	237
SAVE-CATALOG	Save subsystem catalog as PAM file	238
SAVE-SSD	Terminate subsystem definition(s)	240
SEPARATE-ADDRESS-SPACE	Control disjunctive distribution of subsystems in class 5 memory	241
SET-SUBSYSTEM-ATTRIBUTES	Define attributes and entry points of subsystem	243
SHOW-CATALOG	Show subsystem configuration	273
SHOW-SSD	Show contents of SSD object (subsystem definitions)	280
START-CATALOG-CREATION	Define name of static subsystem catalog	285
START-CATALOG-MODIFICATION	Modify static subsystem catalog	286
START-SSD-CREATION	Generate SSD object for adding subsystem definitions	287

Table 13: SSCM statements

ADD-CATALOG-ENTRY
Add subsystem definition(s) to subsystem catalog

Function

This statement copies the definition of new subsystems, which are held in an SSD object, into the subsystem catalog which is currently open.

This statement will be rejected if the catalog into which the definitions are to be integrated is not currently open as a result of a START-CATALOG-CREATION or START-CATALOG-MODIFICATION statement. Any definition for a subsystem which is already contained in the open catalog will be ignored; processing will continue, however, after output of a corresponding message.

Format

ADD-CATALOG-**ENTRY**
FROM-FILE = <filename 1..54>
,**INSTALL**ATION-**USERID** = ***UNCHA**NGED / ***DEF**AULT-**USERID** / <name 1..8>
,**CORR**ECTION-**STATE** = ***UNCHA**NGED / <c-string 3..3> / <text 3..3>

Operands

FROM-FILE = <filename 1..54>
Name of the file in which to search for the subsystem definition(s). This file must be an SSD object generated by SSCM of type ISAM, in which the attributes of one or more subsystems are stored.

INSTALLATION-USERID =
Specifies a user ID under which the subsystem satellites (REP file, object module library, message file, syntax file and subsystem information file) are expected, in cases where these files have not already been assigned to a user ID.

INSTALLATION-USERID = *UNCHANGED*
Default value: the files are expected under the user ID specified in the subsystem definition (SET-SUBSYSTEM-ATTRIBUTES statement, page 243).

INSTALLATION-USERID = *DEFAULT-USERID
The files are expected under the system's default user ID (prefix "$.").

INSTALLATION-USERID = <name 1..8>
User ID under which the files are expected. If a different ID was specified in the SET-SUBSYSTEM-ATTRIBUTES statement for an SSD object, it will be overwritten.

CORRECTION-STATE =
Replaces in the catalog the last three characters of the subsystem version from the SSD file; these characters indicate the release and correction states of the subsystem version. If the subsystem version in the SSD file consists of four characters, the three characters specified for CORRECTION-STATE are chained with them.

CORRECTION-STATE = *UNCHANGED*
Default value: the release and correction states remain unchanged.

CORRECTION-STATE = <text 3..3>
Specifies the release and correction states in the format: `ann`, in which the text elements have the following meanings:

a	Release state; alphabetic character
nn	Correction state; numeric characters

CORRECTION-STATE = <c-string 3..3>
Specifies the release and correction states as a character string in the format: `ann`; for the meanings of the text elements, see CORRECTION-STATE=<text 3..3>.

Notes

- If a subsystem definition contains file names without a user ID and if no installation ID is specified, the relevant DSSM task searches for the files under the user ID TSOS or, in the case of a local subsystem, under the user ID of the calling task.
  If the files are stored under a different user ID, this ID must be specified in the INSTALLATION-USERID operand of one of the commands ADD-CATALOG-ENTRY, SET-SUBSYSTEM-ATTRIBUTES or MODIFY-SUBSYSTEM-ATTRIBUTES.
- If an SSD object containing multiple subsystem definitions is specified in the FROM-FILE operand, the release and correction states of all the subsystems defined in this SSD object will be affected by the value of the CORRECTION-STATE and INSTALLATION-USERID operands.

- The file names of a subsystem are not changed if the release and correction states are specified. Users wishing to change these names must use the MODIFY-SUBSYSTEM-ATTRIBUTES statement to do so.
- If the CORRECTION-STATE operand is specified, a version check for mutually dependent subsystems or for subsystems which have address relations with one another may result in a CHECK-CATALOG error. This can be avoided by specifying suitable entries in MODIFY-SUBSYSTEM-ATTRIBUTES MODIFY-REFER-SUBS=..., MODIFY-RELATED-SUBS=....

ADD-SUBSYSTEM-ENTRIES
Define additional job entry points

Function

This statement can be used to define further job entry points for a subsystem in an object module file (SSD object) in addition to those already specified in SET-SUBSYSTEM-ATTRIBUTES, even exceeding the maximum permitted number of 100.

Each ADD-SUBSYSTEM-ENTRIES statement (which can be executed repeatedly for one and the same subsystem) can be used to define up to 100 additional job entry points for a single subsystem.

ADD-SUBSYSTEM-ENTRIES is rejected if no START-SSD-CREATION statement was executed beforehand.

The statement is rejected if the specified subsystem was defined with the SET-SUBSYSTEM-ATTRIBUTES or the MODIFY-SUBSYSTEM-ATTRIBUTES statement with entry points that are to be supplied dynamically (*BY-PROGRAM(...)).

Execution of the statement is aborted
– if the subsystem specified by TO-SUBSYSTEM and VERSION is not found in the current SSD object or
– if an existing job entry point is to be defined a second time.
An error message is issued.

Format

```
ADD-SUBSYSTEM-ENTRIES

TO-SUBSYSTEM = <structured-name 1..8>(...)

   <structured-name 1..8>(...)

     |  VERSION = <c-string 3..8> / <text 3..8> “

,SUBSYSTEM-ENTRIES = list-poss(100): <text 1..8>(...)

   <text 1..8>(...)

     |   MODE = *LINK / *ISL(...) / *SVC(...) / *SYSTEM-EXIT(...)
     |
     |      *ISL(...)
     |
     |        |  FUNCTION-NUMBER = *NONE / <integer 0..255>(...)
     |        |
     |        |     <integer 0..255>(...)
     |        |
     |        |       |  FUNCTION-VERSION = <integer 1..255>
     |      *SVC(...)
     |
     |        |  NUMBER = <integer 0..255>
     |        |
     |        | ,CALL-BY-SYSTEM-EXIT = *ALLOWED / *FORBIDDEN
     |        |
     |        | ,FUNCTION-NUMBER = *NONE / <integer 0..255>(...)
     |        |
     |        |     <integer 0..255>(...)
     |        |
     |        |       |  FUNCTION-VERSION = <integer 1..255>
     |
     |      *SYSTEM-EXIT(...)
     |
     |        |  NUMBER = <integer 0..127>
     |
     |  ,CONNECTION-ACCESS = *ALL / *SYSTEM / *SIH
     |
     |  ,CONNECTION-SCOPE = *TASK / *PROGRAM / *FREE / *CALL / *OPTIMAL
     |
     |  , FIRST-CONNECTION = *ALLOWED / *FORBIDDEN
```

Operands

TO-SUBSYSTEM = <structured-name 1..8>(...)
Specifies the name and version of the subsystem for which additional job entry points are to be defined.

VERSION = <c-string 3..8> / <text 3..8>
Specifies the version of the subsystem in the format “[V][n]n.m[ann]”. The text elements have the following meanings:

nn	= Main version (numeric)
m	= Revision version (numeric)
ann	= Update status (a=letter, release status; nn=numeric, correction status)

SUBSYSTEM-ENTRIES =
Declares additional entry points (job entry points) which are to be associated with the subsystem. Up to 100 job entry points can be defined per statement.

SUBSYSTEM-ENTRIES = list-poss(100): <text 1..8>
Specifies up to 100 job entry points by name; the type of each entry point must be defined in the substructures.

MODE =
Defines the type of a job entry which is defined for the subsystem.

MODE = *LINK
Default value: the job entry cannot be accessed by indirect linkage, but only by using a CONNECT relation through an external linkage editor symbol.
In the case of different versions of the same subsystem which use the same external linkage editor symbol, DSSM automatically sets up a link to the highest loaded version of the subsystem.

MODE = *ISL(...)
The job entry is implemented by indirect linkage via System Procedure Linkage (for privileged subsystems only). If the specification includes in addition a function and version number for the ISL entry point, the combination of entry point name, function and version numbers must not match any other combination for the various other subsystems in the catalog or the various versions of the same subsystem (if VERSION-COEXISTENCE=*ALLOWED is specified, see the SET-SUBSYSTEM-ATTRIBUTES statement).
For different subsystems, if the job entry is to be accessed by the same ISL entry point, they must be uniquely identified by specifying the function and version numbers.
In the case of different versions of the same subsystem which use the same ISL entry point, then - if the function and version numbers are not specified - DSSM will automatically set up a connection to the highest loaded version of the subsystem.

In the case of different versions of the same subsystem which use the same ISL entry point and for which the function and version numbers are not equal to *NONE, the version to which the connection is set up will be selected in accordance with the function and version numbers stored in the standard header of the caller’s parameter list.
The specification CONNECTION-ACCESS=*ALL (see the SET-SUBSYSTEM-ATTRIBUTES statement) is not permissible for ISL entry points.

FUNCTION-NUMBER =
Specifies whether a particular function and version number of the ISL entry point is to be addressed, because the same ISL entry point can be used by different functions.

FUNCTION-NUMBER = *NONE
Default value: no particular function or version number is to be addressed.

FUNCTION-NUMBER = <integer 0..255>(...)
Number of the ISL entry point. The version must be nominated in the substructure which follows.

FUNCTION-VERSION = <integer 1..255>
Version of the specified ISL function number.

MODE = *SVC(...)
Job entry is to be effected by an indirect connection using a supervisor call (SVC).
If the specification includes in addition a function and version number for the SVC entry point, the combination of entry point name, function and version numbers must not match any other combination for the various other subsystems in the catalog or the various versions of the same subsystem (if VERSION-COEXISTENCE=*ALLOWED is specified, see the SET-SUBSYSTEM-ATTRIBUTES statement).
For different subsystems, if the job entry is to be accessed by the same SVC, they must be uniquely identified by specifying the function and version numbers.
In the case of different versions of the same subsystem which use the same SVC, then - if the function and version numbers are not specified - DSSM will automatically set up a connection to the highest loaded version of the subsystem.
In the case of different versions of the same subsystem which use the same SVC and for which the function and version numbers are not equal to *NONE, the version to which the connection is set up will be selected in accordance with the function and version numbers stored in the standard header of the caller’s parameter list.
If this operand value is specified, it is better to set the operand CONNECTION-ACCESS to the value *SYSTEM, instead of *ALL (see the SET-SUBSYSTEM-ATTRIBUTES statement).

NUMBER = <integer 0..255>
Number of the SVC via which job entry is to be effected. No SVC number greater than 191 may be used in conjunction with CONNECTION-ACCESS=*ALL (see the SET-SUBSYSTEM-ATTRIBUTES statement).

CALL-BY-SYSTEM-EXIT =
Defines whether the specified SVC number may be called from within system exit routines.

CALL-BY-SYSTEM-EXIT = *ALLOWED
Default value: system exit routines are permitted to call the specified SVC number.

CALL-BY-SYSTEM-EXIT = *FORBIDDEN
System exit routines are not permitted to call the specified SVC number.

FUNCTION-NUMBER =
Specifies whether a particular function and version number of the SVC entry point is to be addressed, because the same SVC entry point can be used by different functions.

FUNCTION-NUMBER = *NONE
Default value: no particular function or version number is to be addressed.

FUNCTION-NUMBER = <integer 0..255>(...)
The number of an SVC entry point. The version must be nominated in the substructure which follows.

FUNCTION-VERSION = <integer 1..255>
Version of the specified SVC function number.

MODE = SYSTEM-EXIT(...)
Job entry is to be effected by an indirect connection using system exit routines.
This operand must not be used in conjunction with CONNECTION-ACCESS=*ALL (see the SET-SUBSYSTEM-ATTRIBUTES statement).

NUMBER = <integer 0..127>
Number of the system exit routine.

CONNECTION-ACCESS =
Specifies the access authorization (privileges) required by the subsystem.

CONNECTION-ACCESS = *ALL
Default value: privileged and nonprivileged program runs may access the subsystem.
This operand value must not be used in conjunction with MODE=*SYSTEM-EXIT/*ISL/*SVC (with an SVC number greater than 191; see the SET-SUBSYSTEM-ATTRIBUTES statement).

CONNECTION-ACCESS = *SYSTEM
Only privileged program runs may access the subsystem.

CONNECTION-ACCESS = *SIH
Only tasks running in the SIH processor state may access the subsystem.
The subsystem called also runs in the SIH processor state, i.e. it is uninterruptible.
This operand value is permissible only for subsystems for which the entry point is defined via:

- System Procedure Linkage (MODE=*ISL(FUNCTION-NUMBER=*NONE))
- CONNECTION-SCOPE=*OPTIMAL
- MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=*SYSTEM)

CONNECTION-SCOPE =
Identifies the event which will call up the automatic cleardown of the connection to the specified subsystem job entry point.

CONNECTION-SCOPE = *TASK
Default value: the connection will be cleared when the task terminates.

CONNECTION-SCOPE = *PROGRAM
The connection will be cleared when the program terminates, or before.
Only CONNECTION-SCOPE=*PROGRAM may be specified in conjunction with MEMORY-CLASS=*LOCAL-UNPRIVILEGED (see the SET-SUBSYSTEM-ATTRIBUTES statement).
This operand value is recommended for subsystems which were declared with SUBSYSTEM-ACCESS=*LOW/*HIGH.

CONNECTION-SCOPE = *FREE
DSSM is not to carry out any checking of the connections to the job entry point. The connection will not be automatically cleared - unless explicitly requested. To avoid problems or possible errors when the subsystem is being unloaded, the connections must be managed by the subsystem itself.

CONNECTION-SCOPE = *CALL
On return from this job entry point, DSSM will automatically clear the connections.
This operand value is only available with subsystems for which the job entry is defined by means of System Procedure Linkage (ISL) or supervisor calls (SVC).

CONNECTION-SCOPE = *OPTIMAL
The subsystem is deactivated or suspended when there are no further tasks with a connection to this job entry point.
A routine with an entry point defined with *OPTIMAL must be terminated with RETURN.
If an entry point of a subsystem is defined with CONNECTION-SCOPE=*OPTIMAL, all of its entry points should be defined in the subsystem catalog with MODE≠*LINK.
While a subsystem is deactivated or suspended, no call of the subsystem with CONNECTION-SCOPE=*OPTIMAL is accepted.
Exception: if the subsystem was defined with CREATION-TIME=*AT-SUBSYSTEM-CALL and the calling task is already connected to the subsystem (see the SET-SUBSYSTEM-ATTRIBUTES statement).

FIRST-CONNECTION =
Determines whether or not first connection of the task to the specified job entry point in the subsystem is allowed. At least one job entry point of a subsystem must be defined with FIRST-CONNECTION=*ALLOWED.

FIRST-CONNECTION = <u>*ALLOWED</u>
Default setting: first connection to the specified job entry point is allowed.

FIRST-CONNECTION = *FORBIDDEN
Connection to the specified job entry point via SVC or ISL is not allowed if the task has not yet been connected to another job entry point belonging to the subsystem.
It is not permitted to specify this operand value for job entry points that have been defined with MODE=*LINK/*SYSTEM-EXIT or CONNECTION-ACCESS=*SIH.

ASSIGN-HOLDER-TASK
Distribute subsystems to holder tasks

Function

This statement is used to control the distribution of subsystems to holder tasks. The subsystems listed in the statement can use the holder task as a work task, or will be set up together in a holder task. If this statement is omitted when creating a catalog, SSCM will make a standard assignment of the subsystems in order to limit the number of holder tasks.

This statement is mandatory for subsystems which use the holder task as a work task.

In order to become a work task, the entry points of the subsystem must be defined with one of the following combinations:

CLOSE-CTRL	STOPCOM	DEINIT
*DYNAMIC	*DYNAMIC	*DYNAMIC
*DYNAMIC	*NO	*DYNAMIC
*NO	*DYNAMIC	*DYNAMIC *
*NO	*NO	*DYNAMIC *

* For compatibility reasons it is not obligatory to define a DEINIT routine for a subsystem without a CLOSE-CTRL routine. It should be noted, however, that no guarantee can be given for correct execution of the subsystem in such a case.

The statement may only be used once for an SSD object. If there are two consecutive statements for the same subsystem and there is a conflict (i.e. contradictory declarations), the definition which requires the holder task to be used as a work task will take precedence. A declaration referring to a shared holder task applies to every version of the subsystem, except for any versions which use the holder task as a work task.

ASSIGN-HOLDER-TASK is rejected if none of the following statements has been executed beforehand:

– START-SSD-CREATION
– START-CATALOG-CREATION
– START-CATALOG-MODIFICATION

Format

```
ASSIGN-HOLDER-TASK

TYPE = *WORK-TASK (...) / *SHARED-HOLDER(...)
  *WORK-TASK(...)
    |  SUBSYSTEM-NAME = <structured-name 1..8>
    | ,SUBSYSTEM-VERSION = <c-string 3..8> / <text 3..8>
    | ,TSN = *BY-DSSM / <alphanum-name 1..4>
  *SHARED-HOLDER(...)
    |  BY-SUBSYSTEMS = list-poss(15): <structured-name 1..8>
    | ,TSN = *BY-DSSM / <alphanum-name 1..4>
```

Operands

TYPE =
Specifies whether the holder task is to be used as a work task or the subsystem is to be created in a shared holder task.

TYPE = *WORK-TASK(...)
Default value: the holder task is to be used as a work task.

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem which is to use the holder task as a work task.

SUBSYSTEM-VERSION = <c-string 3..8> / <text 3..8>
Version of the subsystem which is to use the holder task as a work task.
This version must have been declared previously.

TSN =
Specifies the task sequence number (TSN) to be given to the subsystem’s work task.

TSN = *BY-DSSM
Default value: the TSN will be issued by DSSM when the work task is loaded.

TSN = <alphanum-name 1..4>
The TSN to be given to the work task when it is started.
The specified TSN must be uniquely defined and usable when the subsystem is loaded.

TYPE = *SHARED-HOLDER(...)
The subsystem is to be created in a shared holder task.

BY-SUBSYSTEMS = list-poss(15): <structured-name 1..8>
Names of up to 15 subsystems which are to be created in the same holder task. The first of the subsystems in the list must have been declared previously.

TSN =
Specifies the task sequence number (TSN) to be given to the shared holder task.

TSN = <u>*BY-DSSM</u>
Default value: the TSN will be issued by DSSM when the work task is loaded.

TSN = <alphanum-name 1..4>
Task sequence number to be given to the shared holder task when it is started. The specified TSN must be uniquely defined and usable when the subsystem is loaded.

CHECK-CATALOG
Check subsystem definition(s) for consistency

Function

This statement is used to carry out consistency checks on the definitions of subsystems held in a catalog.

CHECK-CATALOG will be rejected if the file name specified by the user does not exist, or if the subsystem catalog is empty.

If the specified file name does not correspond with that of the catalog which is currently open, the following message will be output:

```
SCM0011   DO YOU REALLY WANT TO OVERWRITE MEMORY CATALOG '(&00)'? REPLY (Y/N)
```

If the user replies with **Y**, the virtual definitions in the current catalog will be lost. If the reply is **N**, execution of the CHECK-CATALOG statement will be aborted, and the user can then use the SAVE-CATALOG statement to save to a file all subsystem definitions which have not yet been saved.

i The catalog cannot be saved without first carrying out checks on any link and dependency relations.

Format

CHECK-CATALOG
CATALOG-NAME = ***CURRENT** / <filename 1..54 without-gen-vers>
,**DEPENDENCE-RELATION** = ***YES** / ***NO**
,**LINK-RELATION** = ***YES** / ***NO**
,**RELATED-FILES** = ***NO** / ***YES**
,**OUTPUT** = ***SYSOUT** / ***SYSLST**(...)
***SYSLST**(...)
| **SYSLST-NUMBER** = ***STD** / <integer 1..99>

Operands

CATALOG-NAME =
Specifies the name of the catalog which holds the definitions which are to be checked.

CATALOG-NAME = *CURRENT
Default value: the catalog which is currently open (START-CATALOG-CREATION or START-CATALOG-MODIFICATION statement) is to be checked.

CATALOG-NAME = <filename 1..54 without-gen-vers>
Fully qualified name of the static subsystem catalog whose contents are to be checked.

DEPENDENCE-RELATION = *YES / *NO
Specifies whether the checks on subsystem definitions are also to take into account dependency relations to other subsystems (*YES) or not (*NO). The catalog cannot be saved if *NO was specified in a prior CHECK-CATALOG statement.

LINK-RELATION = *YES / *NO
Specifies whether the checks on subsystem definitions are also to take into account address links to other subsystems (*YES, default value) or not (*NO). The catalog cannot be saved if *NO was specified in a prior CHECK-CATALOG statement.

RELATED-FILES = *NO / *YES
Specifies whether the existence of files which have dependency relations to these subsystems is to be checked (*YES) or not (*NO, default value).
If the names of dependent files were defined with the value *INSTALLED(...), the DEFAULT-NAME specified there will also be checked.

OUTPUT =
Specifies where to output the information generated by the statement, i.e. the results of the check run.

OUTPUT = *SYSOUT
Default value: the messages will be output to the terminal.

OUTPUT = *SYSLST(...)
The messages are to be output to SYSLST.

SYSLST-NUMBER =
Identifies the SYSLST file to which the output is to be directed.

SYSLST-NUMBER = *STD
Default value: output is to go to the default system file SYSLST.

SYSLST-NUMBER = <integer 1..99>
Output is to go to one of the system files from the set SYSLST01 to SYSLST99, the number of which must be specified here.

GENERATE-CATALOG-SOURCE Create SSCM statement list for generation

Function

With this statement, SSCM creates a file containing a list of all SSCM statements required for (re)generation of a subsystem catalog (either for specific subsystems in the input catalog or for all of them).

Format

GENERATE-CATALOG-SOURCE
CATALOG-NAME = ***CURR**ENT / <filename 1..54 without-gen-vers>
,SUBSYSTEM-NAME = ***ALL** / <structured-name 1..8>(...)
<structured-name 1..8>(...)
⏐ VERSION = ***ALL** / <c-string 3..8> / <text 3..8>
,OUTPUT = ***SYSLST**(...)
***SYSLST**(...)
⏐ SYSLST-NUMBER = ***STD** / <integer 1..99>

Operands

CATALOG-NAME =
Specifies the subsystem catalog in which the subsystem definitions are saved.

CATALOG-NAME = *CURRENT
Default setting. The current subsystem catalog is used.

CATALOG-NAME = <filename 1..54 without-gen-vers>
Name of the subsystem catalog.

SUBSYSTEM-NAME =
Subsystems whose definitions are to be output.

SUBSYSTEM-NAME = *ALL
Default setting. The definitions of all subsystems are to be output.

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Name of the subsystem whose definition is to be output.

VERSION = *ALL / <c-string 3..8> / <text 3..8>
Version of the subsystem whose definition is to be output.

OUTPUT = *SYSLST(...)
System file to which the generated information is to be sent.

SYSLST-NUMBER = *STD
Default setting. The information will be sent to the SYSLST file.

SYSLST-NUMBER = <integer 1..99>
The information will be sent to the specified system file in the range SYSLST01 to SYSLST99.

MODIFY-SUBSYSTEM-ATTRIBUTES Modify subsystem attributes

Function

This statement can be used to change any of the attributes and entry points which were defined in the SET-SUBSYSTEM-ATTRIBUTES statement.

When a definition is modified, the following points must be noted:

– the subsystem - identified by its name and version - must be present in the catalog which is currently open
– any attempt to add a job entry or relation which already exists will be rejected
– it is impermissible to attempt to modify or delete a job entry or relation which has not yet been defined
– the mode of a job entry may only be changed if all the parameters are specified; the default values *UNCHANGED will be rejected
– the memory class for a subsystem may only be changed if all the parameters are specified; the default values *UNCHANGED will be rejected

MODIFY-SUBSYSTEM-ATTRIBUTES is rejected if none of the following statements have been executed beforehand:
– START-CATALOG-CREATION
– START-CATALOG-MODIFICATION

Note on syntax

A special data type <symbol>, which is fully described in the “BLSSERV” manual [4], can also be used for the names of the entry points in the following operands (in the format, the data type is specified as <name>):

– LINK-ENTRY
– DYNAMIC-CHECK-ENTRY
– INIT-ROUTINE
– CLOSE-CTRL-ROUTINE
– STOPCOM-ROUTINE
– DEINIT-ROUTINE
– INTERFACE-VERSION
– ADD-SUBS-ENTRIES
– MODIFY-SUBS-ENTRIES
– REMOVE-SUBS-ENTRIES

Format

(part 1 of 4)

MODIFY-SUBSYSTEM-ATTRIBUTES
SUBSYSTEM-NAME = <structured-name 1..8>(...)
<structured-name 1..8>(...)
| **VERSION** = <c-string 3..8> / <text 3..8>
,**INSTALL**ATION-**UNIT** = ***UNCHA**NGED / ***NONE** / ***STD** / <text 1..30>
,**INSTALL**ATION-**USERID** = ***UNCHA**NGED / ***NONE** / ***DEFA**ULT-**USERID** / <name 1..8>
,**COPYRIGHT** = ***UNCHA**NGED / ***NONE** / <c-string 1..54>(...)
<c-string 1..54>(...)
| **YEAR** = ***YEAR-1990** / <c-string 4..4>
,**LIB**RARY = ***UNCHA**NGED / ***STD** / ***CPLINK** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>
***INSTALLED**(...)
| **LOGICAL-ID** = ***UNCHA**NGED / <filename 1..30 without-catid-userid-gen-vers>
| ,**DEFA**ULT-**NAME** = ***UNCHA**NGED / <filename 1..54 without-gen-vers>
,**SUBSYS**TEM-**LOAD-MODE** = ***UNCHA**NGED / ***STD** / ***ADVANCED**
,**REP-FILE** = ***UNCHA**NGED / ***STD** / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>
***INSTALLED**(...)
| **LOGICAL-ID** = ***UNCHA**NGED / <filename 1..30 without-catid-userid-gen-vers>
| ,**DEFA**ULT-**NAME** = ***UNCHA**NGED / <filename 1..54 without-gen-vers> / ***NONE**
,**REP-FILE-MANDATORY** = ***UNCHA**NGED / ***NO** / ***Y**ES
,**MES**SAGE-**FILE** = ***UNCHA**NGED / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>
***INSTALLED**(...)
| **LOGICAL-ID** = ***UNCHA**NGED / <filename 1..30 without-catid-userid-gen-vers>
| ,**DEFA**ULT-**NAME** = ***UNCHA**NGED / <filename 1..54 without-gen-vers> / ***NONE**
,**SUBSYS**TEM-**INFO-FILE** = ***UNCHA**NGED / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>
***INSTALLED**(...)
| **LOGICAL-ID** = ***UNCHA**NGED / <filename 1..30 without-catid-userid-gen-vers>
| ,**DEFA**ULT-**NAME** = ***UNCHA**NGED / <filename 1..54 without-gen-vers> / ***NONE**
,**SYNTAX-F**ILE = ***UNCHA**NGED / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>
***INSTALLED**(...)
| **LOGICAL-ID** = ***UNCHA**NGED / <filename 1..30 without-catid-userid-gen-vers>
| ,**DEFA**ULT-**NAME** = ***UNCHA**NGED / <filename 1..54 without-gen-vers> / ***NONE**

continued ➠

(part 2 of 4)

,**DYNAMIC-CHECK-ENTRY** = ***UNCHA**NGED / ***STD** / ***NO** / <text 1..8 without-sep>

,**CRE**ATION-**TIME** = ***UNCHA**NGED / ***AT-CRE**ATION-**REQ**UEST / ***AT-SUBSYS**TEM-**CALL**(...) / ***AT-DSSM-LOAD** / ***BEFORE-DSSM-LOAD** / ***MANDATORY-AT-STARTUP** / ***BEFORE-SYSTEM-READY** / ***AFTER-SYSTEM-READY**

***AT-SUBSYS**TEM-**CALL**(...)

ON-ACTION = ***STD** / ***ISL-CALL** / *ANY

,**INIT-ROUTINE** = ***UNCHA**NGED / ***NO** / <text 1..8 without-sep>

,**CLOSE-CTRL-ROUTINE** = ***UNCHA**NGED / ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOPCOM-ROUTINE** = ***UNCHA**NGED / ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**DEINIT-ROUTINE** = ***UNCHA**NGED / ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOP-AT-SHUTDOWN** = ***UNCHA**NGED / ***NO** / ***Y**ES

,**INTERFACE-VERSION** = ***UNCHA**NGED / ***NO** / <text 1..8 without-sep>

,**SUBSYS**TEM-**HOLD** = ***UNCHA**NGED / ***ALLOW**ED / ***FORBID**DEN

,**STATE-CHANGE-CMDS** = ***UNCHA**NGED / ***ALLOW**ED / ***FORBID**DEN / ***BY-ADMINISTRATOR-ONLY**

,**FORCED-STATE-CHANGE** = ***UNCHA**NGED / ***ALLOW**ED / ***FORBID**DEN

,**RESET** = ***UNCHA**NGED / ***ALLOW**ED / ***FORBID**DEN

,**RESTART-REQUIRED** = ***UNCHA**NGED / ***NO** / ***Y**ES

,**VERSION-COEXISTENCE** = ***UNCHA**NGED / ***FORBID**DEN / ***ALLOW**ED

,**VERSION-EXCHANGE** = ***UNCHA**NGED / ***FORBID**DEN / ***ALLOW**ED

,**ADD-SUBS-ENTRIES** = ***NONE** / list-poss(100): <text 1..8>(...)

<text 1..8>(...)

MODE = ***LINK** / ***ISL**(...) / ***SVC**(...) / ***SYSTEM-EXIT**(...)

***ISL**(...)

FUNCTION-NUMBER = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SVC**(...)

NUMBER = <integer 0..255>
,**CALL-BY-SYSTEM-EXIT** = ***ALLOW**ED / ***FORBIDDEN**
,**FUNCTION-NUM**BER = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

continued ➠

(part 3 of 4)

```
        |     *SYSTEM-EXIT(...)
        |
        |        |  NUMBER = <integer 0..127>
        |
        |  ,CONNECTION-ACCESS = *ALL / *SYSTEM / *SIH
        |
        |  ,CONNECTION-SCOPE = *TASK / *PROGRAM / *FREE / *CALL / *OPTIMAL
        |
        |  ,FIRST-CONNECTION = *ALLOWED / *FORBIDDEN

,MODIFY-SUBS-ENTRIES = *NONE / list-poss(100): <text 1..8>(...) / *BY-PROGRAM(...)

     <text 1..8>(...)

        |  MODE = *UNCHANGED / *LINK / *ISL(...) / *SVC(...) / *SYSTEM-EXIT(...)
        |
        |     *ISL(...)
        |
        |        |  FUNCTION-NUMBER = *UNCHANGED (...) / *NONE / <integer 0..255>(...)
        |        |
        |        |     *UNCHANGED(...)
        |        |        |  FUNCTION-VERSION = *UNCHANGED / <integer 1..255>
        |        |
        |        |     <integer 0..255>(...)
        |        |        |  FUNCTION-VERSION = <integer 1..255>
        |
        |     *SVC(...)
        |
        |        |  NUMBER = *UNCHANGED / <integer 0..255>
        |        | ,CALL-BY-SYSTEM-EXIT = *UNCHANGED / *ALLOWED / *FORBIDDEN
        |        | ,FUNCTION-NUMBER = *UNCHANGED (...) / *NONE / <integer 0..255>(...)
        |        |
        |        |     *UNCHANGED(...)
        |        |        |  FUNCTION-VERSION = *UNCHANGED / <integer 1..255>
        |        |
        |        |     <integer 0..255>(...)
        |        |        |  FUNCTION-VERSION = <integer 1..255>
        |
        |     *SYSTEM-EXIT(...)
        |
        |        |  NUMBER = <integer 0..127>
        |
        |  ,CONNECTION-ACCESS = *UNCHANGED / *ALL / *SYSTEM / *SIH
        |
        |  ,CONNECTION-SCOPE = *UNCHANGED / *TASK / *PROGRAM / *FREE / *CALL / *OPTIMAL
        |
        |  ,FIRST-CONNECTION = *UNCHANGED /  *ALLOWED / *FORBIDDEN

     *BY-PROGRAM(...)

        |  CONNECTION-SCOPE = *UNCHANGED / *TASK / *PROGRAM

,REMOVE-SUBS-ENTRIES = *NONE / list-poss(100): <text 1..8>

,MEMORY-CLASS = *UNCHANGED / *SYSTEM-GLOBAL(...) / *LOCAL-PRIVILEGED(...) /
        *LOCAL-UNPRIVILEGED(...) / *BY-SLICE(...)
```

continued ➠

(part 4 of 4)

```
    *SYSTEM-GLOBAL(...)
      |  SUBSYSTEM-ACCESS = *LOW / *SYSTEM / *HIGH
    *LOCAL-PRIVILEGED(...)
      |  SIZE = <integer 1..32767>
    *LOCAL-UNPRIVILEGED(...)
      |  SIZE = *UNCHANGED / <integer 1..32767>
      | ,SUBSYSTEM-ACCESS = *UNCHANGED / *LOW / *HIGH
      | ,START-ADDRESS = *UNCHANGED / *ANY / <x-string 7..8>
    *BY-SLICE(...)
      |  SIZE = <integer 1..32767>
,LINK-ENTRY = *UNCHANGED / <text 1..8 without-sep>(...)
    <text 1..8 without-sep>(...)
      |  AUTOLINK = *ALLOWED / *FORBIDDEN
,ADD-REFER-SUBS = *NONE / list-poss(15): <structured-name 1..8>(...)
    <structured-name 1..8>(...)
      |  LOWEST-VERSION = *LOWEST-EXISTING / <c-string 3..8> / <text 3..8>
      | ,HIGHEST-VERSION = *HIGHEST-EXISTING / <c-string 3..8> / <text 3..8>
,MODIFY-REFER-SUBS = *NONE / list-poss(15): <structured-name 1..8>(...)
    <structured-name 1..8>(...)
      |  LOWEST-VERSION = *UNCHANGED / *LOWEST-EXISTING / <c-string 3..8> / <text 3..8>
      | ,HIGHEST-VERSION = *UNCHANGED / *HIGHEST-EXISTING / <c-string 3..8> / <text 3..8>
,REMOVE-REFER-SUBS = *NONE / list-poss(15): <structured-name 1..8>
,UNRESOLVED-EXTERNALS = *UNCHANGED / *ALLOWED / *FORBIDDEN
,CHECK-REFERENCE = *UNCHANGED / *YES / *NO
,ADD-RELATED-SUBS = *NONE / list-poss(100): <structured-name 1..8>(...)
    <structured-name 1..8>(...)
      |  LOWEST-VERSION = *LOWEST-EXISTING / <c-string 3..8> / <text 3..8>
      | ,HIGHEST-VERSION = *HIGHEST-EXISTING / <c-string 3..8> / <text 3..8>
,MODIFY-RELATED-SUBS = *NONE / list-poss(100): <structured-name 1..8>(...)
    <structured-name 1..8>(...)
      |  LOWEST-VERSION = *UNCHANGED / *LOWEST-EXISTING / <c-string 3..8> / <text 3..8>
      | ,HIGHEST-VERSION = *UNCHANGED / *HIGHEST-EXISTING / <c-string 3..8> / <text 3..8>
,REMOVE-RELATED-SUBS = *NONE / list-poss(100): <structured-name 1..8>
```

Operands

In each case, the default value *UNCHANGED means that the value set in the SET-SUBSYSTEM-ATTRIBUTES statement is to remain valid.
If the type of job entry point declared (MODE operand) or the subsystem-specific address space (MEMORY operand) is changed, all the suboperands of MODE or MEMORY must be assigned a value explicitly, i.e. the operand value *UNCHANGED (default value) will be rejected.

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Specifies the name and version of the subsystem whose attributes are to be changed.

VERSION = <c-string> / <text 3..8>
The version of the subsystem must be specified in the format “[V][n]n.m[ann]”. The text elements have the following meanings:

nn	= Main version (numeric)
m	= Revision version (numeric)
ann	= Update status (a=letter, release status; nn=numeric, correction status)

INSTALLATION-UNIT =
Defines the name of the installed software unit. A value other than *NONE must be specified for all subsystems installed with IMON, and likewise if the value *INSTALLED(LOGICAL-ID=...) was defined for the operands SUBSYSTEM-LIBRARY, REP-FILE, SUBSYSTEM-INFO-FILE, MESSAGE-FILE and SYNTAX-FILE.
The syntax rules described in the “IMON“ manual [17] must be observed when defining the name.

INSTALLATION-UNIT = *NONE
No name is assigned. This entry is not allowed for any subsystems installed with IMON.

INSTALLATION-UNIT = *STD
The name specified with the SUBSYSTEM-NAME operand is used as the new name of the installed software unit.

INSTALLATION-UNIT = <text 1..30>
New name of the installed software unit.

INSTALLATION-USERID =
Specifies a user ID under which the relevant DSSM task expects the subsystem satellites (REP file, object module library, message file, syntax file and subsystem information file) if they have not yet been assigned to a user ID, in other words if the file name was specified without a user ID.

INSTALLATION-USERID = *NONE
The files will not be expected under a specific user ID.

INSTALLATION-USERID = *DEFAULT-USERID
The files will be expected under the default user ID (prefix “$.”) or, if the subsystem is a local subsystem, under the user ID of the calling task.

INSTALLATION-USERID = <name 1..8>
User ID under which the subsystem satellites are to be expected.
If this statement applies to an SSD object, the files will only actually be expected under the user ID specified here if no user ID was specified in the ADD-CATALOG-ENTRY statement (inclusion of the subsystem definitions from the SSD object in the catalog, see page 183). The ID specified in ADD-CATALOG-ENTRY takes precedence.

COPYRIGHT =
Specifies whether or not a copyright notice is to be displayed when the subsystem is started and, if so, which one.

COPYRIGHT = *NONE
No copyright notice is to be output.

COPYRIGHT = <c-string 1..54>(...)
Text of the copyright notice which is to be output together with the creation date when the subsystem is started.

YEAR = *YEAR-1990 / <c-string 4..4>
Number of the year which is to appear in the copyright notice as the creation date. This is not subjected to a semantic check.

LIBRARY =
Specifies a new name for the program or object module library (OML) from which the object code for the subsystem is to be loaded when it is activated.

LIBRARY = *STD
When the subsystem is started, the object code will automatically be loaded from the library SYSLNK.<subsysname>.<subsysvers#>. This library is stored under the user ID under which the holder task is running. For local subsystems this is the user ID of the caller, and for global subsystems it is TSOS.
“<subsysvers#>” is a three-character value consisting of the elements “mmm” specified for the operand SUBSYSTEM-NAME=...(VERSION=...).

LIBRARY = *CPLINK
The subsystem which is to be defined is linked to the BS2000/OSD control program (CP) and must have been loaded before DSSM was activated. This operand may be used only in conjunction with the operand CREATION-TIME=*BEFORE-DSSM-LOAD.

LIBRARY = *INSTALLED(...)
The library name must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the program library or object module library.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
Library name if IMON-GPN is not available or if the logical ID is unknown.

LIBRARY = <filename 1..54 without-gen-vers>
Fully qualified file name of the object module library from which the object code for the subsystem is to be loaded.

SUBSYSTEM-LOAD-MODE =
Specifies the load mode of the subsystem (via the BLS-DSSM interface $PBBND1).

SUBSYSTEM-LOAD-MODE = *STD
The BLS (Binder-Loader-Starter system) is called up in STD run mode and loads the subsystem as an object module.

SUBSYSTEM-LOAD-MODE = *ADVANCED
The BLS is called up in ADVANCED run mode and loads the subsystem as a link and load module (LLM).

REP-FILE =
Specifies whether or not system REPs are required for the subsystem which is being defined, and identifies the file in which they are stored. These correction statements are used during activation of the subsystem, and are applied solely to the modules stored and loaded in the object module library, not to other subsystems or the BS2000/OSD control program (CP). A REP file can also be specified for modules of a nonprivileged subsystem. REP-FILE must not be specified together with LIBRARY=*CPLINK.

REP-FILE = *STD
By default, system REPs are loaded from the file SYSREP.<subsysname>.<subsysvers#>. This library is stored under the user ID under which the holder task is running. For local subsystems this is the user ID of the caller, and for global subsystems it is TSOS. “<subsysvers#>” is a three-character value consisting of the elements “mmm” specified for the operand SUBSYSTEM-NAME=...(VERSION=...).

REP-FILE = *NO
No REP files are to be processed for the subsystem.

REP-FILE = *INSTALLED(...)
The name of the REP file must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the REP file.

DEFAULT-NAME =
Name of the REP file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
No new name is assigned.

REP-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the REP file from which the correction statements are to be read.

REP-FILE-MANDATORY =
Specifies whether or not a REP file declared via the REP-FILE operand is to be processed when loading the subsystem.

REP-FILE-MANDATORY = *NO
The use of a REP file is not mandatory, i.e. neither the REP file nor its entries are to be checked when the subsystem is activated. Even if the REP file cannot be accessed or if individual correction statements are invalid, the subsystem will still be started.

REP-FILE-MANDATORY = *YES
If any of the following errors occurs during processing of the REP file, the attempt to load the subsystem will be terminated:
- the REP file is not cataloged, or it cannot be read
- checking of the correction statements reveals an error
- the name of a correction statement is invalid
- DMS reports an error during access to the NOREF file (this file is used during loading of a subsystem to prevent invalid system REPs from being logged at the operator terminal)

MESSAGE-FILE =
Specifies whether there is a subsystem-specific message file which is to be automatically activated when the subsystem is loaded.
For subsystems which are defined with the creation time AT-DSSM-LOAD, a dependency relation to the MIP subsystem must be specified in the RELATED-SUBSYSTEM operand.

MESSAGE-FILE = *NO
No message file is to be activated. This value is mandatory for all subsystems which are defined with the creation time BEFORE-DSSM-LOAD (see also the CREATION-TIME operand), as it is not possible to activate a message file as early as this.

MESSAGE-FILE = *INSTALLED(...)
The name of the message file must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the message file.

DEFAULT-NAME =
Name of the message file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
No new name is assigned.

MESSAGE-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the message file. This will be automatically activated when the subsystem is loaded (START-SUBSYSTEM command) and automatically deactivated when it is unloaded (STOP-SUBSYSTEM).

SUBSYSTEM-INFO-FILE =
Specifies whether or not a subsystem information file (SSINFO) is available. This file contains subsystem-specific data (subsystem satellites and configuration data) which cannot be processed centrally by DSSM.

SUBSYSTEM-INFO-FILE = *NO
No information file is available for the subsystem.

SUBSYSTEM-INFO-FILE = *INSTALLED(...)
The name of the information file must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the information file.

DEFAULT-NAME =
Name of the information file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
A new name is assigned.

SUBSYSTEM-INFO-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the information file. This name is automatically passed to the activation and deactivation routines (INIT-/DEINIT-/CLOSE-CTRL-ROUTINE operands) when they are called.

SYNTAX-FILE =
Specifies whether the subsystem has linked to it a syntax file which will be activated automatically when the subsystem is loaded. For subsystems which are defined with the start attribute MANDATORY-AT-STARTUP, a dependency relation to the SDF subsystem must be declared in the REFERENCED-SUBSYSTEM operand.

SYNTAX-FILE = *NO
No syntax file is to be activated. This value is mandatory for all subsystems which are defined with the start time BEFORE-DSSM-LOAD or AT-DSSM-LOAD (see also the CREATION-TIME operand), as it is not possible to activate a syntax file as early as this.

SYNTAX-FILE = *INSTALLED(...)
The name of the syntax file must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the syntax file.

DEFAULT-NAME =
Name of the syntax file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
No new name is assigned.

SYNTAX-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the syntax file which is to be automatically activated when the subsystem is loaded.

DYNAMIC-CHECK-ENTRY =
Specifies whether a dynamic identity check is to be carried out on the subsystem. For this purpose, an entry point must be specified, at which both the subsystem name (eight characters) and also the version number (four or seven characters) must be located. DSSM checks whether the identification specified in the definition agrees with the subsystem which is loaded.

DYNAMIC-CHECK-ENTRY = *STD
The entry point specified in the LINK-ENTRY operand is to be applied in carrying out the identity check.

DYNAMIC-CHECK-ENTRY = *NO
No check is to be carried out. However, this value of the operand must not be used for any subsystems which are loaded before DSSM is activated (CREATION-TIME=*BEFORE-DSSM-LOAD).

DYNAMIC-CHECK-ENTRY = <text 1..8 without-sep>
Name of the entry point which is to be used in applying the identity check.

CREATION-TIME =
Specifies the point in time at which activation of the subsystem (CREATE routine) is initiated. There are two separate phases during system initialization when DSSM takes over the control of system initialization after it has been called by the startup routine:

Phase 1: The DSSM code is loaded, the DSSM task is generated and started. This task reserves class 5 memory, reads in the subsystem catalog, and starts those subsystems which have been defined with the start attributes BEFORE-DSSM-LOAD and AT-DSSM-LOAD. After these subsystems have been loaded, control of system initialization returns to the startup routine.

Phase 2: Following a second call, all those subsystems are loaded which have been defined with the start attributes MANDATORY-AT-STARTUP, BEFORE-SYSTEM-READY and AFTER-SYSTEM-READY.
In the case of the first two of these start attributes, loading of the subsystems is synchronized with the start routine (i.e. loading must be completed), but for the last of them asynchronous loading is initiated.
Control of system initialization returns to the startup routine.

If different versions of a subsystem are to be defined, it is only possible to specify the start attributes, for use in phases 1 and 2 of system initialization, for <u>one</u> of these versions.

CREATION-TIME = *AT-CREATION-REQUEST
The subsystem must be explicitly loaded by means of the START-SUBSYSTEM command.

CREATION-TIME = *AT-SUBSYSTEM-CALL(...)
The subsystem is to be automatically loaded when the first SVC or ISL call is made. This operand value is reserved for subsystems which are called via SVC or ISL.
If two or more versions of a subsystem are defined with this operand value, VERSION-COEXISTENCE=*ALLOWED must be specified for all of these versions and FUNCTION-NUMBER and FUNCTION-VERSION must be specified for their SVC or ISL entry points, which were declared with CONNECTION-ACCESS with a value other than *SIH.
At least one of the specified subsystems must have been declared with SUBSYSTEM-ENTRIES ..., MODE=*SVC or *ISL (corresponding to the value of the ON-ACTION operand).

ON-ACTION =
Determines what initiates automatic loading of the subsystem.

ON-ACTION = <u>*STD</u>
Default setting: loading begins when any SVC entry point belonging to the subsystem is called.

ON-ACTION = *ISL-CALL
Loading begins when any ISL entry point belonging to the subsystem is called.

ON-ACTION = *ANY
Loading begins when any SVC or ISL entry point belonging to the subsystem is called.

CREATION-TIME = *AT-DSSM-LOAD
The subsystem is to be loaded during system initialization (phase 1) under the control of the DSSM task.
The subsystem must be a privileged one, and may only have address or dependency relations to subsystems which are also defined with this start attribute or with the start attribute BEFORE-DSSM-LOAD.
The files for this subsystem must be held on the home pubset under the TSOS user ID, because at the time of startup the user catalog is not accessible and IMPORT-PUBSET processing has not been completed.
It is not permissible to link in a syntax file for these subsystems.

CREATION-TIME = *BEFORE-DSSM-LOAD
The subsystem is to be loaded during system initialization (phase 1), but not under the control of the DSSM task.
Such subsystems are linked to the control program, and do not need to be synchronized with the DSSM task when activated. However, after the subsystem is loaded it again runs under the control of DSSM and can, from the user’s point of view, be controlled in the same way as other subsystems.
No address or dependency relations are possible to subsystems which are defined with any other start attribute. Nor is it permissible to link in a message or syntax file. All job entries (SUBSYSTEM-ENTRIES operand) must be declared, because DSSM creates the (privileged) connection to these job entries. Responsibility for ensuring that at least one version of this subsystem is available at any given time rests entirely with the subsystem developer.
The name of the link context for these subsystems must be unique, because DSSM must also honor an unload request even if the subsystem code has not been loaded. An entry point (DYNAMIC-CHECK-ENTRY operand) must have been specified.

CREATION-TIME = *BEFORE-SYSTEM-READY
The subsystem is to be loaded during system initialization (phase 2). Activation is initiated synchronously; not until loading is complete (or a load error occurs) does control return to the startup routine, which can then report “SYSTEM READY”.
The subsystem must be privileged, and may only have address or dependency relations to subsystems defined with the same attribute or with the start attribute BEFORE-DSSM-LOAD, AT-DSSM-LOAD or MANDATORY-AT-STARTUP.
The files for this subsystem must be cataloged on the home pubset.
If a nonprivileged subsystem is declared with this operand value, it is automatically assigned the value *AFTER-SYSTEM-READY. SSCM issues a message.

CREATION-TIME = *MANDATORY-AT-STARTUP
The subsystem must be loaded during system initialization (phase 2). Activation is initiated synchronously - as in the case of BEFORE-SYSTEM-READY. By contrast with the latter, however, loading of the subsystem must in this case be completed **successfully**. Otherwise a message is passed to the startup routine reporting that a mandatory subsystem could not be loaded. In this case, the startup routine will decide whether processing should continue or be terminated.
The subsystem must be privileged, and may only have address or dependency relations to subsystems defined with the same attribute or with the start attribute BEFORE-DSSM-LOAD or AT-DSSM-LOAD. The files for this subsystem must be cataloged on the home pubset.

CREATION-TIME = *AFTER-SYSTEM-READY
The loading of this subsystem is to be initiated during system initialization (phase 2). Execution of this routine is not synchronized with the startup routine, which can report “SYSTEM READY” before subsystem loading is complete.
The subsystem may only have address or dependency relations to subsystems defined with the same attribute or with the start attribute BEFORE-DSSM-LOAD, AT-DSSM-LOAD, MANDATORY-AT-STARTUP or BEFORE-SYSTEM-READY. The files for this subsystem must be cataloged on the home pubset.

INIT-ROUTINE =
Specifies whether there is an initialization routine for the subsystem which must be performed when it is started or resumed. In this case, the name of an entry point must have been declared, and DSSM will delegate initialization to the holder task of the subsystem concerned. It is strongly recommended that an entry point be defined for all subsystems which have the start attribute BEFORE-DSSM-LOAD. During loading of the subsystem, (i.e. when the initialization routine is carried out) the subsystem is then informed that DSSM can assume control over the opening and closing of relations.

INIT-ROUTINE = *NO
No initialization routine is to be performed.

INIT-ROUTINE = <text 1..8 without-sep>
Name of the entry point to the initialization routine.
In the holder task, control is passed to the initialization routine, so that the subsystem can initialize itself. To permit this, it is passed:
– the name and the version of the subsystem, as defined in SSMCAT
– the name of the SSINFO file, if one was specified in the SUBSYSTEM-INFO-FILE operand
– the address of the entry point specified during loading and linking (LINK-ENTRY operand)
– the link context name used by the dynamic binder loader (DBL)
– the name of the memory pool (for subsystems in class 5 or class 6 memory), in order that the subsystem can refer to its own selectable units/load units during dynamic loading
– the name of the message file
– the address of the SUBSYSTEM-PARAMETER operand, if a string has been specified in the START-SUBSYSTEM command

At the end of initialization, a return message is expected from the subsystem, indicating whether initialization has been successful and whether the holder task is to be used as a work task (as specified in the ASSIGN-HOLDER-TASK statement, page 193). Depending on what is reported here, the task will from then on be controlled by DSSM or by the subsystem.

CLOSE-CTRL-ROUTINE =
Specifies whether the subsystem incorporates a routine for controlling its suspension/deactivation.
If a subsystem is deactivated (by a STOP-SUBSYSTEM or HOLD-SUBSYSTEM command), DSSM passes control to this routine at the identified entry point in the holder task, or (for *DYNAMIC) this is reported via a bourse or FITC link (as determined by return messages during initialization).
The parameters passed are the same ones as are passed for the INIT-ROUTINE operand. Branching to this routine ensures that a connection to the subsystem still exists.

If a CLOSE-CTRL routine exists, it is possible to change versions without interrupting the BS2000 session. At any given point in time, there is exactly one valid version (either the old version is still available, or the new version has already become available). Without a CLOSE-CTRL routine, changing versions always entails interrupting the connection so that the STOPCOM routine of the old version and the INIT routine of the new version can execute (see also page 45).

CLOSE-CTRL-ROUTINE = *NO
The subsystem incorporates no routine that controls the deactivation or suspension of the subsystem.

CLOSE-CTRL-ROUTINE = *DYNAMIC
This routine is called up via the bourse or FITC port. The subsystem passes the required parameters to the CLOSE-CTRL routine dynamically at the end of the INIT routine, and DSSM is informed of the identity of the bourse or FITC port.
In order for the CLOSE-CTRL routine to run, an INIT routine (INIT-ROUTINE operand) and a STOPCOM routine (operand STOPCOM-ROUTINE=*NO/*DYNAMIC) must have been specified.
The holder task of the subsystem must be a work task when the CLOSE-CTRL routine is called (ASSIGN-HOLDER-TASK statement, page 193).

CLOSE-CTRL-ROUTINE = <text 1..8 without-sep>
Name of the entry point of the relevant subsystem routine.

STOPCOM-ROUTINE =
Specifies whether the subsystem incorporates a routine which can carry out active termination of tasks.

STOPCOM-ROUTINE = *NO
The subsystem concerned incorporates no such routine.

STOPCOM-ROUTINE = *DYNAMIC
This routine is called via the bourse or FITC. The subsystem passes the required parameters to the STOPCOM routine dynamically at the end of the CLOSE-CTRL routine or (if none exists) at the end of the INIT routine. DSSM is informed of the identity of the bourse or FITC port.
A prerequisite for the use of the STOPCOM routine is than an INIT routine has been specified (INIT-ROUTINE operand). When the STOPCOM routine is called, the holder task for the subsystem must be used as a work task (ASSIGN-HOLDER-TASK statement, page 193).

STOPCOM-ROUTINE = <text 1..8 without-sep>
Name of the entry point to the subsystem routine concerned.

DEINIT-ROUTINE =
Specifies whether the subsystem incorporates a routine which can carry out deinitialization of the subsystem. This deinitialization routine causes the resources which were requested by the subsystem (memory, files, devices) to be returned.
If a subsystem is deactivated (by a STOP-SUBSYSTEM or HOLD-SUBSYSTEM command), then DSSM passes control to this routine at the identified entry point in the holder task, or (for *DYNAMIC) this is reported via a bourse or FITC link (as determined by return messages during initialization).
If a subsystem is defined with an INIT routine and a CLOSE-CTRL routine, a DEINIT routine – with the same operand value as the CLOSE-CRTL routine – must be specified.
The parameters which are passed are the same as for the INIT-ROUTINE operand.
Branching to this routine ensures that calling tasks will no longer be connected to the subsystem and all existing call relations to the subsystem are deleted.

DEINIT-ROUTINE = *NO
The subsystem concerned does not incorporate a deinitialization routine to request the release of resources; this is done by DSSM itself.

DEINIT-ROUTINE = *DYNAMIC
The routine is called via the bourse or FITC.
The subsystem passes the required parameters to the DEINIT routine dynamically at the end of the STOPCOM routine or, if none exists, at the end of the CLOSE-CTRL routine or, if neither a STOPCOM nor a CLOSE-CTRL routine is incorporated, at the end of the INIT routine. DSSM is informed of the identity of the bourse or FITC port.
A prerequisite for the use of the DEINIT routine is than an INIT routine has been specified (INIT-ROUTINE operand). When the DEINIT routine is called, the holder task for the subsystem must be used as a work task (ASSIGN-HOLDER-TASK statement, page 193).

DEINIT-ROUTINE = <text 1..8 without-sep>
Name of the entry point to the subsystem routine concerned.

STOP-AT-SHUTDOWN =
Specifies whether the subsystem is to be unloaded automatically at shutdown after the user tasks have terminated.

STOP-AT-SHUTDOWN = *NO
The subsystem will not be unloaded automatically.
This parameter should not be specified for subsystems which have address relations to other subsystems which are defined with STOP-AT-SHUTDOWN=*YES.

STOP-AT-SHUTDOWN = *YES
The subsystem will be unloaded automatically at shutdown.
This specification will be ignored if no STOPCOM, DEINIT or CLOSE-CRTL routine is specified. In this case, SSCM issues a message.

INTERFACE-VERSION =
Identifies the entry point via which DSSM can access the interface version which is to be used for calling the INIT, DEINIT, STOPCOM or CLOSE-CTRL routine.

INTERFACE-VERSION = *NO
The subsystem does not call a INIT, DEINIT, STOPCOM or CLOSE-CTRL routine.

INTERFACE-VERSION = <text 1..8 without-sep>
Name of the entry point. The entry point points to the standard header where the interface version is stored. The standard header is generated by calling the macro $ESMINT(I) with MF=I/L.
This operand is mandatory for subsystems for which an INIT, DEINIT, STOPCOM or CLOSE-CTRL routine has been specified.

SUBSYSTEM-HOLD =
Specifies whether the subsystem which is loaded may be suspended or unloaded.

SUBSYSTEM-HOLD = *ALLOWED
The subsystem which is loaded may be suspended and unloaded. The commands HOLD-SUBSYSTEM and STOP-SUBSYSTEM are permissible for this subsystem.

SUBSYSTEM-HOLD = *FORBIDDEN
The commands HOLD-SUBSYSTEM and STOP-SUBSYSTEM must not be used for this subsystem; it will only be unloaded at shutdown - as specified by the STOP-AT-SHUTDOWN operand.
Unloading the subsystem by replacing it with another subsystem entails no interruption.

STATE-CHANGE-CMDS =
Specifies whether or not the DSSM commands for controlling the subsystem (START-SUBSYSTEM, STOP-SUBSYSTEM, HOLD-SUBSYSTEM and RESUME-SUBSYSTEM) may be used during a session.
If a changeover is made from one version of a subsystem to another, the value specified for STATE-CHANGE-CMDS for the version being replaced is ignored.

STATE-CHANGE-CMDS = *ALLOWED
The commands may be used from the operator terminal and under the privileged user ID (the user ID which has the SUBSYSTEM-MANAGEMENT system privileges).

STATE-CHANGE-CMDS = *FORBIDDEN
The commands must not be used - neither from the operator terminal nor under the privileged user ID.

STATE-CHANGE-CMDS = *BY-ADMINISTRATOR-ONLY
The commands may only be used under the privileged user ID; the commands are not available to the operator at the operator terminal.
If a subsystem is deactivated (with a STOP-SUBSYSTEM or HOLD-SUBSYSTEM command), DSSM passes control to this routine at the specified entry point in the holder task, or (if *DYNAMIC was specified) this is reported via a bourse or FITC link (as determined by return messages during initialization).
The parameters passed are the same ones as are passed for the INIT-ROUTINE operand.
Branching to this routine ensures that calling tasks will no longer be connected to the subsystem. Tasks which are still in a call relation to the subsystem remain unaffected by this.

FORCED-STATE-CHANGE =
Specifies whether use of the operand FORCED=*YES is permitted within the commands STOP-SUBSYSTEM and HOLD-SUBSYSTEM. This function can be used to force the unconditional deactivation of the subsystem.

FORCED-STATE-CHANGE = *FORBIDDEN
It is not possible to force deactivation of the subsystem. DSSM will reject any use of the FORCED operand in the commands concerned, and will issue a corresponding error message.

FORCED-STATE-CHANGE =* ALLOWED
The operand FORCED=*YES may be used for this subsystem.
This operand value must not be used in conjunction with SUBSYSTEM-HOLD=*FORBIDDEN.

RESET =
Specifies whether the operand RESET=*YES is permitted within the commands START-SUBSYSTEM and RESUME-SUBSYSTEM. This function can be used to force the unconditional loading or resumption of the subsystem, even if the state of the subsystem is currently IN-DELETE or IN-HOLD.

RESET = *FORBIDDEN
It is not possible to force the activation of the subsystem. DSSM will reject the use of the RESET operand in the commands concerned, and will issue a corresponding error message.

RESET = *ALLOWED
The operand RESET=*YES may be used for this subsystem.
This operand value must not be specified together with SUBSYSTEM-HOLD=*FORBIDDEN.

RESTART-REQUIRED =
Specifies whether the initialization routine for the subsystem is to be executed if the holder task terminates abnormally.

RESTART-REQUIRED = *NO
The initialization routine is not used to restart the subsystem.

RESTART-REQUIRED = *YES
If the holder task terminates abnormally, the initialization routine should be used. Provision must have been made in the INIT-ROUTINE operand for executing this routine.

VERSION-COEXISTENCE =
Specifies whether more than one version of the same subsystem may be active at a time.

VERSION-COEXISTENCE = *FORBIDDEN
The current version of the subsystem cannot coexist with another version of the same subsystem.

VERSION-COEXISTENCE = *ALLOWED
The current version of the subsystem can coexist with another version of the same subsystem (coexistence mode).
In the definition of the job entry point (SUBSYSTEM-ENTRIES operand), indirect links via system exit routines must not have been specified. If different versions of the same subsystem are loaded and the same job entry point is defined for these, the link which is implemented is always to the highest loaded version of the subsystem.
If coexistent subsystems access coexistent syntax files, the latter must have been declared in the SSD object and cannot be administered by SDF.
However, where the links are via SVC and ISL, it is possible to select a version using the operands FUNCTION-NUMBER and FUNCTION-VERSION.

VERSION-EXCHANGE =
Specifies whether a subsystem may be loaded in exchange mode. Exchange mode allows the temporary coexistence of two versions of the same subsystem. If version B of a subsystem is loaded whilst version A of the subsystem is already active, all new callers will be connected to version B. Jobs which are connected to version A will still be processed. When all the jobs which use version A have been processed, this will automatically be terminated.
In the definition it should be noted that the “old” version which is being replaced must not be dependent on the “new” version which replaces it.

VERSION-EXCHANGE = *FORBIDDEN
The current version of the subsystem must not be replaced.

VERSION-EXCHANGE = *ALLOWED
Exchange mode, which allows the temporary coexistence of two subsystems, is permitted for the current subsystem version.

ADD-SUBS-ENTRIES / MODIFY-SUBS-ENTRIES =
Indicates whether new job entries are to be defined (ADD), or the attributes of existing job entries are to be changed (MODIFY).

ADD-SUBS-ENTRIES / MODIFY-SUBS-ENTRIES = <u>*NONE</u>
Default value: new job entries are not to be added, nor are the attributes of existing job entries to be modified.

ADD-SUBS-ENTRIES / MODIFY-SUBS-ENTRIES = list-poss(100): <text 1..8>
Either declares names of entry points for a maximum of 100 new job entries for the subsystem (for each of which the type must be defined in the substructures (ADD), or modifies job entry points which have already been defined (MODIFY).

MODE =
Defines the type of a job entry point which is defined for the subsystem.
If the type of the entry point declared is modified, all the suboperands of MODE must be assigned a value explicitly; i.e. the operand value *UNCHANGED (default value for MODIFY-SUBS-ENTRIES) will be rejected.

MODE = *LINK
The job entry point cannot be accessed by indirect linkage, but only by using a CONNECT relation through an external linkage editor symbol.
In the case of different versions of the same subsystem which use the same external linkage editor symbol, DSSM automatically sets up the link to the highest loaded version of the subsystem.

MODE = *ISL(...)
The job entry is effected by indirect linkage via System Procedure Linkage (for privileged subsystems only). If the specification includes in addition a function and version number for the ISL entry point, the combination of entry point name, function and version numbers must not match any other combination for the various other subsystems in the catalog or the various versions of the same subsystem (if VERSION-COEXISTENCE=*ALLOWED is specified).
For different subsystems, if the job entry point is to be accessed by the same ISL entry point, they must be uniquely identified by specifying the function and version numbers.
In the case of different versions of the same subsystem which use the same ISL entry point, then - if the function and version numbers are not specified - DSSM will automatically set up a connection to the highest loaded version of the subsystem.
In the case of different versions of the same subsystem which use the same ISL entry point and for which the function and version numbers are not equal to *NONE, the version to which the connection is set up will be selected in accordance with the function and version numbers stored in the standard header of the caller's parameter list.
It is not permissible to enter a value of *ALL for the CONNECTION-ACCESS operand in reference to ISL entry points.

FUNCTION-NUMBER =
Specifies whether a particular function and version number of the ISL entry point is to be addressed, as the same ISL entry point can be used by different functions.

FUNCTION-NUMBER = *NONE
Default value: no particular function or version number is to be addressed.

FUNCTION-NUMBER = <integer 0..255>(...)
Number of the ISL entry point. The version must be nominated in the substructure which follows.

FUNCTION-VERSION = <integer 1..255>
Version of the specified ISL function number.

MODE = *SVC(...)
Job entry is to be effected by an indirect connection using a supervisor call (SVC).
If the specification includes in addition a function and version number for the SVC entry point, the combination of entry point name, function and version numbers must not match any other combination for the various other subsystems in the catalog or the various versions of the same subsystem (if VERSION-COEXISTENCE=*ALLOWED is specified).
For different subsystems, if the job entry is to be accessed by the same SVC, they must be uniquely identified by specifying the function and version numbers.
In the case of different versions of the same subsystem which use the same SVC, then – if the function and version numbers are not specified – DSSM will automatically set up a connection to the highest loaded version of the subsystem.
In the case of different versions of the same subsystem which use the same SVC and for which the function and version numbers are not equal to *NONE, the version to which the connection is set up will be selected in accordance with the function and version numbers stored in the standard header of the caller's parameter list.

NUMBER = <integer 0..255>
Number of the SVC via which job entry is to be effected. No SVC number greater than 191 may be used in conjunction with CONNECTION-ACCESS=*ALL.

CALL-BY-SYSTEM-EXIT =
Defines whether the specified SVC number may be called from within system exit routines.

CALL-BY-SYSTEM-EXIT = *ALLOWED
System exit routines are permitted to call the specified SVC number.

CALL-BY-SYSTEM-EXIT = *FORBIDDEN
System exit routines are not permitted to call the specified SVC number.

FUNCTION-NUMBER =
Specifies whether a particular function and version number of the SVC entry point is to be addressed, as the same SVC entry point can be used by different functions.

FUNCTION-NUMBER = *NONE
No particular function or version number is to be addressed.

FUNCTION-NUMBER = <integer 0..255>(...)
Number of an SVC entry point. The version must be nominated in the substructure which follows.

FUNCTION-VERSION = <integer 1..255>
Version of the specified SVC function number.

MODE = SYSTEM-EXIT(...)
Job entry is to be effected by an indirect connection using system exit routines.
This operand must not be used in conjunction with CONNECTION-ACCESS=*ALL.

NUMBER = <integer 0..127>
Number of the system exit routine.

CONNECTION-ACCESS =
Specifies the access authorization (privileges) required by the subsystem.

CONNECTION-ACCESS = *ALL
Privileged and nonprivileged program runs may access the subsystem.
This operand value must not be used in conjunction with MODE=*SYSTEM-EXIT/*ISL/*SVC (with an SVC number greater than 191).

CONNECTION-ACCESS = *SYSTEM
Only privileged program runs may access the subsystem.

CONNECTION-ACCESS = *SIH
Only tasks running in the SIH processor state may access the subsystem.
The subsystem called also runs in the SIH processor state, i.e. it is uninterruptible.
This operand value is permissible only for subsystems for which the entry point is defined via:
– System Procedure Linkage (MODE=*ISL(FUNCTION-NUMBER=*NONE))
– CONNECTION-SCOPE=*OPTIMAL
– MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=*SYSTEM)

CONNECTION-SCOPE =
Identifies the event which will call up the automatic cleardown of the connection to the specified subsystem job entry.

CONNECTION-SCOPE = *TASK
The connection will be cleared when the task terminates.

CONNECTION-SCOPE = *PROGRAM
The connection will be cleared when the program terminates, or before.
Only CONNECTION-SCOPE=*PROGRAM may be specified in conjunction with MEMORY-CLASS=*LOCAL-UNPRIVILEGED.
This operand value is recommended for subsystems which were declared with SUBSYSTEM-ACCESS=*LOW/*HIGH or MEMORY-CLASS=*BY-SLICE.

CONNECTION-SCOPE = *FREE
DSSM is not to carry out any checking of the connections to the job entry point. The connection will not be automatically cleared - unless explicitly requested. To avoid problems or possible errors when the subsystem is being unloaded, the connections must be managed by the subsystem itself.

CONNECTION-SCOPE = *CALL
On return from this job entry point, DSSM will automatically clear the connections. This operand value is only available with subsystems for which the job entry point is defined by means of System Procedure Linkage (ISL) or supervisor calls (SVC).

CONNECTION-SCOPE = *OPTIMAL
The subsystem is deactivated or suspended when there are no further tasks with a connection to this entry point.
A routine with an entry point defined with *OPTIMAL must be terminated with RETURN.
If an entry point of a subsystem is defined with CONNECTION-SCOPE=*OPTIMAL, all of its entry points must be defined in the subsystem catalog with MODE≠*LINK.
While a subsystem is being deactivated or suspended, no call of the subsystem with CONNECTION-SCOPE=*OPTIMAL is accepted.

FIRST-CONNECTION =
Determines whether or not first connection of the task to the specified job entry point in the subsystem is allowed. At least one job entry point of a subsystem must be defined with FIRST-CONNECTION=*ALLOWED.

FIRST-CONNECTION = *ALLOWED
First connection to the specified job entry point is allowed.
This value is the default setting for the ADD-SUBS-ENTRIES statement.

FIRST-CONNECTION = *FORBIDDEN
Connection to the specified job entry point via SVC or ISL is not allowed if the task has not yet been connected to another job entry point belonging to the subsystem.
It is not permitted to specify this operand value for job entry points that have been defined with MODE=*LINK/*SYSTEM-EXIT or CONNECTION-ACCESS=*SIH.

MODIFY-SUBS-ENTRIES = *NONE / list-poss(100): <text 1..8> / *BY-PROGRAM(...)
The values *NONE and list-poss(100): <text 1..8> are described on page 220 at the ADD-SUBS-ENTRIES operand.

MODIFY-SUBS-ENTRIES = *BY-PROGRAM(...)
The entry points of the specified subsystem are supplied dynamically from the BLS name list at load time instead of statically from the catalog. A prerequisite for this functionality is the use of BLSSERV as of version 2.1, that supports using EEN names as entry points for DSSM subsystems.

The statement is rejected if the subsystem was not previously also defined with *BY-PROGRAM with the SET-SUBSYSTEM-ATTRIBUTES statement. MODIFY-SUBS-ENTRIES is used for changing the connection settings.

If *BY-PROGRAM is used, the ADD-SUBS-ENTRIES and REMOVE-SUBS-ENTRIES operands must be specified with *NONE.

CONNECTION-SCOPE = *TASK / *PROGRAM
The connection is shut down at task or program termination.

REMOVE-SUBS-ENTRIES =
Defines whether or not existing job entries which have been defined for the subsystem are to be deleted.

REMOVE-SUBS-ENTRIES = <u>*NONE</u>
Default value: none of the job entries is to be deleted.

REMOVE-SUBS-ENTRIES = list-poss(100): <text 1..8>
Names of the job entries (up to 100) which are no longer to apply to the subsystem.

MEMORY-CLASS =
Specifies the subsystem-specific address space in which the subsystem is to be loaded. System administration can use this operand to define the address space valid for the subsystem concerned so as to meet the special requirements of the installation.
If the subsystem-specific address space is modified, each of the suboperands of MEMORY must be assigned a value explicitly (the operand value *UNCHANGED (default value) will be rejected).

MEMORY-CLASS = *SYSTEM-GLOBAL(...)
The subsystem will be loaded in class 3 or class 4 memory. Resident CSECTs will be given class 3 memory, all others will be given pageable class 4 memory.

SUBSYSTEM-ACCESS =
Identifies the access authorization (privileges) and the location of the requested memory.

SUBSYSTEM-ACCESS = <u>*LOW</u>
Nonprivileged address space below the 16-Mbyte boundary is allocated.

SUBSYSTEM-ACCESS = *SYSTEM
Subsystems declared with this operand value are privileged subsystems to which privileged address space above the 16-Mbyte boundary is allocated.
This operand value is mandatory for subsystems whose entry point is declared via SVC (MODE=*SVC) or for which an INIT, STOPCOM, DEINIT or CLOSE-CTRL routine is declared. It is not permitted in combination with CONNECTION-ACCESS=*ALL and MODE=*LINK.

SUBSYSTEM-ACCESS = *HIGH
Nonprivileged address space up to 2 Gbytes is allocated.

MEMORY-CLASS = *LOCAL-PRIVILEGED(...)
The subsystem is given a memory pool in nonprivileged class 5 memory, located below the 16-Mbyte boundary.
This specification is suitable for nonprivileged subsystems which demand a relatively large amount of address space (approx. 1 Mbyte) and have to be set up below the 16-Mbyte boundary. These subsystems are loaded in memory pools at the same address, in order to manage the use of the limited address space below 16 Mbytes.
Although such subsystems are loaded in parallel in the same address space, they cannot be used simultaneously by the same task (see also the SEPARATE-ADDRESS-SPACE statement, page 241).
The subsystem must not contain any resident CSECTs as otherwise a later attempt to activate it will be aborted.

SIZE = <integer 1..32767>
Size of the required address space (in 4Kbyte pages) for the memory pool in class 5 memory. This value should be set at least high enough to ensure that the subsystem and any selectable units/load units which it may load dynamically can be loaded in their entirety. The upper limit is generation-specific.

MEMORY-CLASS = *LOCAL-UNPRIVILEGED(...)
The subsystem is given a memory pool in nonprivileged class 6 memory. This specification is reserved for subsystems which can be executed like a program.
In keeping with this, their access authorization (privileges) must be defined with the value *ALL in the CONNECTION-ACCESS operand.
This operand value must not be specified together with an entry point which was defined with CONNECTION-ACCESS=*SYSTEM.
The subsystem must not contain any resident CSECTs as otherwise a later attempt to activate it will be aborted.
If this operand value is specified, only CONNECTION-SCOPE=*PROGRAM is permissible for clearing the connection to the specified subsystem entry point.

SIZE = <integer 1..32767>
Size of the required address space (in 4Kbyte pages) for the memory pool in class 6 memory. This value should be set at least high enough to ensure that the subsystem and any selectable units/load units which it may load dynamically can be loaded in their entirety. The upper limit is generation-specific.

SUBSYSTEM-ACCESS =
Identifies the location of the requested memory space.

SUBSYSTEM-ACCESS = *LOW
Nonprivileged address space below the 16-Mbyte boundary is allocated.
Because this specification is suitable for subsystems which can be executed like programs, it is advisable additionally to specify CONNECTION-SCOPE=*PROGRAM.

SUBSYSTEM-ACCESS = *HIGH
Nonprivileged address space up to 2 Gbytes is allocated.
Because this specification is suitable for subsystems which can be executed like programs, it is advisable additionally to specify CONNECTION-SCOPE=*PROGRAM.

START-ADDRESS =
Defines the start address in class 6 memory.

START-ADDRESS = *ANY
The location of the subsystem in class 6 memory will be determined by DSSM.

START-ADDRESS = <x-string 7..8>
Start address in the segment raster at which the subsystem’s start address is to be located. The value specified must be an 8-character hexadecimal constant which is a multiple of X'100000'.

MEMORY-CLASS = *BY-SLICE(...)
The specified subsystem is a nonprivileged subsystem and consists of an LLM, which in turn consists of a shareable code (program area) and a non-shareable code (data area). The program area is loaded into the shareable address space (this corresponds to MEMORY-CLASS=*SYSTEM-GLOBAL). The data area is loaded into the user address space of the holder task and is copied into the private user address spaces of the connected tasks at the same address.
The following values must be specified together with MEMORY-CLASS=*BY-SLICE: SUBSYSTEM-LOAD-MODE=*ADVANCED and CONNECTION-ACCESS=*ALL.

SIZE = <integer 1..32767>
Specifies the size of the requested memory space for the data area in 4K pages.
The value chosen here must be sufficiently large to allow the subsystem and, if appropriate, selectable units/load units dynamically loaded by the subsystem to be loaded in full. The upper limit is dependent on generation.

LINK-ENTRY = <text 1..8 without-sep>(...)
Defines the name of the object module/ENTRY/CSECT required for loading (for the operand in the call of the $PBBND1 macro to the dynamic binder loader DBL). The subsystem must be loaded in its entirety by this ENTRY (if necessary, using autolink).

AUTOLINK =
Controls invocation of the autolink function during linking and loading.
The linkage editor's autolink function permits the automatic insertion of modules which are not explicitly inserted by appropriate statements. The main purpose of this function is to save users of higher-level programming languages from having to make explicit statements to insert individually the numerous modules of the runtime system which are required. Further details of the autolink function can be found in the "BLSSERV" manual [4].
The autolink function can also be implicitly circumvented if the first external reference encountered during linkage editing of the object module which is to be loaded points to a prelinked module. The advantage of this approach is that the paging behavior when the subsystem is later executed can be optimized at this preliminary stage (during linkage editing). In addition, errors during linkage editing can be avoided in this way.

AUTOLINK = *ALLOWED
The autolink function is allowed.

AUTOLINK = *FORBIDDEN
The autolink function is suppressed.

ADD-REFER-SUBS / MODIFY-REFER-SUBS =
Specifies whether to set up a list containing subsystems to which there are address relations, for use in resolving external references (ADD), or if there already exists a list which is to be modified (MODIFY).

ADD-REFER-SUBS / MODIFY-REFER-SUBS = *NONE
Default value: no list is to be set up, nor is an existing list to be modified.

ADD-REFER-SUBS / MODIFY-REFER-SUBS = list-poss(15):<structured-name 1..8>
There are external references to be specified (ADD) or modified (MODIFY).
External references can be nominated for a maximum of 15 other subsystems; these subsystems must be used in resolving the external references. If any of the subsystems nominated here is missing at the time of activation or deactivation (and if a check of the external references has also been requested by the operand CHECK-REFERENCE=*YES), the action will be aborted.
It is also possible to address the BS2000/OSD control program via these external references - using the name CP. In the substructure which follows, it is possible to specify either exactly one version, or a range of versions within which all versions are to be referred to.
If a version range list is used to limit the version of the CP subsystem, DSSM checks the compatibility of the current CP version against the versions in the range list. The subsystem will only be loaded if it is a compatible version.

The following restrictions should be noted when specifying subsystems to which there are address relations:

- No address relations may be declared to subsystems which have the attribute MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED/*BY-SLICE.
- If the attribute SUBSYSTEM-ACCESS=*SYSTEM is specified for the subsystem which is being defined, no subsystem may be addressed if it is defined with SUBSYSTEM-ACCESS=*LOW or SUBSYSTEM-ACCESS=*HIGH.
- Subsystems which have the attribute STOP-AT-SHUTDOWN=*YES may have address relations only to other subsystems which also have this attribute.
- As a rule, a nonprivileged subsystem must not have any address relations to the control program (CP)
- If a reference is made to a subsystem which has at least one version that may be operated in coexistence or exchange mode, a unique version must be specified
- Any address relations must be defined in accordance with the start attributes (CREATION-TIME operand); i.e. a subsystem may have relations to other subsystems only if these were started at the same time or earlier.

LOWEST-VERSION =
Specifies the lowest value (lowest version) in the subsystem version range list.

LOWEST-VERSION = *LOWEST-EXISTING
The lowest version in the catalog is to be addressed.

LOWEST-VERSION = <c-string 3..8> / <text 3..8>
Version of the subsystem which is to be used as the lower limit of the range of versions.

HIGHEST-VERSION =
Specifies the upper value (highest version) in the subsystem version range list.

HIGHEST-VERSION = *HIGHEST-EXISTING
The highest version in the catalog is to be addressed.

HIGHEST-VERSION = <c-string 3..8> / <text 3..8>
Version of the subsystem which is to be used as the upper limit of the range of versions.

MODIFY-REFER-SUBS =
See the description of ADD-REFER-SUBS.

REMOVE-REFER-SUBS =
Indicates whether or not existing external references to other subsystems are to be deleted.

REMOVE-REFER-SUBS = *NONE
Default value: none of the existing external references to other subsystems is to be deleted.

REMOVE-REFER-SUBS = list-poss(15): <structured-name 1..8>
Names of a maximum of 15 external references which are to be made invalid.

UNRESOLVED-EXTERNALS =
Defines how the load procedure is to behave if there are unresolved external references.

UNRESOLVED-EXTERNALS = *FORBIDDEN
If unresolved external references occur, the load procedure will be terminated.

UNRESOLVED-EXTERNALS = *ALLOWED
The load procedure will be continued; unresolved external references will be given the value X'FFFFFFFF'.

CHECK-REFERENCE =
Defines whether or not the subsystems specified in the REFERENCED-SUBSYSTEM operand are to be checked in respect of their status and availability.

CHECK-REFERENCE = *YES
The referenced subsystems will be checked. If any of them is missing, DSSM will abandon the activation or unloading of the subsystem.

CHECK-REFERENCE = *NO
DSSM is not to carry out any check.
Even if the user generates complex subsystems with this statement, DSSM will still execute the requested functions **despite** the risk of conflicts:

– The START-SUBSYSTEM command loads the specified subsystem even if a subsystem to which defined relations exist has not yet been fully loaded.
– DSSM executes the RESUME-SUBSYSTEM, STOP-SUBSYSTEM and HOLD-SUBSYSTEM commands without performing any check on relations or dependencies.

ADD-RELATED-SUBS / MODIFY-RELATED-SUBS =
Specifies whether a list of subsystems for which dependency relations exist are to be created (ADD) or an existing list of dependent subsystems is to be modified (MODIFY).

ADD-RELATED-SUBS / MODIFY-RELATED-SUBS = *NONE
Default value: no dependency relations are to be defined or modified for the subsystem.

ADD-RELATED-SUBS / MODIFY-RELATED-SUBS = list-poss(100): <structured-name 1..8>
Dependency relations are to be defined (ADD) or modified (MODIFY) for up to 100 other subsystems, without which the subsystem currently being defined cannot function.
It is also permissible to define dependency relations to the BS2000/OSD control program (CP). The rules and restrictions which apply when doing so are analogous to those for address relations, where they are described in more detail (see the ADD-REFER-SUBS operand).
A dependency relation always points to a single version of a subsystem. In the substructure which follows, it is possible to specify either exactly one version, or a range of versions within which all versions are to be referred to.

The following general restrictions should be noted when specifying dependent subsystems:

– The relation which is defined must not contain closed loops. A loop arises if subsystem A is dependent on B, B is dependent on C, and this is in turn dependent on A
– If the subsystem which is being defined has been given the attribute MEMORY-CLASS=*SYSTEM-GLOBAL, then it is not permissible to address any subsystems defined with MEMORY-CLASS=*LOCAL-PRIVILEGED or *LOCAL-UNPRIVILEGED.
– For subsystems which have the attribute SUBSYSTEM-ACCESS=*SYSTEM, no dependency relations may be defined to subsystems to which SUBSYSTEM-ACCESS =*LOW or SUBSYSTEM-ACCESS=*HIGH or MEMORY-CLASS=*BY-SLICE applies
– The dependency relations must be defined to correspond to the start attributes (CREATION-TIME operand); i.e. a subsystem may only be dependent on subsystems which are started at the same time or earlier.

LOWEST-VERSION =
Specifies the lowest value (lowest version) in the subsystem version range list.

LOWEST-VERSION = *LOWEST-EXISTING
The lowest version in the catalog is to be addressed.

LOWEST-VERSION = <c-string 3..8> / <text 3..8>
Version of the subsystem which is to be used as the lower limit of the range of versions.

HIGHEST-VERSION =
Specifies the upper value (highest version) in the subsystem version range list.

HIGHEST-VERSION = *HIGHEST-EXISTING
The highest version in the catalog is to be addressed.

HIGHEST-VERSION = <c-string 3..8> / <text 3..8>
Version of the subsystem which is to be used as the upper limit of the range of versions.

MODIFY-RELATED-SUBS =
See the description of ADD-RELATED-SUBS.

REMOVE-RELATED-SUBS =
Specifies whether existing dependency relations to other subsystems are to be deleted.

REMOVE-RELATED-SUBS = *NONE
Default value: none of the existing dependency relations to other subsystems is to be deleted.

REMOVE-RELATED-SUBS = list-poss(100): <structured-name 1..8>
Names of a maximum of 100 subsystems to which all dependency relations are to be removed.

MODIFY-WORK-TASK-ATTRIBUTE
Modify work task parameters

Function

This statement can be used to specify subsystems which are no longer to use their holder tasks as a work task. It is possible to specify in addition a new TSN which is to be given to the holder task. This statement will be rejected if the specified subsystem does not exist, or if it does not use the holder task as a work task.

MODIFY-WORK-TASK-ATTRIBUTE will be rejected if neither of the following statements has been executed beforehand:

– START-CATALOG-CREATION
– START-CATALOG-MODIFICATION

Format

MODIFY-WORK-TASK-ATTRIBUTE
SUBSYSTEM-NAME = <structured-name 1..8>
,**SUBSYS**TEM-**VERS**ION = <c-string 3..8> / <text 3..8>
,**WORK-TASK** = ***UNCHA**NGED (...) / ***NO**
***UNCHA**NGED(...)
⏐ **TSN** = ***BY-DSSM** / <alphanum-name 1..4>

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem which is using the holder task as a work task.

SUBSYSTEM-VERSION = <c-string 3..8> / <text 3..8>
Version of the subsystem which is using the holder task as a work task.
This version must have been declared previously.

WORK-TASK =
Specifies whether the holder task is no longer to be used as a work task, or if the holder task is simply to be given a new TSN.

WORK-TASK = <u>*UNCHANGED</u>
The subsystem is to continue to use the holder task as a work task. However, a new TSN can be issued for the holder task, in the substructure which follows.

TSN =
Specifies the task sequence number (TSN) to be given to the subsystem’s work task.

TSN = <u>*BY-DSSM</u>
Default value: the TSN is issued by DSSM when the work task is loaded.

TSN = <alphanum-name 1..4>
Task sequence number to be given to the work task when it is started.
The specified TSN must be uniquely defined and usable when the subsystem is loaded.

WORK-TASK = *NO
The holder task for the specified subsystem is no longer to be used as a work task.

REMOVE-ADDR-SPACE-SEPARATION Revoke disjunctive distribution of subsystems in class 5 memory

Function

This statement can be used to revoke the declarations made using a SEPARATE-ADDRESS-SPACE statement (see page 241) and specifying the non-overlapping (disjunctive) distribution of subsystems in class 5 memory.

Use of this statement is irrelevant for subsystems in class 3 or class 4 memory (operand MEMORY-CLASS=*SYSTEM-GLOBAL in the SET-SUBSYSTEM-ATTRIBUTES statement) and for subsystems in class 6 memory (MEMORY-CLASS=*LOCAL-UNPRIVILEGED). Subsystems in class 6 memory are never used in parallel, and may therefore overlap in the address space; subsystems in class 3 or 4 memory never overlap.

REMOVE-ADDR-SPACE-SEPARATION will be rejected if neither of the following statements has been executed beforehand:
– START-CATALOG-CREATION
– START-CATALOG-MODIFICATION

The name of the subsystem must be contained in the catalog. If any of the address relations specified in the list of subsystems does not exist, it will be ignored; processing of the statement will continue.

Format

REMOVE-ADDR-SPACE-SEPARATION
SUBSYSTEM-NAME = <structured-name 1..8>
,**FROM-SUBSYSTEMS** = list-poss(15): <structured-name 1..8>

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem which until now was not permitted to overlap with other subsystems.
The subsystem must be known to SSCM and must already be defined.

FROM-SUBSYSTEMS = list-poss(15): <structured-name 1..8>
List of a maximum of 15 subsystems for each of which a declaration has been made to the effect that it must not overlap with the subsystem named in the SUBSYSTEM-NAME operand. The declaration will be revoked for these subsystems.

REMOVE-CATALOG-ENTRY
Logically delete definition of subsystem from subsystem catalog

Function

This statement is used to logically delete the definition of a subsystem which is stored in the subsystem catalog.

The statement will be rejected if a current (non-empty) catalog, in which the definition is stored, has not been opened by a START-CATALOG-CREATION or START-CATALOG-MODIFICATION statement.

The statement will also be rejected if the subsystem specified is not contained in the catalog named. If the subsystem is removed from the catalog, then any disjunctive relations (declared by a SEPARATE-ADDRESS-SPACE statement) will also be deleted.

i Where there are dependency or load relations, the removal of a subsystem may lead to errors during consistency checks on the catalog.

Format

REMOVE-CATALOG-ENTRY

SUBSYSTEM-NAME = <structured-name 1..8>(...)
 	
  <structured-name>(...)
 	
    | **VERSION** = ***ALL** / <c-string 3..8> / <text 3..8>

Operands

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Name of the subsystem whose definition is to be deleted from the catalog.

VERSION = *ALL / <c-string 3..8> / <text 3..8>
Specifies the version of the subsystem whose definition is to be deleted from the catalog. If VERSION=*ALL is specified, the definitions of all versions of the specified subsystem are deleted from the catalog.

SAVE-CATALOG
Save subsystem catalog as PAM file

Function

With the aid of this statement, a subsystem catalog created in memory by means of statements entered earlier can be saved as a PAM file.

The file name is either derived from the last START-CATALOG-CREATION statement (see page 285) or from the START-CATALOG-MODIFICATION statement (see page 286), or a new file name can be specified in fully qualified form.

SAVE-CATALOG will be rejected with an error message:

– if neither of the statements START-CATALOG-CREATION or START-CATALOG-MODIFICATION has been executed beforehand
– if the catalog created in memory contains no definitions, i.e. is empty
– if the catalog already contains a file with the name (other than *CURRENT) specified in CATALOG-NAME and REPLACE-OLD-FILE=*NO was specified.

Note that this statement must be preceded by a check run for the catalog. This check run can be initiated by a CHECK-CATALOG statement (see page 196).

Format

SAVE-CATALOG
CATALOG-NAME = ***CURRENT** / <filename 1..51 without-gen-vers>(...) | **REPLACE-OLD-FILE** = ***NO** / ***Y**ES ,**FORCED** = ***NO** / ***Y**ES / ***FOR-ADD-SUBSYSTEM**

Operands

CATALOG-NAME =
File name under which the catalog is to be saved.

CATALOG-NAME = <u>*CURRENT</u>
The catalog will be saved under the file name specified in the last START-CATALOG-CREATION or START-CATALOG-MODIFICATION statement.

CATALOG-NAME = <filename 1..51 without-gen-vers>
Fully qualified file name under which the catalog is to be saved.

> **REPLACE-OLD-FILE= <u>*NO</u> / *YES**
> Specifies whether an existing, cataloged file with the name specified in CATALOG-NAME may be overwritten (*YES) or not (*NO).

FORCED =
Determines how SSCM is to behave if there are errors in the subsystem definition.

FORCED = <u>*NO</u>
Default value: in the case of errors which SSCM recognizes during the analysis of the subsystem definition, the catalog is not to be saved in a PAM file.
Errors may arise in the definitions of subsystems due to a set of relations (address, dependency or external reference relations) which contains loops. Such a loop results if a relation (“→”) is defined which, after a number of steps, refers back to itself:

Subsystem A ⟶ Subsystem B
Subsystem B ⟶ Subsystem C
Subsystem C ⟶ Subsystem A

FORCED = *YES
In spite of any errors which SSCM may identify during the analysis of the subsystem definitions, the catalog is to be saved in a PAM file.
Responsibility for a catalog saved in this way lies with the user; the behavior of DSSM cannot be guaranteed.

FORCED = *FOR-ADD-SUBSYSTEM
The specified catalog will be saved even if the following errors arise:
– a subsystem includes dependency relations to undefined subsystems or
– a subsystem is to share a holder task with an undefined subsystem.

Use of this operand is a good idea when a catalog is to be created that contains nothing but new subsystems, which may, however, have relations to other subsystems defined in the old catalog. This new catalog is added to the existing old catalog by means of the command ADD-SUBSYSTEM ...,TYPE=*NEW-SUBSYSTEMS.

SAVE-SSD
Terminate subsystem definition(s)

Function

The SAVE-SSD statement terminates the sequence of statements used to define one or more subsystems initiated by the statements START-SSD-CREATION (see page 287) and SET-SUBSYSTEM-ATTRIBUTES (see page 243).
This statement requests SSCM to save the subsystem definition(s) to the ISAM file named in the START-SSD-CREATION statement. This ISAM file is called the SSD object.

SAVE-SSD will be rejected if neither of the following statements has been executed beforehand:
- START-SSD-CREATION
- SET-SUBSYSTEM-ATTRIBUTES

SAVE-SSD is aborted and an error message is issued if the SSD object contains subsystems which
- have been defined with CLOSE-CTRL-ROUTINE=*DYNAMIC, but the holder task is not being used as a work task
- have different INSTALLATION-UNIT names (installed software units)

Format

SAVE-SSD

SEPARATE-ADDRESS-SPACE
Control disjunctive distribution of subsystems in class 5 memory

Function

This statement is used to control the disjunctive distribution of subsystems in class 5 memory. Using this statement, it is possible to prevent unwanted overlaps in the address spaces of subsystems which could be caused by the SET-SUBSYSTEM-ATTRIBUTES statements.

Use of this statement is irrelevant for subsystems in class 3 or class 4 memory (operand MEMORY-CLASS=*SYSTEM-GLOBAL in the SET-SUBSYSTEM-ATTRIBUTES statement), and for subsystems in class 6 memory (MEMORY-CLASS=*LOCAL-UNPRIVILEGED).
Subsystems in class 6 memory are never used in parallel, and may therefore overlap in the address space; subsystems in class 3 or 4 memory never overlap.

SET-SUBSYSTEM-ATTRIBUTES will be rejected if none of the following statements has been executed beforehand:
– START-SSD-CREATION
– START-CATALOG-CREATION
– START-CATALOG-MODIFICATION

Format

SEPARATE-ADDRESS-SPACE
SUBSYSTEM-NAME = <structured-name 1..8>
,**FROM-SUBSYSTEMS** = list-poss(15): <structured-name 1..8>

Operands

SUBSYSTEM-NAME = <structured-name 1..8>
Name of the subsystem which must not overlap with other subsystems.
The subsystem must be known to SSCM and must already be defined.

FROM-SUBSYSTEMS = list-poss(15): <structured-name 1..8>
List of a maximum of 15 subsystems, none of which may overlap with the subsystem named in the SUBSYSTEM-NAME operand.

Notes

The SEPARATE-ADDRESS-SPACE statement always creates at least two disjunctive relations, as can be seen from the following example:

```
//SEPARATE-ADDRESS-SPACE SUBSYSTEM-NAME=ONE,FROM-SUBSYSTEMS=(TWO,THREE)
```

Execution of this statement creates the following relations:

– the address space of subsystem ONE is non-overlapping with the address space of subsystem TWO
– the address space of subsystem ONE is non-overlapping with the address space of subsystem THREE
– the address space of subsystem TWO is non-overlapping with the address space of subsystem ONE
– the address space of subsystem THREE is non-overlapping with the address space of subsystem ONE

SET-SUBSYSTEM-ATTRIBUTES
Define attributes and entry points of subsystem

Function

This statement enables all the attributes and entry points for a subsystem to be set.

The way in which the definition of a new subsystem will be compiled depends on what statement preceded the definition:

– After a START-CATALOG-CREATION statement (page 285), the subsystem can be integrated into the **catalog**.

– After a START-SSD-CREATION statement (page 287), the subsystem can be integrated into the **SSD object**.
  However, it is not permissible to set up different versions of the same subsystem in the same SSD object.

Subsystems defined with the operand MEMORY-CLASS=*SYSTEM-GLOBAL ..., SUBSYSTEM-ACCESS=*SYSTEM are privileged subsystems.

When setting up the definition, the following points should be noted:

– The name and version of the subsystem must not already be present in the subsystem catalog.
– Two different versions of one and the same subsystem cannot be defined in the same SSD object.
– The subsystem name CP is reserved for DSSM and must not be specified.
– The names of the entry points must not be present in duplicate.
– The name of the subsystem which is being defined must not be included in the list of subsystems to which there are address or dependency relations.
– Within this list, a subsystem name must not appear more than once.

SET-SUBSYSTEM-ATTRIBUTES is rejected if none of the following statements were executed beforehand:

– START-SSD-CREATION
– START-CATALOG-CREATION
– START-CATALOG-MODIFICATION

Note on syntax

The special data type <symbol>, which is described in detail in the “BLSSERV” manual [4], can also be used for the names of the entry points in the following operands (in the format, the data type is specified as <name>):

– LINK-ENTRY
– DYNAMIC-CHECK-ENTRY
– INIT-ROUTINE
– CLOSE-CTRL-ROUTINE
– DEINIT-ROUTINE
– INTERFACE-VERSION
– SUBSYSTEM-ENTRIES
– STOPCOM-ROUTINE

Format

(part 1 of 4)

```
SET-SUBSYSTEM-ATTRIBUTES

 SUBSYSTEM-NAME = <structured-name 1..8>(...)
   <structured-name 1..8>(...)
     |  VERSION = <c-string 3..8> / <text 3..8>
,INSTALLATION-UNIT = *NONE / *STD / <text 1..30>
,INSTALLATION-USERID = *NONE / *DEFAULT-USERID / <name 1..8>
,COPYRIGHT = *NONE / <c-string 1..54>(...)
   <c-string 1..54>(...)
     |  YEAR = *YEAR-1990 / <c-string 4..4>
,LIBRARY = *STD / *CPLINK / *INSTALLED(...) / <filename 1..54 without-gen-vers>
   *INSTALLED(...)
     |  LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
     | ,DEFAULT-NAME = <filename 1..54 without-gen-vers>
,SUBSYSTEM-LOAD-MODE = *ANY / *STD / *ADVANCED
,REP-FILE = *STD / *NO / *INSTALLED(...) /  <filename 1..54 without-gen-vers>
   *INSTALLED(...)
     |  LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
     | ,DEFAULT-NAME = <filename 1..54 without-gen-vers> / *NONE
,REP-FILE-MANDATORY = *NO / *YES
```

Continued ➠

(part 2 of 4)

,**MES**SAGE-**FILE** = ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

  ***INSTALLED**(...)

  | **LOGICAL-ID** = <filename 1..30 without-catid-userid-gen-vers>
  | ,**DEFAULT-NAME** = <filename 1..54 without-gen-vers> / ***NONE**

,**SUBSYSTEM-INFO-FILE** = ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

  ***INSTALLED**(...)

  | **LOGICAL-ID** = <filename 1..30 without-catid-userid-gen-vers>
  | ,**DEFAULT-NAME** = <filename 1..54 without-gen-vers> / ***NONE**

,**SYNTAX-F**ILE = ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

  ***INSTALLED**(...)

  | **LOGICAL-ID** = <filename 1..30 without-catid-userid-gen-vers>
  | ,**DEFAULT-NAME** = <filename 1..54 without-gen-vers> / ***NONE**

,**DYNAMIC-CHECK-ENTRY** = ***STD** / ***NO** / <text 1..8 without-sep>

,**CRE**ATION-**TIME** = ***AT-CRE**ATION-**REQ**UEST / ***AT-SUBSYS**TEM-**CALL**(...) / ***AT-DSSM-LOAD** /
***BEFORE-DSSM-LOAD** / ***MANDATORY-AT-STARTUP** / ***BEFORE-SYSTEM-READY** /
***AFTER-SYSTEM-READY**

  ***AT-SUBSYS**TEM-**CALL**(...)

  | **ON-ACTION** = ***STD** / ***ISL-CALL** / *ANY

,**INIT-ROUTINE** = ***NO** / <text 1..8 without-sep>

,**CLOSE-CTRL-ROUTINE** = ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOPCOM-ROUTINE** = ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**DEINIT-ROUTINE** = ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOP-AT-SHUTDOWN** = ***NO** / ***Y**ES

,**INTERFACE-VERSION** = ***NO** / <text 1..8 without-sep>

,**SUBSYSTEM-HOLD** = ***ALLOW**ED / ***FORBIDDEN**

,**STATE-CHANGE-CMDS** = ***ALLOW**ED / ***FORBIDDEN** / ***BY-ADMINISTRATOR-ONLY**

,**FORCED-STATE-CHANGE** = ***FORBIDDEN** / ***ALLOW**ED

,**RESET** = ***FORBIDDEN** / ***ALLOW**ED

,**RESTART-REQUIRED** = ***NO** / ***Y**ES

,**VERSION-COEXISTENCE** = ***FORBIDDEN** / ***ALLOW**ED

,**VERSION-EXCHANGE** = ***FORBIDDEN** / ***ALLOW**ED

Continued ➠

(part 3 of 4)

```
,SUBSYSTEM-ENTRIES = *NONE / list-poss(100): <text 1..8>(...) / *BY-PROGRAM(...)
   <text 1..8>(...)
     |  MODE = *LINK / *ISL(...) / *SVC(...) / *SYSTEM-EXIT(...)
     |    *ISL(...)
     |      |  FUNCTION-NUMBER = *NONE / <integer 0..255>(...)
     |      |    <integer 0..255>(...)
     |      |      |  FUNCTION-VERSION = <integer 1..255>
     |    *SVC(...)
     |      |  NUMBER = <integer 0..255>
     |      |  ,CALL-BY-SYSTEM-EXIT = *ALLOWED / *FORBIDDEN
     |      |  ,FUNCTION-NUMBER = *NONE / <integer 0..255>(...)
     |      |    <integer 0..255>(...)
     |      |      |  FUNCTION-VERSION = <integer 1..255>
     |    *SYSTEM-EXIT(...)
     |      |  NUMBER = <integer 0..127>
     |  ,CONNECTION-ACCESS = *ALL / *SYSTEM / *SIH
     |  ,CONNECTION-SCOPE = *TASK / *PROGRAM / *FREE / *CALL / *OPTIMAL
     |  , FIRST-CONNECTION = *ALLOWED / *FORBIDDEN
   *BY-PROGRAM(...)
     |  CONNECTION-SCOPE = *TASK / *PROGRAM
,MEMORY-CLASS = *SYSTEM-GLOBAL (...) / *LOCAL-PRIVILEGED(...) / *LOCAL-UNPRIVILEGED(...) /
                *BY-SLICE(...)
   *SYSTEM-GLOBAL(...)
     |  SUBSYSTEM-ACCESS = *LOW / *SYSTEM / *HIGH
   *LOCAL-PRIVILEGED(...)
     |  SIZE = <integer 1..32767>
   *LOCAL-UNPRIVILEGED(...)
     |  SIZE = <integer 1..32767>
     |  ,SUBSYSTEM-ACCESS = *LOW / *HIGH
     |  ,START-ADDRESS = *ANY / <x-string 7..8>
```

Continued ➠

(part 4 of 4)

```
    *BY-SLICE(...)
      |  SIZE = <integer 1..32767>
,LINK-ENTRY = <text 1..8 without-sep>(...)
    <text 1..8 without-sep>(...)
      |  AUTOLINK = *FORBIDDEN / *ALLOWED
,REFERENCED-SUBSYSTEM = *NONE / list-poss(15): <structured-name 1..8>(...)
    <structured-name 1..8>(...)
      |  LOWEST-VERSION = *LOWEST-EXISTING / <c-string 3..8> / <text 3..8>
      | ,HIGHEST-VERSION = *HIGHEST-EXISTING / <c-string 3..8> / <text 3..8>
,UNRESOLVED-EXTERNALS = *FORBIDDEN / *ALLOWED
,CHECK-REFERENCE = *YES / *NO
,RELATED-SUBSYSTEM = *NONE / list-poss(100): <structured-name 1..8>(...)
    <structured-name 1..8>(...)
      |  LOWEST-VERSION = *LOWEST-EXISTING / <c-string 3..8> / <text 3..8>
      | ,HIGHEST-VERSION = *HIGHEST-EXISTING / <c-string 3..8> / <text 3..8>
```

Operands

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Specifies the name and version of the subsystem which is to be defined.

VERSION = <c-string 3..8> / <text 3..8>
The version of the subsystem must be specified in the format “[V][n]n.m[ann]”. The text elements have the following meanings:

nn	= Main version (numeric)
m	= Revision version (numeric)
ann	= Update status (a=letter, release status; nn=numeric, correction status)

INSTALLATION-UNIT =
Defines the name of the installed software unit. A value other than *NONE must be specified for all subsystems installed with IMON, and likewise if the value *INSTALLED(LOGICAL-ID=...) was defined for the operand SUBSYSTEM-LIBRARY, REP-FILE, SUBSYSTEM-INFO-FILE, MESSAGE-FILE or SYNTAX-FILE.
The syntax rules described in the “IMON“ manual [17] must be observed when defining the name.

INSTALLATION-UNIT = *NONE
Default setting: no name is assigned. This default setting is not allowed for any subsystems installed with IMON.

INSTALLATION-UNIT = *STD
The name specified via the SUBSYSTEM-NAME operand is used as the name of the installed software unit.

INSTALLATION-UNIT = <text 1..30>
Name of the installed software unit.

INSTALLATION-USERID =
Defines a user ID under which the relevant DSSM task expects the subsystem satellites (REP file, object module library, message file, syntax file and subsystem information file) if these files have not yet been assigned to a user ID, i.e. the file name was specified without a user ID.

INSTALLATION-USERID = *NONE
Default value: the files will not be expected under a specific user ID.

INSTALLATION-USERID = *DEFAULT-USERID
The files are expected under the default system ID (prefix “$.”) or under the user ID of the calling task if the subsystem is a local subsystem.

INSTALLATION-USERID = <name 1..8>
User ID under which the subsystem satellites are to be expected.
If this statement applies to an SSD object, the files will only actually be expected under the user ID specified here if no user ID was specified in the ADD-CATALOG-ENTRY statement (inclusion of subsystem definitions from the SSD object in the catalog, see page 183). The ID specified in ADD-CATALOG-ENTRY takes precedence.

COPYRIGHT =
Specifies whether or not a copyright notice is to be displayed when the subsystem is started and, if so, which one.

COPYRIGHT = *NONE
Default value: no copyright notice is to be output.

COPYRIGHT = <c-string 1..54>(...)
Text of the copyright notice which is to be output together with the creation date when the subsystem is started.

YEAR = *YEAR-1990 / <c-string 4..4>
Number of the year which is to appear in the statement as the creation date. This is not subjected to a semantic check.

LIBRARY =
Specifies the name of the program or object module library (OML) from which the object code for the subsystem is to be loaded when it is activated.

LIBRARY = *STD
Default value: when the subsystem is started, the object code will automatically be loaded from the library SYSLNK.<subsysname>.<subsysvers#>. This library is stored under the user ID under which the holder task is running. For local subsystems this is the user ID of the caller, and for global subsystems it is TSOS. “<subsysvers#>” is a three-character value consisting of the elements “mmm” (for the operand SUBSYSTEM-NAME=...(VERSION=...)).

LIBRARY = *CPLINK
The subsystem which is to be defined is linked to the BS2000/OSD control program (CP) and must have been loaded before DSSM was activated. This operand may only be used in conjunction with the operand CREATION-TIME=*BEFORE-DSSM-LOAD.

LIBRARY = *INSTALLED(...)
The library name must be determined by calling IMON (administration of installation paths). If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to the subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the program library or object module library.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
New library name if IMON-GPN is not available or if the logical name is unknown.

LIBRARY = <filename 1..54 without-gen-vers>
Fully qualified file name of the object module library from which the object code for the subsystem is to be loaded.

SUBSYSTEM-LOAD-MODE =
Specifies the load mode of the subsystem (via the BLS-DSSM interface $PBBND1).

SUBSYSTEM-LOAD-MODE = *ANY
Default value: the BLS (Binder-Loader-Starter system) is called up in STD run mode and loads the subsystem as an object module. If an error occurs during loading, BLS will be called a second time. The call takes place in ADVANCED run mode, and the subsystem is loaded as a link and load module (LLM).
Support for this operand value is provided only for the sake of compatibility.

SUBSYSTEM-LOAD-MODE = *STD
The BLS is called up in STD run mode and loads the subsystem as an object module.

SUBSYSTEM-LOAD-MODE = *ADVANCED
The BLS is called up in ADVANCED run mode and loads the subsystem as a link and load module (LLM).

REP-FILE =
Specifies whether or not system REPs are required for the subsystem which is being defined, and identifies the file in which they are stored. These correction statements are used during activation of the subsystem, and are applied solely to the modules stored and loaded in the object module library, not to other subsystems or the BS2000/OSD control program. A REP file can also be specified for modules of a nonprivileged subsystem. REP-FILE must not be specified together with LIBRARY=*CPLINK.

REP-FILE = <u>*STD</u>
Unless another file is specified, system REPs are loaded from the REP file with the name SYSREP.<subsysname>.<subsysvers#>. This REP file is stored under the user ID under which the holder task is running. For local subsystems this is the user ID of the caller, and for global subsystems it is TSOS.
"<subsysvers#>" is a three-character value consisting of the elements "mmm" specified for the operand SUBSYSTEM-NAME=...(VERSION=...).

REP-FILE = *NO
No REP files are to be processed for the subsystem.

REP-FILE = *INSTALLED(...)
The name of the REP file must be determined by calling IMON-GPN (administration of installation paths). If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the REP file.

DEFAULT-NAME =
Name of the REP file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
No new name is assigned.

REP-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the REP file from which the correction statements are to be read.

REP-FILE-MANDATORY =
Specifies whether or not a REP file declared via the REP-FILE operand is to be processed when loading the subsystem.

REP-FILE-MANDATORY = *NO
Default value: the use of a REP file is not mandatory, i.e. neither the REP file nor its entries are to be checked when the subsystem is activated. Even if the REP file can‘t be accessed or if individual correction statements are invalid, the subsystem will still be started.

REP-FILE-MANDATORY = *YES
If any of the following errors occurs during processing of the REP file, any attempt to load the subsystem will be terminated:

- the REP file is not cataloged, or it cannot be read
- checking of the correction statements reveals an error
- the name of a correction statement is invalid
- DMS reports an error during access to the NOREF file (this file is used during loading of a subsystem to prevent invalid system REPs from being logged at the operator terminal)

MESSAGE-FILE =
Specifies whether there is a subsystem-specific message file which is to be automatically activated when the subsystem is loaded. For subsystems which are defined with the creation time AT-DSSM-LOAD, a dependency relation to the MIP subsystem must be specified in the RELATED-SUBSYSTEMS operand.

MESSAGE-FILE = *NO
Default value: no message file is to be activated. This value is mandatory for all subsystems which are defined with the creation time BEFORE-DSSM-LOAD (see also the CREATION-TIME operand), as it is not possible to activate a message file as early as this.

MESSAGE-FILE = *INSTALLED(...)
The name of the message file must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to the subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the message file.

DEFAULT-NAME =
Name of the message file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
No new name is assigned.

MESSAGE-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the message file. This will be automatically activated when the subsystem is loaded (START-SUBSYSTEM command) and automatically deactivated when it is unloaded (STOP-SUBSYSTEM).

SUBSYSTEM-INFO-FILE =
Specifies whether or not a subsystem information file (SSINFO) is available. This file contains subsystem-specific data (subsystem satellites and configuration data) which cannot be processed centrally by DSSM.

SUBSYSTEM-INFO-FILE = *NO
Default value: no information file is available for the subsystem.

SUBSYSTEM-INFO-FILE = *INSTALLED(...)
The name of the information file must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the information file.

DEFAULT-NAME =
Name of the information file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
A new name is assigned.

SUBSYSTEM-INFO-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the information file. This name is automatically passed to the activation and deactivation routines (INIT-/DEINIT-/STOPCOM-/CLOSE-CTRL-ROUTINE operands) when they are called.

SYNTAX-FILE =
Specifies whether the subsystem has linked to it a syntax file which will be activated automatically when the subsystem is loaded. For subsystems which are defined with the start attribute MANDATORY-AT-STARTUP, a dependency relation to the SDF subsystem must be declared in the RELATED-SUBSYSTEMS operand.

SYNTAX-FILE = <u>*NO</u>
Default value: no syntax file is to be activated. This value is mandatory for all subsystems which are defined with the start time BEFORE-DSSM-LOAD or AT-DSSM-LOAD (see also the CREATION-TIME operand), as it is not possible to activate a syntax file as early as this.

SYNTAX-FILE = *INSTALLED(...)
The name of the syntax file must be determined by calling IMON-GPN (administration of installation paths).
If one of the subsystem satellites is referenced with a logical ID, logical IDs must be specified for all satellites belonging to this subsystem. If a logical ID is assigned, a value other than *NONE must be assigned for the INSTALLATION-UNIT operand.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logical ID of the syntax file.

DEFAULT-NAME =
Name of the syntax file if IMON-GPN is not available or if the logical ID is unknown.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
A new name is assigned.

DEFAULT-NAME = *NONE
No new name is assigned.

SYNTAX-FILE = <filename 1..54 without-gen-vers>
Fully qualified name of the syntax file which is to be automatically activated when the subsystem is loaded.

DYNAMIC-CHECK-ENTRY =
Specifies whether a dynamic identity check is to be carried out on the subsystem. For this purpose, an entry point must be specified, at which both the subsystem name (eight characters) and also the version number (four or seven characters) must be located. DSSM checks whether the identification specified in the definition agrees with the subsystem which is loaded.

DYNAMIC-CHECK-ENTRY = <u>*STD</u>
By default, the entry point specified in the LINK-ENTRY operand will be applied in carrying out the identity check.

DYNAMIC-CHECK-ENTRY = *NO
No check is to be carried out. However, this value of the operand must <u>not</u> be used for any subsystems which are loaded before DSSM is activated (CREATION-TIME=*BEFORE-DSSM-LOAD).

DYNAMIC-CHECK-ENTRY = <text 1..8 without-sep>
Name of the entry point which is to be used in applying the identity check.

CREATION-TIME =
Specifies the point in time at which activation of the subsystem (CREATE routine) is initiated. There are two separate phases during system initialization when DSSM takes over the control of system initialization after it has been called by the startup routine:

Phase 1:	The DSSM code is loaded, the DSSM task is generated and started. This task reserves class 5 memory, reads in the subsystem catalog, and starts those subsystems which have been defined with the start attributes BEFORE-DSSM-LOAD and AT-DSSM-LOAD. After these subsystems have been loaded, control of system initialization returns to the startup routine.
Phase 2:	Following a second call, all those subsystems are loaded which have been defined with the start attributes MANDATORY-AT-STARTUP, BEFORE-SYSTEM-READY and AFTER-SYSTEM-READY. In the case of the first two of these start attributes, loading of the subsystems is synchronized with the startup routine (i.e. loading must be completed), but for the last of them asynchronous loading is initiated. Control of system initialization returns to the startup routine.

If different versions of a subsystem are to be defined, it is only possible to specify the start attributes, for use in phases 1 and 2 of system initialization, for <u>one</u> of these versions.

CREATION-TIME = <u>*AT-CREATION-REQUEST</u>
Default value: the subsystem must be explicitly loaded by means of the START-SUBSYSTEM command.

CREATION-TIME = *AT-SUBSYSTEM-CALL(...)
The subsystem is to be automatically loaded when the first SVC or ISL call is made. This operand value is reserved for subsystems which are called via SVC or ISL.
If two or more versions of a subsystem are defined with this operand value, VERSION-COEXISTENCE=*ALLOWED must be specified for all of these versions and FUNCTION-NUMBER and FUNCTION-VERSION must be specified for their SVC or ISL entry points, which were declared with CONNECTION-ACCESS with a value other than *SIH.
At least one of the specified subsystems must have been declared with SUBSYSTEM-ENTRIES ..., MODE=*SVC/*ISL (corresponding to the value of the ON-ACTION operand).

ON-ACTION =
Determines what initiates automatic loading of the subsystem.

ON-ACTION = <u>*STD</u>
Default setting: loading begins when any SVC entry point belonging to the subsystem is called.

ON-ACTION = *ISL-CALL
Loading begins when any ISL entry point belonging to the subsystem is called.

ON-ACTION = *ANY
Loading begins when any SVC or ISL entry point belonging to the subsystem is called.

CREATION-TIME = *AT-DSSM-LOAD
The subsystem is to be loaded during system initialization (phase 1) under the control of the DSSM task.
The subsystem must be a privileged one, and may only have address or dependency relations to subsystems which are also defined with this start attribute or with the start attribute BEFORE-DSSM-LOAD.
The files for this subsystem must be held on the home pubset under the TSOS user ID, because at the time of startup the user catalog is not accessible and IMPORT-PUBSET processing has not been completed.
Specification of a syntax file is not permitted for these subsystems.

CREATION-TIME = *BEFORE-DSSM-LOAD
The subsystem is to be loaded during system initialization (phase 1), but not under the control of the DSSM task.
Such subsystems are linked to the control program, and do not need to be synchronized with the DSSM task when activated. However, after the subsystem is loaded it again runs under the control of DSSM and can, from the user’s point of view, be controlled in the same way as other subsystems.
No address or dependency relations are possible to subsystems which are defined with any other start attribute. Nor is it permissible to link in a message or syntax file. All job entries (SUBSYSTEM-ENTRIES operand) must be declared, because DSSM creates the (privileged) connection to these job entries. Responsibility for ensuring that at least one version of this subsystem is available at any given time (e.g. PLAM) rests entirely with the subsystem developer.

The name of the link context for these subsystems must be unique because DSSM must also honor an unload request even if DSSM has not loaded the subsystem code. An entry point (operand DYNAMIC-CHECK-ENTRY) must have been specified.

CREATION-TIME = *BEFORE-SYSTEM-READY
The subsystem is to be loaded during system initialization (phase 2). Activation is initiated synchronously; not until loading is complete (or a load error occurs) does control return to the startup routine, which can then report “SYSTEM READY”.
The subsystem must be privileged, and may only have address or dependency relations to subsystems defined with the same attribute or with the start attribute BEFORE-DSSM-LOAD, AT-DSSM-LOAD or MANDATORY-AT-STARTUP. The files for this subsystem must be cataloged on the home pubset.
If a nonprivileged subsystem is declared with this operand value, it is automatically given the value *AFTER-SYSTEM-READY. SSCM issues a message.

CREATION-TIME = *MANDATORY-AT-STARTUP
The subsystem must be loaded during system initialization (phase 2). Activation is initiated synchronously - as in the case of BEFORE-SYSTEM-READY. By contrast with the latter however, loading of the subsystem must in this case be completed **successfully**. Otherwise a message is passed to the startup routine reporting that a mandatory subsystem could not be loaded. In this case, the startup routine will decide whether processing should continue or be terminated.
The subsystem must be privileged, and may only have address or dependency relations to subsystems defined with the same attribute or with the start attribute BEFORE-DSSM-LOAD or AT-DSSM-LOAD. The files for this subsystem must be cataloged on the home pubset.

CREATION-TIME = *AFTER-SYSTEM-READY
The loading of this subsystem is to be initiated during system initialization (phase 2). Execution of this routine is not synchronized with the startup routine, which can report “SYSTEM READY” before subsystem loading is complete.
The subsystem may only have address or dependency relations to subsystems defined with the same attribute or with the start attribute BEFORE-DSSM-LOAD, AT-DSSM-LOAD, MANDATORY-AT-STARTUP or BEFORE-SYSTEM-READY. The files for this subsystem must be cataloged on the home pubset.

INIT-ROUTINE =
Specifies whether there is an initialization routine for the subsystem which must be performed when it is started or resumed. In this case, the name of an entry point must be known, and DSSM will delegate initialization to the holder task of the subsystem concerned. It is strongly recommended that an entry point be defined for all subsystems which have the start attribute BEFORE-DSSM-LOAD. During loading of the subsystem (i.e. when the initialization routine is carried out), the subsystem is then informed that DSSM can assume control over the opening and closing of relations.
An INIT routine must be declared with RESTART-REQUIRED=*YES.

INIT-ROUTINE = *NO
Default value: no initialization routine is run.

INIT-ROUTINE = <text 1..8 without-sep>
Name of the entry point to the initialization routine.
In the holder task, control is passed to the initialization routine, so that the subsystem can initialize itself. To permit this, it is passed:

- the name and version of the subsystem, as defined in SSCMCAT
- the name of the SSINFO file, if one was specified in the SUBSYSTEM-INFO-FILE operand
- the address of the entry point specified during loading and linking (LINK-ENTRY operand)
- the link context name used by the dynamic binder loader
- the name of the memory pool (for subsystems in class 5 or class 6 memory), in order that the subsystem can refer to its own selectable units/load units during dynamic loading
- the name of the message file
- the address of the SUBSYSTEM-PARAMETER operand, if a string has been specified in the START-SUBSYSTEM command

At the end of initialization, a return message is expected from the subsystem, indicating whether initialization was successful and whether the holder task is to be used as a work task (as specified in the ASSIGN-HOLDER-TASK statement, page 193). Depending on what is reported here, the task will from then on be controlled by DSSM or by the subsystem.

CLOSE-CTRL-ROUTINE =
Specifies whether a special routine is to be run for the subsystem if it is suspended or deactivated.
If a subsystem is deactivated (by a STOP-SUBSYSTEM or HOLD-SUBSYSTEM command), DSSM passes control to this routine at the identified entry point in the holder task, or (for *DYNAMIC) this is reported via a bourse or FITC link (as determined by return messages during initialization).

The parameters passed are the same ones as are passed for the INIT-ROUTINE operand. Branching to this routine ensures that a connection to the subsystem still exists.
If a CLOSE-CTRL routine exists, it is possible to change versions without interrupting the BS2000 session. At any given point in time, there is exactly one valid version (either the old version is still available, or the new version has already become available). Without a CLOSE-CTRL routine, changing versions always entails interrupting the connection so that the STOPCOM routine of the old version and the INIT routine of the new version can execute (see also page 45).

CLOSE-CTRL-ROUTINE = *NO
Default value: the subsystem designates no routine which is to be run when the subsystem is deactivated or suspended.

CLOSE-CTRL-ROUTINE = *DYNAMIC
This routine is called up via the bourse or FITC. The subsystem passes the required parameters to the CLOSE-CTRL routine dynamically at the end of the INIT routine, and DSSM is informed of the identity of the bourse or FITC port.
In order for the CLOSE-CTRL routine to run, an INIT routine (INIT-ROUTINE operand) and a STOPCOM routine (operand STOPCOM-ROUTINE=*NO/*DYNAMIC) must have been specified.
The holder task of the subsystem must be a work task when the CLOSE-CTRL routine is called (ASSIGN-HOLDER-TASK statement, page 193).

CLOSE-CTRL-ROUTINE = <text 1..8 without-sep>
Name of the entry point of the relevant subsystem routine.

STOPCOM-ROUTINE =
Specifies whether the subsystem incorporates a routine which can carry out active termination of tasks.
If a subsystem is deactivated (by a STOP-SUBSYSTEM or HOLD-SUBSYSTEM), then DSSM passes control to this routine at the identified entry point in the holder task, or (for *DYNAMIC) this is reported via a bourse or FITC link (as determined by return messages during initialization).
The parameters which are passed are the same as for the INIT-ROUTINE operand. Branching to this routine ensures that calling tasks will no longer be connected to the subsystem. Tasks which still have outstanding call relations to the subsystem are unaffected by this.
If CLOSE-CTRL-ROUTINE=*DYNAMIC is declared, the operand STOPCOM-ROUTINE=*NO/*DYNAMIC must be specified.

STOPCOM-ROUTINE = *NO
Default value: the subsystem concerned incorporates no such routine.

STOPCOM-ROUTINE = *DYNAMIC
This routine is called via the bourse or FITC. The subsystem passes the required parameters to the STOPCOM routine dynamically at the end of the CLOSE-CTRL routine or (if none exists) at the end of the INIT routine. DSSM is informed of the identity of the bourse or FITC port.
A prerequisite for the use of the STOPCOM routine is than an INIT routine has been specified (INIT-ROUTINE operand). When the STOPCOM routine is called, the holder task for the subsystem must be used as a work task (ASSIGN-HOLDER-TASK statement, page 193).

STOPCOM-ROUTINE = <text 1..8 without-sep>
Name of the entry point to the subsystem routine concerned.

DEINIT-ROUTINE =
Specifies whether the subsystem incorporates a routine which can carry out deinitialization of the subsystem. This deinitialization routine causes the resources which were requested by the subsystem (memory, files, devices) to be returned.
If a subsystem is deactivated (by a STOP-SUBSYSTEM or HOLD-SUBSYSTEM command), then DSSM passes control to this routine at the identified entry point in the holder task, or (for *DYNAMIC) this is reported via a bourse or FITC link (as determined by return messages during initialization).
If a subsystem is defined with an INIT routine and a CLOSE-CTRL routine, a DEINIT routine - with the same operand value as the CLOSE-CRTL routine - must be specified.
The parameters which are passed are the same as for the INIT-ROUTINE operand.
Branching to this routine ensures that calling tasks will no longer be connected to the subsystem and all existing call relations to the subsystem are deleted.

DEINIT-ROUTINE = *NO
Default value: the subsystem concerned does not incorporate a deinitialization routine to request the release of resources.

DEINIT-ROUTINE = *DYNAMIC
The routine is called via the bourse or FITC.The subsystem passes the required parameters to the DEINIT routine dynamically at the end of the STOPCOM routine or, if none exists, at the end of the CLOSE-CTRL routine or, if neither a STOPCOM nor a CLOSE-CTRL routine is incorporated, at the end of the INIT routine. DSSM is informed of the identity of the bourse or FITC port.
A prerequisite for the use of the DEINIT routine is than an INIT routine has been specified (INIT-ROUTINE operand). When the DEINIT routine is called, the holder task for the subsystem must be used as a work task (ASSIGN-HOLDER-TASK statement, page 193).

DEINIT-ROUTINE = <text 1..8 without-sep>
Name of the entry point to the subsystem routine concerned.

STOP-AT-SHUTDOWN =
Specifies whether the subsystem is to be unloaded automatically at shutdown after the user tasks have terminated.

STOP-AT-SHUTDOWN = *NO
Default value: the subsystem will not be unloaded automatically.
This parameter should not be specified for subsystems which have address relations to other subsystems which are defined with STOP-AT-SHUTDOWN=*YES.

STOP-AT-SHUTDOWN = *YES
The subsystem will be unloaded automatically at shutdown.
This specification will be ignored if no STOPCOM, DEINIT or CLOSE-CRTL routine is specified.

INTERFACE-VERSION =
Identifies the entry point via which DSSM can access the interface version which is to be used for calling the INIT, DEINIT, STOPCOM or CLOSE-CTRL routine.
This operand is mandatory for subsystems for which an entry point was specified (INIT, CLOSE-CTRL, STOPCOM, DEINIT routine).

INTERFACE-VERSION = *NO
Default value: the subsystem does not call a INIT, DEINIT, STOPCOM or CLOSE-CTRL routine.

INTERFACE-VERSION = <text 1..8 without-sep>
Name of the entry point. The entry point points to the standard header where the interface version is stored. The standard header is generated by calling the macro $ESMINT(I) with MF=I/L. This operand is mandatory for subsystems for which an INIT, DEINIT, STOPCOM or CLOSE-CTRL routine was defined.

SUBSYSTEM-HOLD =
Specifies whether the subsystem which is loaded may be suspended or unloaded.

SUBSYSTEM-HOLD = *ALLOWED
Default value: the subsystem which is loaded may be suspended and unloaded. The commands HOLD-SUBSYSTEM and STOP-SUBSYSTEM are permissible for this subsystem.

SUBSYSTEM-HOLD = *FORBIDDEN
The commands HOLD-SUBSYSTEM and STOP-SUBSYSTEM must not be used for this subsystem; it will only be unloaded at shutdown - as specified by the STOP-AT-SHUTDOWN operand.
Unloading the subsystem by replacing it with another subsystem entails no interruption.

STATE-CHANGE-CMDS =
Specifies whether or not the DSSM commands for controlling the subsystem (START-SUBSYSTEM, STOP-SUBSYSTEM, HOLD-SUBSYSTEM and RESUME-SUBSYSTEM) may be used during a session.
If a change is made from one version of a subsystem to another, the value specified for STATE-CHANGE-CMDS for the version being replaced is ignored.

STATE-CHANGE-CMDS = <u>*ALLOWED</u>
Default value: the commands may be used from the operator terminal and under the privileged user ID (the user ID which has the SUBSYSTEM-MANAGEMENT system privilege).

STATE-CHANGE-CMDS = *FORBIDDEN
The commands must not be used - neither from the operator terminal nor under the privileged user ID.

STATE-CHANGE-CMDS = *BY-ADMINISTRATOR-ONLY
The commands may only be used under the privileged user ID; the commands are not available to the operator at the operator terminal.

FORCED-STATE-CHANGE =
Specifies whether use of the operand FORCED=*YES is permitted within the commands STOP-SUBSYSTEM and HOLD-SUBSYSTEM. This function can be used to force the unconditional deactivation of the subsystem.

FORCED-STATE-CHANGE = <u>*FORBIDDEN</u>
Default value: it is not possible to force deactivation of the subsystem. DSSM will reject any use of the FORCED operand in the commands concerned, and will issue a corresponding error message.

FORCED-STATE-CHANGE = *ALLOWED
The operand FORCED=*YES may be used for this subsystem.
This operand value must not be used in conjunction with SUBSYSTEM-HOLD=*FORBIDDEN.

RESET =
Specifies whether the operand RESET=*YES is permitted within the commands START-SUBSYSTEM and RESUME-SUBSYSTEM. This function can be used to force the unconditional loading or resumption of the subsystem, even if the state of the subsystem is currently IN-DELETE or IN-HOLD.

RESET = <u>*FORBIDDEN</u>
Default value: it is not possible to force the activation of the subsystem. DSSM will reject the use of the RESET operand in the commands concerned, and will issue a corresponding error message.

RESET = *ALLOWED
The operand RESET=*YES may be used for this subsystem.
This operand value must not be specified together with SUBSYSTEM-HOLD=*FORBIDDEN.

RESTART-REQUIRED =
Specifies whether the initialization routine for the subsystem is to be executed if the holder task terminates abnormally.

RESTART-REQUIRED = <u>*NO</u>
Default value: the initialization routine is not used to restart the subsystem.

RESTART-REQUIRED = *YES
If the holder task terminates abnormally, the initialization routine should be used. Provision must have been made in the INIT-ROUTINE operand for executing this routine.

VERSION-COEXISTENCE =
Specifies whether more than one version of the same subsystem may be active simultaneously.

VERSION-COEXISTENCE = <u>*FORBIDDEN</u>
Default value: the current version of the subsystem cannot coexist with another version of the same subsystem.

VERSION-COEXISTENCE = *ALLOWED
The current version of the subsystem can coexist with another version of the same subsystem (coexistence mode).
In the definition of the job entry point (SUBSYSTEM-ENTRIES operand), indirect links via system exit routines must not have been specified. If different versions of the same subsystem are loaded and the same job entry point is defined for these, the link which is implemented is always to the highest loaded version of the subsystem.
If coexistent subsystems access coexistent syntax files, the latter must have been declared in the SSD object and cannot be administered by SDF.
However, where the links are via SVC and ISL, it is possible to select a version using the operands FUNCTION-NUMBER and FUNCTION-VERSION.

VERSION-EXCHANGE =
Specifies whether a subsystem may be loaded in exchange mode. Exchange mode allows the temporary coexistence of two versions of the same subsystem. If version B of a subsystem is loaded whilst version A of the subsystem is already active, all new callers will be connected to version B. Jobs which are connected to version A will still be processed. When all the jobs which use version A have been processed, this will automatically be terminated.
In the definition it should be noted that the “old” version which is being replaced must not be dependent on the “new” version which replaces it.

VERSION-EXCHANGE = *FORBIDDEN
Default value: the current version of the subsystem must not be replaced.

VERSION-EXCHANGE = *ALLOWED
Exchange mode, which allows the temporary coexistence of two subsystems, is permitted for the current subsystem version.

SUBSYSTEM-ENTRIES =
Specifies the entry points (job entries) which are linked to the subsystem. Up to 100 entry points can be declared. If more than 100 entry points are desired for a subsystem, the additional ones can be defined by means of the statement MODIFY-SUBSYSTEM-ATTRIBUTES (for a subsystem in the catalog) or the statement ADD-SUBSYSTEM-ENTRIES (for a subsystem in an SSD object).

SUBSYSTEM-ENTRIES = *NONE
Default value: no new job entries are to be declared.

SUBSYSTEM-ENTRIES = list-poss(100): <text 1..8>
Declares the names of the entry points for a maximum of 100 job entries; the type of each of these must be defined in the substructures.

MODE =
Defines the type of a job entry point which is defined for the subsystem.

MODE = *LINK
Default value: the job entry point cannot be accessed by indirect linkage, but only by using a CONNECT relation through an external linkage editor symbol.
In the case of different versions of the same subsystem which use the same external linkage editor symbol, DSSM automatically sets up the link to the highest loaded version of the subsystem.

MODE = *ISL(...)
The job entry is effected by indirect linkage via System Procedure Linkage (for privileged subsystems only). If the specification includes in addition a function and version number for the ISL entry point, the combination of entry point name, function and version numbers must not match any other combination for the various other subsystems in the catalog or the various versions of the same subsystem (if VERSION-COEXISTENCE=*ALLOWED is specified).
For different subsystems, if the job entry point is to be accessed by the same ISL entry point, they must be uniquely identified by specifying the function and version numbers.
In the case of different versions of the same subsystem which use the same ISL entry point, then - if the function and version numbers are not specified - DSSM will automatically set up a connection to the highest loaded version of the subsystem.
In the case of different versions of the same subsystem which use the same ISL entry point and for which the function and version numbers are not equal to *NONE, the version to which the connection is set up will be selected in accordance with the function

and version numbers stored in the standard header of the caller’s parameter list.
The value *ALL for the CONNECTION-ACCESS operand is not permissible for ISL entry points.

FUNCTION-NUMBER =
Specifies whether a particular function and version number of the ISL entry point is to be addressed, as the same ISL entry point can be used by different functions.

FUNCTION-NUMBER = *NONE
Default value: no particular function or version number is to be addressed.

FUNCTION-NUMBER = <integer 0..255>(...)
Number of the ISL entry point. The version must be nominated in the substructure which follows.

FUNCTION-VERSION = <integer 1..255>
Version of the specified ISL function number.

MODE = *SVC(...)
Job entry is to be effected by an indirect connection using a supervisor call (SVC).
If the specification includes in addition a function and version number for the SVC entry point, the combination of entry point name, function and version numbers must not match any other combination for the various other subsystems in the catalog or the various versions of the same subsystem (if VERSION-COEXISTENCE=*ALLOWED is specified).
For different subsystems, if the job entry point is to be accessed by the same SVC, they must be uniquely identified by specifying the function and version numbers.
In the case of different versions of the same subsystem which use the same SVC, then - if the function and version numbers are not specified - DSSM will automatically set up a connection to the highest loaded version of the subsystem.
In the case of different versions of the same subsystem which use the same SVC and for which the function and version numbers are not equal to *NONE, the version to which the connection is set up will be selected in accordance with the function and version numbers stored in the standard header of the caller’s parameter list.
If this operand value is specified, it is better to set the operand CONNECTION-ACCESS to the value *SYSTEM, instead of to *ALL.

NUMBER = <integer 0..255>
Number of the SVC via which job entry is to be effected. No SVC number greater than 191 may be used in conjunction with CONNECTION-ACCESS=*ALL.

CALL-BY-SYSTEM-EXIT =
Defines whether the specified SVC number may be called from within system exit routines.

CALL-BY-SYSTEM-EXIT = *ALLOWED
Default value: system exit routines are permitted to call the specified SVC number.

CALL-BY-SYSTEM-EXIT = *FORBIDDEN
System exit routines are not permitted to call the specified SVC number.

FUNCTION-NUMBER =
Specifies whether a particular function and version number of the SVC entry point is to be addressed, as the same SVC entry point can be used by different functions.

FUNCTION-NUMBER = <u>*NONE</u>
Default value: no particular function or version number is to be addressed.

FUNCTION-NUMBER = <integer 0..255>(...)
Number of an SVC entry point. The version must be nominated in the substructure which follows.

FUNCTION-VERSION = <integer 1..255>
Version of the specified SVC function number.

MODE = SYSTEM-EXIT(...)
Job entry is to be effected by an indirect connection using system exit routines.
This operand must not be used in conjunction with CONNECTION-ACCESS=*ALL.

NUMBER = <integer 0..127>
Number of the system exit routine.

CONNECTION-ACCESS =
Specifies the access authorization (privileges) required by the subsystem.

CONNECTION-ACCESS = <u>*ALL</u>
Default value: privileged and nonprivileged program runs may access the subsystem.
This operand value must not be used in conjunction with MODE=*SYSTEM-EXIT/*ISL/*SVC (with an SVC number greater than 191).

CONNECTION-ACCESS = *SYSTEM
Only privileged program runs may access the subsystem.

CONNECTION-ACCESS = *SIH
Only tasks running in the SIH processor state may access the subsystem.
The subsystem called also runs in the SIH processor state, i.e. it is uninterruptible.

This operand value is permissible only for subsystems for which the entry point is defined via:

– System Procedure Linkage (MODE=*ISL(FUNCTION-NUMBER=*NONE))
– CONNECTION-SCOPE=*OPTIMAL
– MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=*SYSTEM)

CONNECTION-SCOPE =
Identifies the event which will call up the automatic cleardown of the connection to the specified subsystem job entry.

CONNECTION-SCOPE = *TASK
Default value: the connection will be cleared when the task terminates.

CONNECTION-SCOPE = *PROGRAM
The connection will be cleared when the program terminates, or before.
Only CONNECTION-SCOPE=*PROGRAM may be specified in conjunction with MEMORY-CLASS=*LOCAL-UNPRIVILEGED.
This operand value is recommended for subsystems which were declared with SUBSYSTEM-ACCESS=*LOW/*HIGH or MEMORY-CLASS=*BY-SLICE.

CONNECTION-SCOPE = *FREE
DSSM is not to carry out any checking of the connections to the job entry point. The connection will not be automatically cleared - unless explicitly requested. To avoid problems or possible errors when the subsystem is being unloaded, the connections must be managed by the subsystem itself.

CONNECTION-SCOPE = *CALL
On return from this job entry point, DSSM will automatically clear the connections. This operand value is only available with subsystems for which the job entry is defined by means of System Procedure Linkage (ISL) or supervisor calls (SVC).

CONNECTION-SCOPE = *OPTIMAL
The subsystem is deactivated or suspended when there are no further tasks with a connection to this entry point.
A routine with an entry point defined with *OPTIMAL must be terminated with RETURN.
If an entry point of a subsystem is defined with CONNECTION-SCOPE=*OPTIMAL, all of its entry points should be defined in the subsystem catalog with MODE≠*LINK.
While a subsystem is deactivated or suspended, no call of the subsystem with CONNECTION-SCOPE=*OPTIMAL is accepted.

FIRST-CONNECTION =
Determines whether or not first connection of the task to the specified job entry point in the subsystem is allowed. At least one job entry point of a subsystem must be defined with FIRST-CONNECTION=*ALLOWED.

FIRST-CONNECTION = *ALLOWED
Default value: first connection to the specified job entry point is allowed.

FIRST-CONNECTION = *FORBIDDEN
Connection to the specified job entry point via SVC or ISL is not allowed if the task has not yet been connected to another job entry point belonging to the subsystem.
It is not permitted to specify this operand value for job entry points that have been defined with MODE=*LINK/*SYSTEM-EXIT or CONNECTION-ACCESS=*SIH.

SUBSYSTEM-ENTRIES = *BY-PROGRAM(...)
The entry points of the specified subsystem are supplied dynamically from the BLS name list at load time instead of statically from the catalog. A prerequisite for this functionality is the use of BLSSERV as of version 2.1, that supports using Extended External Names (EEN names) as entry points for DSSM subsystems.

The value cannot be specified together with the SUBSYSTEM-ACCESS=*SYSTEM operand.

The MODE, CONNECTION-ACCESS and FIRST-CONNECTION operands are supplied with the default values if the entry points are defined with this value.

i The value *BY-PROGRAM is not restricted to EEN names. It can also be specified for subsystems that have a large number of entry points.

CONNECTION-SCOPE = <u>*TASK</u> / *PROGRAM
The connection is shut down at task or program termination.

MEMORY-CLASS =
Specifies the subsystem-specific address space in which the subsystem is to be loaded. System administration can use this operand to define the address space valid for the subsystem concerned so as to meet the special requirements of the installation.

MEMORY-CLASS = <u>*SYSTEM-GLOBAL</u>(...)
Default value: the subsystem will be loaded in class 3 or class 4 memory. Resident CSECTs will be given class 3 memory, all others will be given pageable class 4 memory.

SUBSYSTEM-ACCESS =
Identifies the access authorization (privileges) and location of the requested memory.

SUBSYSTEM-ACCESS = <u>*LOW</u>
Default value: nonprivileged address space below the 16-Mbyte boundary is allocated.

SUBSYSTEM-ACCESS = *SYSTEM
Subsystems declared with this operand value are privileged subsystems to which privileged address space above the 16-Mbyte boundary is allocated.
This operand value is mandatory for subsystems whose entry point is declared via SVC (MODE=*SVC) or for which an INIT, STOPCOM, DEINIT or CLOSE-CTRL routine is declared. It is not permitted with CONNECTION-ACCESS=*ALL and MODE=*LINK.
The value cannot be specified together with an entry point that was specified together with CONNECTION-ACCESS=*ALL or MODE=*LINK.
The value cannot be specified together with the SUBSYSTEM-ENTRIES=*BY-PROGRAM operand.

SUBSYSTEM-ACCESS = *HIGH
Nonprivileged address space up to 2 Gbytes is allocated.

MEMORY-CLASS = *LOCAL-PRIVILEGED(...)
The subsystem is given a memory pool in nonprivileged class 5 memory, located below the 16-Mbyte boundary.
This specification is suitable for nonprivileged subsystems which demand a relatively large amount of address space (approx. 1 Mbyte) and have to be set up below the 16-Mbyte boundary. These subsystems are loaded in memory pools at the same address, in order to manage the use of the limited address space below 16 Mbytes.
Although such subsystems are loaded in parallel in the same address space, they cannot be used simultaneously by the same task (see also the SEPARATE-ADDRESS-SPACE statement, page 241).
The subsystem must not contain any resident CSECTs, as otherwise a later attempt to activate it will be aborted.

SIZE = <integer 1..32767>
Size of the required address space (in 4Kbyte pages) for the memory pool in class 5 memory. This value should be set at least high enough to ensure that the subsystem and any selectable units/load units which it may load dynamically can be loaded in their entirety. The upper limit is generation-specific.

MEMORY-CLASS = *LOCAL-UNPRIVILEGED(...)
The subsystem is given a memory pool in nonprivileged class 6 memory. This specification is reserved for subsystems which can be executed like a program.
In keeping with this, their access authorization (privileges) must be defined with the value *ALL in the CONNECTION-ACCESS operand.
This operand value must not be specified together with an entry point which was defined with CONNECTION-ACCESS=*SYSTEM.

The subsystem must not contain any resident CSECTs, as otherwise a later attempt to activate it will be aborted.
If this operand value is specified, only CONNECTION-SCOPE=*PROGRAM is permissible for clearing the connection to the specified subsystem entry point.

SIZE = <integer 1..32767>
Size of the required address space (in 4Kbyte pages) for the memory pool in class 6 memory. This value should be set at least high enough to ensure that the subsystem and any selectable units/load units which it may load dynamically can be loaded in their entirety. The upper limit is generation-specific.

SUBSYSTEM-ACCESS =
Identifies the location of the requested memory space.

SUBSYSTEM-ACCESS = *LOW
Default value: nonprivileged address space below the 16-Mbyte boundary is allocated.
Because this specification is suitable for subsystems which can be executed like programs, it is advisable additionally to specify CONNECTION-SCOPE=*PROGRAM.

SUBSYSTEM-ACCESS = *HIGH
Nonprivileged address space up to 2 Gbytes is allocated.
Because this specification is suitable for subsystems which can be executed like programs, it is advisable additionally to specify CONNECTION-SCOPE=*PROGRAM.

START-ADDRESS =
Defines the start address in class 6 memory.

START-ADDRESS = <u>*ANY</u>
Default value: the location of the subsystem in class 6 memory will be determined by DSSM.

START-ADDRESS = <x-string 7..8>
Start address in the segment raster at which the subsystem’s start address is to be located. The value specified must be an 8-character hexadecimal constant which is a multiple of X'100000'.

MEMORY-CLASS = *BY-SLICE(...)
The specified subsystem is a nonprivileged subsystem and consists of an LLM, which in turn consists of a shareable code (program area) and a non-shareable code (data area). The program area is loaded into the shareable address space (this corresponds to MEMORY-CLASS=*SYSTEM-GLOBAL). The data area is loaded into the user address space of the holder task and is copied into the private user address spaces of the connected tasks at the same address.
The following values must be specified together with MEMORY-CLASS=*BY-SLICE: SUBSYSTEM-LOAD-MODE=*ADVANCED and CONNECTION-ACCESS=*ALL.
The value can only be specified together with the MODE=*LINK and CONNECTION-SCOPE=*TASK or *PROGRAM operands.

SIZE = <integer 1.32767>
Specifies the size of the requested memory space for the data area in 4K pages.
The value chosen here must be sufficiently large to allow the subsystem and, if appropriate, selectable units/load units dynamically loaded by the subsystem to be loaded in full. The upper limit is dependent on generation.

LINK-ENTRY = <text 1..8 without-sep>(...)
Defines the name of the object module/ENTRY/CSECT required for loading (for the operand in the call of the $PBBND1 macro to the dynamic binder loader DBL). The subsystem must be completely loaded by this ENTRY (if necessary, using autolink).

AUTOLINK =
Controls invocation of the autolink function during linking and loading.
The linkage editor’s autolink function permits the automatic insertion of modules which are not explicitly inserted by appropriate statements. The main purpose of this function is to save users of higher-level programming languages from having to make explicit

statements to insert individually the numerous modules of the runtime system which are required. Further details of the autolink function will be found in the "BLSSERV" manual [4].

The autolink function can also be implicitly circumvented if the first external reference encountered during linkage editing of the object module which is to be loaded points to a prelinked module. The advantage of this approach is that the paging behavior when the subsystem is later executed can be optimized at this preliminary stage (during linkage editing). In addition, errors during linkage editing can be avoided in this way.

AUTOLINK = *FORBIDDEN /*ALLOWED
Default value: the autolink function will be suppressed or allowed.

REFERENCED-SUBSYSTEM =
Specifies whether there is a list of subsystems to which there are address relations, and which is to be used for resolving external references

REFERENCED-SUBSYSTEM = *NONE
Default value: the subsystem which is being defined has no external references.

REFERENCED-SUBSYSTEM = list-poss(15): <structured-name 1..8>
The system which is being defined has external references to a maximum of 15 other subsystems; these subsystems must be used in resolving the external references. If any of the subsystems nominated here is missing at the time of activation or deactivation (and if a check of the external references has also been requested by the operand CHECK-REFERENCE=*YES), the action will be aborted.
It is also possible to address the BS2000/OSD control program via these external references - using the name CP. In the substructure which follows, it is possible to specify either exactly one version, or a range of versions within which all versions are to be referred to. If a version range list is used to limit the version of the CP subsystem, DSSM checks the compatibility of the current CP version against the versions in the range list. The subsystem will only be loaded if it is a compatible version.

The following restrictions should be noted when specifying subsystems to which there are address relations:

– No address relations may be declared to subsystems which have the attribute MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED/*BY-SLICE.
– Subsystems which have the attribute STOP-AT-SHUTDOWN=*NO may have address relations only to other subsystems which also have this attribute.
– If the attribute SUBSYSTEM-ACCESS=*SYSTEM is specified for the subsystem being defined, subsystems defined with SUBSYSTEM-ACCESS=*LOW or SUBSYSTEM-ACCESS=*HIGH must not be addressed.
– As a rule, a nonprivileged subsystem must not have any address relations to the control program (CP).

- If a reference is made to a subsystem which has at least one version that may be operated in coexistence or exchange mode, a unique version must be specified.
- Any address relations must be defined in accordance with the start attributes (CREATION-TIME operand); i.e. a subsystem may only have relations to other subsystems if these are started at the same time or earlier.

UNRESOLVED-EXTERNALS =
Defines how the load procedure is to behave if there are unresolved external references.

UNRESOLVED-EXTERNALS = *FORBIDDEN
Default value: if unresolved external references occur, the load procedure will be terminated.

UNRESOLVED-EXTERNALS = *ALLOWED
The load procedure will be continued, unresolved external references will be given the value X'FFFFFFFF'.

CHECK-REFERENCE =
Defines whether or not the subsystems specified in the REFERENCED-SUBSYSTEM operand are to be checked in respect of their status and availability.

CHECK-REFERENCE = *YES
Default value: the referenced subsystems will be checked. If any of them is missing, DSSM will abandon the activation or unloading of the subsystem.

CHECK-REFERENCE = *NO
DSSM is not to carry out any check. However, even if the user generates complex subsystems with this statement, DSSM will still perform the requested functions **in spite of** the risk of conflicts:

- the START-SUBSYSTEM command will load the specified subsystem, even if there is a subsystem to which it has defined relations and which is not yet fully loaded;
- the commands RESUME-SUBSYSTEM, STOP-SUBSYSTEM and HOLD-SUBSYSTEM will be executed by DSSM without any checks being made on relations and dependencies.

RELATED-SUBSYSTEM =
Defines whether or not there is a list of subsystems for which there are dependency relations.

RELATED-SUBSYSTEM = *NONE
Default value: there are no dependency relations for the subsystem which is being defined.

RELATED-SUBSYSTEM = list-poss(100): <structured-name 1..8>
The subsystem which is being defined has dependency relations to up to 100 other subsystems, without which the subsystem currently being defined cannot function.
It is also permissible to define dependency relations to the BS2000/OSD control program

(CP). The rules and restrictions which apply when doing so are analogous to those for address relations, where they are described in more detail (REFERENCED-SUBSYSTEM operand).
A dependency relation always points to a single version of a subsystem. In the substructure which follows, it is possible to specify either exactly one version, or a range of versions within which all versions are to be referred to.

The following general restrictions should be noted when specifying dependent subsystems:

– The relation which is defined must not contain closed loops. A loop arises if subsystem A is dependent on B, B is dependent on C and this is in turn dependent on A.
– If the subsystem which is being defined has been given the attribute MEMORY-CLASS=*SYSTEM-GLOBAL, it is not permissible to address any subsystems defined with MEMORY-CLASS=*LOCAL-PRIVILEGED or *LOCAL-UNPRIVILEGED.
– For subsystems which have the attribute SUBSYSTEM-ACCESS=*SYSTEM, no dependency relations may be defined to subsystems to which SUBSYSTEM-ACCESS=*LOW or SUBSYSTEM-ACCESS=*HIGH or MEMORY-CLASS=*BY-SLICE applies.
– The dependency relations must be defined to correspond to the start attributes (CREATION-TIME operand); i.e. a subsystem may only be dependent on subsystems which are started at the same time or earlier.

LOWEST-VERSION =
Specifies the lowest value (lowest version) in the subsystem version range list.

LOWEST-VERSION = *LOWEST-EXISTING
The lowest version in the catalog is to be addressed.

LOWEST-VERSION = <c-string 3..8> / <text 3..8>.
Version of the subsystem which is to be used as the lower limit of the range of versions.

HIGHEST-VERSION =
Specifies the upper value (highest version) in the subsystem version range list.

HIGHEST-VERSION = *HIGHEST-EXISTING
The highest version in the catalog is to be addressed.

HIGHEST-VERSION = <c-string 3..8> / <text 3..8>.
Version of the subsystem which is to be used as the upper limit of the range of versions.

SHOW-CATALOG
Show subsystem configuration

Function

This statement can be used to output the contents of a subsystem configuration, which is stored in a subsystem catalog, to either the screen or a file, as required.
The output always contains the name of the subsystem catalog and information on whether and how the catalog is saved in the event of an error (this behavior is set in the statement SAVE-CATALOG ..., FORCED=...).

SHOW-CATALOG will be rejected if the specified file or subsystem name or the specified subsystem version does not exist. If the subsystem catalog is empty, a warning message will be issued and the statement aborted.

If the specified file name does not correspond to that of the catalog which is currently open, the following message is output:

```
SCM0011   DO YOU REALLY WANT TO OVERWRITE MEMORY CATALOG '(&00)'? REPLY (Y/N)
```

If the reply is **Y**, the virtual definitions in the current catalog will be lost. If the reply is **N**, execution of the SHOW-CATALOG statement will be aborted. With the aid of SAVE-CATALOG, the user can store in a file all subsystem definitions not yet saved.

If the CHECK-CATALOG statement was not used to run a consistency check beforehand, SHOW-CATALOG may detect consistency errors in the subsystem catalog.

Format

```
SHOW-CATALOG
---------------------------------------------------------------------
 CATALOG-NAME = *CURRENT / <filename 1..54 without-gen-vers>
,CONTAINED-SUBSYSTEMS = *NO / *YES
,SUBSYSTEM-NAME = *ALL / <structured-name 1..8>(...)
    <structured-name 1..8>(...)
      |  VERSION = *ALL / <c-string 3..8> / <text 3..8>
,GENERAL-ATTRIBUTES = *YES / *NO
,INTERNAL-ENTRIES = *YES / *NO
,MEMORY-ATTRIBUTES = *YES / *NO
,RELATED-FILES = *YES / *NO
,LINK-ATTRIBUTES = *YES / *NO
,REFERENCE-RELATION = *YES / *NO
,DEPENDENCE-RELATION = *YES / *NO
,ADDR-SPACE-RELATION = *YES / *NO
,HOLDER-TASK-INFO = *YES / *NO
,SUBSYSTEM-ENTRIES = *YES / *NO
,OUTPUT = *SYSLST(...) / *SYSOUT
    *SYSLST(...)
      |  SYSLST-NUMBER = *STD / <integer 1..99>
      | ,SUMMARY = *YES / *NO
```

Operands

CATALOG-NAME =
Specifies the name of the catalog which contains the definitions that are to be displayed. If this catalog file name does not exist or if the specified catalog is empty, the statement will be rejected.

CATALOG-NAME = *CURRENT
Default value: the contents of the catalog which is currently open (START-CATALOG-CREATION or START-CATALOG-MODIFICATION statement) are to be output.

CATALOG-NAME = <filename 1..54 without-gen-vers>
Fully qualified name of the static subsystem catalog whose contents are to be displayed.

CONTAINED-SUBSYSTEMS = <u>*NO</u> / *YES
Specifies whether to output the list of DSSM subsystems defined in the catalog (*YES) or not (*NO).

SUBSYSTEM-NAME =
Specifies the subsystems on which information is being requested.
If the name of the subsystem or a version on which information is requested does not exist, the statement will be rejected. The scope of the information to be output for the subsystems can be restricted by specifying appropriate criteria in the following operands.

SUBSYSTEM-NAME = <u>*ALL</u>
Default value: information is requested about all the subsystems which are listed in the catalog directory.

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Name of the subsystem for which SSCM is to provide information from the catalog.

VERSION =
Specifies the version of the selected subsystem.

VERSION = <u>*ALL</u>
The information which is output should cover all versions of the subsystem stored in the catalog.

VERSION = <c-string 3..8> / <text 3..8>
Information from the catalog should only be provided for these versions of the selected subsystem.

GENERAL-ATTRIBUTES = <u>*YES</u> / *NO
Specifies whether the following general attributes of the named subsystems are to be read from the catalog (*YES) or not (*NO):

- when is the subsystem to be started after system initialization? (CREATION-TIME)
- in what mode is the subsystem to be loaded? (SUBSYSTEM-LOAD-MODE)
- is the subsystem to be automatically unloaded at shutdown? (STOP-AT-SHUTDOWN)
- may the subsystem be suspended or unloaded after it has been loaded? (SUBSYSTEM-HOLD)
- may the commands for controlling a subsystem be used? (STATE-CHANGE-CMDS)
- is the FORCE option permitted? (FORCED-STATE-CHANGE)
- is the RESET option permitted? (RESET)
- must the initialization routine be executed if the holder task is terminated abnormally? (RESTART-REQUIRED)

– may more than one version of the subsystem be active simultaneously? (VERSION-COEXISTENCE)
– may two versions of a subsystem be dynamically exchanged? (VERSION-EXCHANGE)
– What is the name of the installation unit of the subsystem? (INSTALLATION-UNIT)
– What is the copyright (text and date) of the subsystem? (COPYRIGHT)

INTERNAL-ENTRIES = *YES / *NO
Specifies whether SSCM is to provide the following information about the entry points to the specified subsystems (*YES) or not (*NO):

– the names of the entry points for the subsystem routines INIT, STOPCOM, DEINIT and CLOSE-CTRL
– the name of the entry point to be used for dynamic identity checking (DYNAMIC-CHECK-ENTRY)
– the name of the interface version to be used in calling the INIT, STOPCOM, DEINIT or CLOSE-CTRL routine (INTERFACE-VERSION).

MEMORY-ATTRIBUTES = *YES / *NO
Specifies whether the following memory-related information on the subsystems, which is stored in the catalog, is to be output (*YES) or not (*NO):

– memory class (MEMORY-CLASS)
– size of required address space (SIZE)
– start address of the subsystem code (START-ADDRESS)
– privileges and access authorization for the address space (SUBSYSTEM-ACCESS)

RELATED-FILES = *YES / *NO
Specifies whether to supply information about the subsystem satellites (*YES) or not (*NO). This output includes information as to whether it is mandatory to use a REP file for this subsystem (REP-FILE-MANDATORY) and concerning the user ID under which the satellites are cataloged (INSTALLATION-USERID).

The term “subsystem satellites” covers:
– the subsystem’s object module file (LIBRARY)
– the message file (MESSAGE-FILE)
– the syntax file (SYNTAX-FILE)
– the subsystem’s information file (SUBSYSTEM-INFO-FILE)
– the REP file (REP-FILE)

LINK-ATTRIBUTES = <u>*YES</u> / *NO
Specifies whether the information on the linkage and loading of the subsystem is to be read from the catalog (*YES) or not (*NO):

– the name of the object module/ENTRY/CSECT required for loading (LINK-ENTRY)
– the inclusion of the autolink function (AUTOLINK)
– the information on system response when there are unresolved external references (UNRESOLVED)
– the inclusion of a check run for referenced subsystems (CHECK-REFERENCE)

REFERENCE-RELATION = <u>*YES</u> / *NO
Specifies whether the list of subsystems to which address relations exist is to be included in the output of catalog information (*YES) or not (*NO).

DEPENDENCE-RELATION = <u>*YES</u> / *NO
Specifies whether the list of subsystems to which dependency relations exist is to be included in the output of catalog information (*YES) or not (*NO).

ADDR-SPACE-RELATION = <u>*YES</u> / *NO
Specifies whether the list of subsystems with which any address space overlap must be avoided is to be included in the output of catalog information (*YES) or not (*NO).

HOLDER-TASK-INFO = <u>*YES</u> / *NO
Specifies whether the identity of the holder task and the list of subsystems which are to be created within a common holder task is to be included in the output of catalog information (*YES) or not (*NO).

SUBSYSTEM-ENTRIES = <u>*YES</u> / *NO
Specifies whether the list of entry points declared in the definition of the subsystem and the following attributes of these entries are to be read from the catalog (*YES) or not (*NO):

– type of the declared job entry point (MODE)
– number of the routine (for an SVC or system exit) (NUMBER)
– the function number of the entry point (FUNCTION-NUMBER)
– the version of the function number (FUNCTION-VERSION)
– information about calling via system exit routines (CALL-BY-SYSTEM-EXIT)
– the privileges and access authorization for entry points (CONNECTION-ACCESS and CONNECTION-SCOPE)

OUTPUT =
Specifies where the information generated by the statement is to be output.

OUTPUT = *SYSLST(...)
Default value: the messages are to be output to SYSLST.

SYSLST-NUMBER =
Identifies the SYSLST file to which the output is to be directed.

SYSLST-NUMBER = *STD
Default value: output should go to the standard system file, SYSLST.

SYSLST-NUMBER = <integer 1..99>
Output is to go to one of the system files from the set SYSLST01 to SYSLST99, the number of which is given here.

SUMMARY = *YES / *NO
Specifies whether the requested output is to be preceded by a summary of all the subsystems listed in the catalog directory (*YES) or not (*NO).
For OUTPUT=*SYSLST(SUMMARY=*YES) see the overview of abbreviations below.

OUTPUT = *SYSOUT
The messages will be output to the data display terminal.

Overviews of the subsystems in the catalog (see operand OUTPUT=*SYSLST(SUMMARY=*YES)) make use of the following abbreviations:

for CREATION-TIME

ACR	:	*AT-CREATION-REQUEST
ASC	:	*AT-SUBSYSTEM-CALL
ADL	:	*AT-DSSM-LOAD
BDL	:	*BEFORE-DSSM-LOAD
MAS	:	*MANDATORY-AT-STARTUP
BSR	:	*BEFORE-SYSTEM-READY
ASR	:	*AFTER-SYSTEM-READY

for MEMORY-CLASS

S	:	*SYSTEM-GLOBAL
P	:	*LOCAL-PRIVILEGED
U	:	*LOCAL-UNPRIVILEGED
B	:	*BY-SLICE

for SUBSYSTEM-ACCESS

SYS	:	*SYSTEM
ALL	:	*LOW / *HIGH

for INTERNAL-ENTRIES

DYN	:	*DYNAMIC
YES	:	name
NO	:	*NO

for CONNECTION-ACCESS

SYS	:	*SYSTEM

for STATE-CHANGE-CMDS

ADM	:	*BY-ADMINISTRATOR-ONLY

for REP-FILE

MAN	:	REP-FILE = *STD / filename and REP-FILE-MANDATORY = *YES

for GENERAL-ATTRIBUTES

YES	:	*ALLOWED
NO	:	*FORBIDDEN

for RELATED-FILES

YES	:	*STD / filename
NO	:	*NO
IMO		*INSTALLED

SHOW-SSD
Show contents of SSD object (subsystem definitions)

Function

This statement outputs the contents of an SSD object either to the screen or to another file. The definitions of one or more subsystems are stored in an SSD object.
Each of the subsystem definitions which is stored in the specified SSD object (ISAM file) will be output separately.

The name and version of the SSD object are always output, as is the name of the domain for the SSD object and information on whether PULS problem messages have been incorporated in the SSD object.

It should be borne in mind that SHOW-SSD does not output the complete contents of the SSD object, but only the subsystem definition statements entered since the last ADD-CATALOG-ENTRY statement.

Format

```
SHOW-SSD

 SSD-FILE-NAME = *CURRENT / <filename 1..54 without-gen-vers>
,GENERAL-ATTRIBUTES = *YES / *NO
,INTERNAL-ENTRIES = *YES / *NO
,MEMORY-ATTRIBUTES = *YES / *NO
,RELATED-FILES = *YES / *NO
,LINK-ATTRIBUTES = *YES / *NO
,REFERENCE-RELATION = *YES / *NO
,DEPENDENCE-RELATION = *YES / *NO
,ADDR-SPACE-RELATION = *YES / *NO
,HOLDER-TASK-INFO = *YES / *NO
,SUBSYSTEM-ENTRIES = *YES / *NO
,OUTPUT = *SYSLST(...) / *SYSOUT

    *SYSLST(...)
      |
      |  SYSLST-NUMBER = *STD / <integer 1..99>
```

Operands

SSD-FILE-NAME =
Specifies the name of the SSD object (ISAM file) which contains the definitions that are to be displayed. If there is no ISAM file with this name or if the specified file is empty, the statement will be rejected.

SSD-FILE-NAME = *CURRENT
Default value: the contents of the SSD object which is currently open (START-SSD-CREATION statement) are to be output.

SSD-FILE-NAME = <filename 1..54 without-gen-vers>
Fully qualified name of the SSD object whose contents are to be displayed.

GENERAL-ATTRIBUTES = *YES / *NO
Specifies whether the following general attributes of the subsystems defined in the SSD object are to be displayed (*YES) or not (*NO):

– when is the subsystem to be started after system initialization? (CREATION-TIME)
– in which load mode is the subsystem to be loaded? (SUBSYSTEM-LOAD-MODE)
– is the subsystem to be automatically unloaded at shutdown? (STOP-AT-SHUTDOWN)
– may the subsystem be suspended or unloaded after it has been loaded? (SUBSYSTEM-HOLD)
– may the commands for controlling a subsystem be used? (STATE-CHANGE-CMDS)
– is the FORCE option permitted? (FORCED-STATE-CHANGE)
– is the RESET option permitted? (RESET)
– must the initialization routine be executed if the holder task is terminated abnormally? (RESTART-REQUIRED)
– may more than one version of the subsystem be active simultaneously? (VERSION-COEXISTENCE)
– may two versions of a subsystem be dynamically exchanged? (VERSION-EXCHANGE)
– What is the name of the installation unit of the subsystem? (INSTALLATION-UNIT)
– What is the copyright (text and date) of the subsystem? (COPYRIGHT)

INTERNAL-ENTRIES = *YES / *NO
Specifies whether SSCM is to provide the following information about the entry points to the subsystems contained in the file (*YES) or not (*NO):

– the names of the entry points for the subsystem routines INIT, STOPCOM, DEINIT and CLOSE-CTRL
– the name of the entry point to be used for dynamic identity checking (DYNAMIC-CHECK-ENTRY)
– the name of the interface version to be used in calling the INIT, STOPCOM, DEINIT or CLOSE-CTRL routine (INTERFACE-VERSION).

MEMORY-ATTRIBUTES = *YES / *NO
Specifies whether the following memory-related information on the subsystems, which is stored in the SSD object as part of the subsystem definition, is to be output (*YES) or not (*NO):

– memory class (MEMORY-CLASS)
– size of the required address space (SIZE)
– start address of the subsystem code (START-ADDRESS)
– privileges and access authorization for the address space (SUBSYSTEM-ACCESS)

RELATED-FILES = *YES / *NO
Specifies whether information about the subsystem satellites is to be supplied (*YES) or not (*NO). This output includes information as to whether it is mandatory to use a REP file for this subsystem (REP-FILE-MANDATORY) and concerning the user ID under which the satellites are cataloged (INSTALLATION-USERID). The term “subsystem satellites” covers:

– the subsystem’s object module file (LIBRARY)
– the message file (MESSAGE-FILE)
– the syntax file (SYNTAX-FILE)
– the subsystem’s information file (SUBSYSTEM-INFO-FILE)
– the REP file (REP-FILE)

LINK-ATTRIBUTES = *YES / *NO
Specifies whether the information on the linkage and loading of the subsystem is to be read from the SSD object (*YES) or not (*NO):

– the name of the object module/ENTRY/CSECT required for loading (LINK-ENTRY)
– the inclusion of the autolink function (AUTOLINK)
– the information on system response when there are unresolved external references (UNRESOLVED)
– the inclusion of a check run for referenced subsystems (CHECK-REFERENCE)

REFERENCE-RELATION = <u>*YES</u> / *NO
Specifies whether the list of subsystems to which address relations exist is to be included in the output of SSD file information (*YES) or not (*NO).

DEPENDENCE-RELATION = <u>*YES</u> / *NO
Specifies whether the list of subsystems to which dependency relations exist is to be included in the output of SSD object information (*YES) or not (*NO).

ADDR-SPACE-RELATION = <u>*YES</u> / *NO
Specifies whether the list of subsystems with which any address space overlap must be avoided is to be included in the output of SSD file information (*YES) or not (*NO).

HOLDER-TASK-INFO = <u>*YES</u> / *NO
Specifies whether the identity of the holder task and the list of subsystems which are to be created within a common holder task are to be included in the output of SSD file information (*YES) or not (*NO).

SUBSYSTEM-ENTRIES = <u>*YES</u> / *NO
Specifies whether the list of entry points declared in the definition of the subsystem and the following attributes of these entries are to be read from the SSD object (*YES) or not (*NO):

- type of the declared entry point (MODE)
- number of the routine (for an SVC or system exit) (NUMBER)
- function number of the entry point (FUNCTION-NUMBER)
- version of the function number (FUNCTION-VERSION)
- information about invocation via system exit routines (CALL-BY-SYSTEM-EXIT)
- the privileges and access authorization for entry points (CONNECTION-ACCESS and CONNECTION-SCOPE)

OUTPUT =
Specifies where the information generated by the statement is to be output.

OUTPUT = <u>*SYSLST</u>(...)
Default value: the messages are to be output to SYSLST.

SYSLST-NUMBER =
Identifies the SYSLST file to which the output is to be directed.

SYSLST-NUMBER = <u>*STD</u>
Default value: output is to go to the standard system file, SYSLST.

SYSLST-NUMBER = <integer 1..99>
Output is to go to one of the system files from the set SYSLST01 to SYSLST99, the number of which must be given here.

OUTPUT = SYSOUT
The messages will be output to the data display terminal.

START-CATALOG-CREATION
Define name of static subsystem catalog

Function

This statement declares the name of a new static subsystem catalog. The SSD objects containing the definitions of the subsystems can then be inserted in this file.

START-CATALOG-CREATION will be rejected and an error message issued if a file of the same name already exists, or if the same catalog entry has been created using CREATE-FILE. The definition is terminated by saving the new catalog (SAVE-CATALOG statement, page 238).

However, if there are two consecutive START-CATALOG-CREATION or START-CATALOG-MODIFICATION statements, the following message will appear in interactive jobs:

```
SCM0011   DO YOU REALLY WANT TO OVERWRITE MEMORY CATALOG '(&00)'? REPLY (Y/N)
```

If the reply is **Y**, the reserved memory space will be released, and the declared attributes of the catalog will be abandoned.
If the reply is **N**, the second START-CATALOG-CREATION statement is considered invalid and the first statement can be terminated by SAVE-CATALOG.

In the case of batch jobs, the response **Y** is given implicitly.

Format

START-CATALOG-CREATION
CATALOG-NAME = ***STD** / <filename 1..51 without-gen-vers>

Operands

CATALOG-NAME =
Name of the static subsystem catalog which is to be created.
If no catalog of this name exists, the statement will be rejected.

CATALOG-NAME = *STD
The default value is the file SYS.SSD.CAT.X on the home pubset.

CATALOG-NAME = <filename 1..51 without-gen-vers>
Fully qualified file name.

START-CATALOG-MODIFICATION
Modify static subsystem catalog

Function

This statement allows an existing static subsystem catalog to be modified. New SSD objects containing the definitions of the subsystems can then be inserted in this file.

The definition of a modified catalog is terminated by means of the statements CHECK-CATALOG (see page 196) and SAVE-CATALOG (see page 238).

START-CATALOG-MODIFICATION will be rejected if the name specified for the file does not exist, or was not created beforehand by means of a START-CATALOG-CREATION statement.

If there are two consecutive START-CATALOG-MODIFICATION statements or if START-CATALOG-MODIFICATION is specified after the statement START-CATALOG-CREATION without a concluding SAVE-CATALOG statement, the following message will be displayed in interactive jobs:

```
SCM0011  DO YOU REALLY WANT TO OVERWRITE MEMORY CATALOG '(&00)'? REPLY (Y/N)
```

If the reply is **Y**, the reserved memory space will be released, and the changes just made to attributes of the catalog will be abandoned.
If the reply is **N**, the second START-CATALOG-MODIFICATION statement is considered invalid, and the first statement can be concluded with SAVE-CATALOG.

In the case of batch jobs, the response **Y** is given implicitly.

Format

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

Operands

CATALOG-NAME = <filename 1..54 without-gen-vers>
Fully qualified file name of the subsystem catalog which is to be modified.

START-SSD-CREATION
Generate SSD object for adding subsystem definitions

Function

The START-SSD-CREATION statement initiates the generation of an SSD object for adding subsystem definitions. The user must specify the name of the SSD object, the name of the file in which it is to be stored, and the names of the files required or referenced by the subsystem.

START-SSD-CREATION is rejected and an error message issued if the file to be specified for SSD-FILE-NAME already exists.

The SAVE-SSD statement (see page 240) for saving all declared attributes can be entered only after successfully executing the SET-SUBSYSTEM-ATTRIBUTES statement.

Format

START-SSD-CREATION
SSD-NAME = <name 1..8>
,**VERSION** = <integer 1..999>
,**DOMAIN** = <structured-name 1..13>
,**CORRECTION** = **<u>*NONE</u>** / list-poss(20): <alphanum-name 8..8>
,**SSD-FILE-NAME** = <filename 1..51 without-userid without-gen-vers>
,**BLOCK-CONTROL-INFO** = **<u>*STD</u>** / ***WITHIN-DATA-BLOCK**

Operands

SSD-NAME = <name 1..8>
Name of the SSD object.
The name must comply with the following convention: the first three letters correspond to the product's message class, and the last three letters are "SSC".

VERSION = <integer 1.999>
Version of the SSD object.

DOMAIN = <structured-name 1..13>
Identifier of the MONSYS domain for the SSD object. The name of this domain is used internally by the system to implement the unique, consistent assignment of all components to the subsystem.

CORRECTION =
Indication of whether PULS problem messages are incorporated in the SSD object for the subsystems.

CORRECTION = <u>*NONE</u>
Default value: no PULS problem messages are incorporated in the SSD object.

CORRECTION = list-poss(20) <alphanum-name 8..8>
List of a maximum of 20 problem messages which are corrected in the SSD object.

SSD-FILE-NAME = <filename 1..51 without-userid without-gen-vers>
Name of the new ISAM file which is to be created and in which the SSD object is to be saved. If the file name already exists, the statement will be rejected.

BLOCK-CONTROL-INFO =
Specifies the file format in which the file for the SSD object is to be created.

BLOCK-CONTROL-INFO = <u>*STD</u>
Default setting: SSCM first attempts to create the SSD object as a K file (block format PAMKEY) with block length 1 (BUFFER-LENGTH=*STD(SIZE=1)).
In the event of an error an attempt is made to create the SSD object with block length 2. If this attempt is also unsuccessful, the SSD object is created as an NK file with block format DATA and block length 2.

BLOCK-CONTROL-INFO = *WITHIN-DATA-BLOCK
SSCM creates the SSD object as an NK file in the DATA block format and with block length 2.

Notes

Successful processing of the START-SSD-CREATION statement can be followed only by the statements listed below. The ones marked with an asterisk can be used more than once in order to define different subsystems (but not different versions of the same subsystem):

- ADD-SUBSYSTEM-ENTRIES (*)
- ASSIGN-HOLDER-TASK (*)
- SEPARATE-ADDRESS-SPACE (*)
- SET-SUBSYSTEM-ATTRIBUTES (*)
- SHOW-SSD

4.4 Installing SSCM

It is recommended to install SSCM with IMON.

SSCM V2.3 is delivered with the following files:

- the module library SYSLNK.SSCM.023, which contains the SSCM object module
- the system syntax file SYSSDF.SSCM.023 containing statements for SSCM
- the user syntax file SYSSDF.SSCM.023.USER containing statements for SSCM
- the procedure SYSPRC.SSCM.023, which SSCM starts by means of START-PROGRAM
- the subsystem declaration file SYSSSC.SSCM.023.120, for use as of BS2000/OSD-BC V3.0
- the message file SYSMES.SSCM.023 for use as of BS2000/OSD-BC V3.0
- the REP file SYSREP.SSCM.023 (if present)

Before you can install SSCM V2.3, it is first necessary to satisfy the following conditions:

1. The SSCM subsystem definition must be entered in the subsystem catalog.
2. The SYSLNK.SSCM.023 library must be cataloged under the installation user ID that was named with ADD-CATALOG-ENTRY or SET-SUBSYSTEM-ATTRIBUTES when SSCM was entered in the subsystem catalog (by default, this is the system user ID whose files generally begin with “$.”).
3. The SSCM subsystem is activated by means of the command START-SUBSYSTEM SSCM and it is started by means of the command START-SSCM.
   This subsystem also includes the message file, the syntax file and the REP file, which are activated automatically. The system syntax file SYSSDF.SSCM.023 must have the attribute SHARE=SPECIAL.

If the procedure SYSPRC.SSCM.023 is used to call SSCM V2.3, points 1 and 3 cease to apply.

Coexistence of SSCM versions

SSCM versions V1.0 to V2.3 can be defined simultaneously in the system catalog. The version can be selected with the SELECT-PRODUCT-VERSION command. The START-SSCM command then starts the selected version.

For information on compatibility and portability between the various BS2000/OSD, SSCM and DSSM versions see page 28.

4.5 Examples

The following examples show sequences of SSCM statements that are intended to facilitate understanding of the procedure for creating and modifying certain objects.

4.5.1 Generation of an SSD object

```
/START-SSCM
//START-SSD-CREATION SSD-NAME=SS1SSC,VERSION=001,DOMAIN=DSSM,
        CORRECTION=*NONE,SSD-FILE-NAME=SYSSSC.SS1.001 —————————————————  (1)
//SET-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS1(VERSION=1.0),
        SUBSYSTEM-ENTRIES=ENTRY1(CONNECTION-ACCESS=*ALL,
                                 CONNECTION-SCOPE=*TASK),
        REFERENCED-SUBSYSTEM=SS2(LOWEST-VERS=01.0,HIGHEST-VERSION=02.5),
        RELATED-SUBSYSTEM=SS3(LOWEST-VERSION=02.0,HIGHEST-VERSION=02.0),
        LINK-ENTRY=E1 ————————————————————————————————————————————————  (2)
//SEPARATE-ADDRESS-SPACE SUBSYSTEM-NAME=SS1,
        FROM-SUBSYSTEMS=(SS5,SS6,SS9) ————————————————————————————————  (3)
//ASSIGN-HOLDER-TASK TYPE=*SHARED-HOLDER(
        BY-SUBSYSTEMS=(SS1,SS2,SS3),TSN=*BY-DSSM) ————————————————————  (4)
//SAVE-SSD ————————————————————————————————————————————————————————————  (5)
```

(1) Declaration of the SSD object.

(2) Definition of the chief attributes of the SS1 subsystem: the entry points, references and dependencies.

(3) Specification of the subsystems which must not share address space with SS1; in this example, these are the subsystems SS5, SS6 and SS9.

(4) Specification of the holder task attributes of SS1, SS2 and SS3.

(5) Saving of the SSD object containing the definition of subsystem SS1. In addition, the SSD object is saved to the ISAM file SYSSSC.SS1.001 specified under (1).

4.5.2 Creation of a static subsystem catalog

```
/START-SSCM
//START-CATALOG-CREATION CATALOG-NAME=KAT1 ————————————————————————————  (1)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSC.SS1.001,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ————  (2)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSD.SS2.001,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ————  (3)
//SET-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS3(VERSION=V2.5A00),
        :
        : ——————————————————————————————————————————————————————————————  (4)
//CHECK-CATALOG ———————————————————————————————————————————————————————  (5)
//SAVE-CATALOG CATALOG-NAME=*CURRENT,FORCED=*NO ———————————————————————  (6)
```

(1) Declaration of the static subsystem catalog.

(2) Addition of the subsystem definition in the object form for SS1 to the subsystem catalog.

(3) Addition of the subsystem definition in the UGEN format for SS2 to the subsystem catalog.

(4) Definition of the attributes of the SS3 subsystem.

(5) Performance of a catalog check.

(6) Saving of the resultant static subsystem catalog.

4.5.3 Modification of a static subsystem catalog

```
/START-SSCM
//START-CATALOG-MODIFICATION CATALOG-NAME=KAT1 ————————————————————————  (1)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSC.SS5,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ————  (2)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSD.SS4,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ————  (3)
//SET-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS1(V01.0A10),
        LINK-ENTRY=SS1ENT,MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=
        *SYSTEM),INIT-ROUTINE=INITROUT,DEINIT-ROUTINE=*DYNAMIC,
        INTERFACE-VERSION=INTVERS —————————————————————————————————————  (4)
//SEPARATE-ADDRESS-SPACE SUBSYSTEM-NAME=SS1,
        FROM-SUBSYSTEMS=(SS5,SS6) —————————————————————————————————————  (5)
//ASSIGN-HOLDER-TASK TYPE=*WORK-TASK(SUBSYSTEM-NAME=SS1,
        SUBSYSTEM-VERSION=V01.0A10) ———————————————————————————————————  (6)
//REMOVE-CATALOG-ENTRY SUBSYSTEM-NAME=SS7(VERSION=V03.2A00) ———————————  (7)
//MODIFY-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS2(VERSION=V11.0A10),
        INSTALLATION-USERID=UIDXYZ,....
        : —————————————————————————————————————————————————————————————  (8)
//REMOVE-ADDR-SPACE-SEPARATION SUBSYSTEM-NAME=SS1,
        FROM-SUBSYSTEMS=SS2 ———————————————————————————————————————————  (9)
//MODIFY-WORK-TASK-ATTRIBUTE SUBSYSTEM-NAME=SS3,
        SUBSYSTEM-VERSION=00.1,WORK-TASK=*NO  —————————————————————————— (10)
//CHECK-CATALOG ————————————————————————————————————————————————————————— (11)
//SAVE-CATALOG CATALOG-NAME=*CURRENT,FORCED=*NO ————————————————————————— (12)
```

(1) Specification of the catalog whose contents are to be modified.

(2) Addition of a subsystem definition contained in an SSD object.

(3) Addition of a subsystem definition that was written in the old DSSMGEN syntax (UGEN format).

(4)-(6) Addition of a new subsystem and specification of its attributes.

(7) Deletion of a subsystem definition from the catalog.

(8) Modification of a subsystem definition which has already been entered in the catalog.

(9) Removal of the strict address space separation on both subsystems.

(10) Modification of the work task attributes of a subsystem which is being used as a work task.

(11) Performance of a catalog check.

(12) Saving of the modified static subsystem catalog.

Related publications

Ordering manuals

Please apply to your local office for ordering the manuals.

[1] **ADAM** (BS2000/OSD)
Abstract Device Access Method
User Guide

Target group
The manual is intended for Assembler programmers
Contents
ADAM can be used to address devices in an Assembler program which are not supported by the logical access methods of BS2000/OSD. The manual describes the ADAM macros required for programming.

[2] **AID** (BS2000)
Advanced Interactive Debugger
Core Manual
User Guide

Target group
Programmers in BS2000
Contents
– Overview of the AID system
– Description of facts and operands which are the same for all programming languages
– Messages
– Comparison between AID and IDA
Applications
Testing of programs in interactive or batch mode

[3] **ARCHIVE** (BS2000/OSD)
User Guide

Target group
- BS2000/OSD users
- BS2000/OSD system administrators
- BS2000/OSD operators

Contents
Functions and statements of the program ARCHIVE for logical data saving. ARCHIVE is used for saving, reconstructing and transferring files and job variables.

[4] **BLSSERV**
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 as a component of the BLSSERV subsystem, plus the method used for calling it. It also includes a description of the commands for calling the static loader ELDE and a description of the migration from DLL to DBL.

[5] **BINDER** (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.

[6] **CALENDAR** (BS2000/OSD)
User Guide

Target group
Users and systems support in BS2000/OSD
Contents
This manual describes the product CALENDAR for creating calendars with complex scheduling dates. It comprises a mask-driven interactive interface (calendar editor) and a program interface for Assembler and C.

[7] **CRTE** (BS2000/OSD)
Common RunTime Environment
User Guide

Target group
This manual addresses all programmers and system administrators in a BS2000 environment.
Contents
It describes the common runtime environment for COBOL85, COBOL2000, C and C++ objects and for "language mixes":
– CRTE components
– ILCS program communication interface
– linkage examples

[8] **DAB** (BS2000/OSD)
Disk Access Buffer
User Guide

Target group
This manual is addressed to systems support.
Contents
The manual begins with some introductory chapters dealing with DAB caching, the DAB cache media and the DAB functions, and continues with detailed descriptions of the DAB commands.
Overview of contents:
– DAB caching, DAB media, DAB functions
– DAB application notes, performance behavior, installation, starting and terminating DAB
– DAB commands and messages

[9] **BS2000/OSD-BC**
Utility Routines
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
The manual describes the utilities delivered with the BS2000 basic configuration BS2000/OSD-BC.

[10] **Distributed Print Services** (BS2000/OSD)
Printing in Computer Networks
User Guide

Target group
This manual is intended for nonprivileged users, device administrators and systems support of BS2000/OSD.
Contents
The manual provides descriptions of the principles, use and administration of Distributed Print Services for each of these user groups. Possible uses of Distributed Print Services are illustrated by examples.

[11] **DRV** (BS2000/OSD)
Dual Recording by Volume
User Guide

Target group
Systems support, operators and nonprivileged users
Contents
This manual describes the DRV (Dual Recording by Volume) method, which enables data to be kept in duplicate on two disks. It explains how DRV mode is set and controlled and how DRV users can obtain information on disk allocation.

[12] **EDT** (BS2000/OSD)
Statements
User Guide

Target group
This manual is intended for established EDT users as well as users not yet familiar with EDT.
Contents
The manual describes the processing of SAM and ISAM files, elements from program libraries and POSIX files. It also contains descriptions of the EDT operating modes, statement codes, procedures and statements.

[13] **BS2000/OSD-BC**
Introductory Guide to DMS
User Guide

Target group
This manual is addressed to nonprivileged users and systems support staff.
Contents
It describes file management and processing in BS2000.
Attention is focused on the following topics:
– volumes and files
– file and catalog management
– file and data protection
– OPEN, CLOSE and EOV processing
– DMS access methods (SAM, ISAM,...)

[14] **BS2000/OSD-BC**
Introductory Guide to Systems Support
User Guide

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

[15] **FDDRL** (BS2000/OSD)
User Guide

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

[16] **HSMS / HSMS-SV** (BS2000/OSD)
Hierarchical Storage Management System
2 Volumes
User Guide

Target group
- BS2000/OSD users
- BS2000/OSD system administrators
- HSMS administrators

Contents
Volume 1 contains the description of the functions, management and the installation
- Description of the data saving, archival, migration and data transfer functions
- HSMS management, invocation, execution and installation
- HSMS messages

Volume 2 contains the description of the HSMS statements in alphabetical order

[17] **IMON** (BS2000/OSD)
Installation Monitor
User Guide

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

[18] **JV** (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.

[19] **BS2000/OSD-BC**
Commands (volumes 1-5)
User Guides

Target group
The manual addresses both nonprivileged BS2000/OSD users and systems support.
Contents
This manual contains BS2000/OSD commands (basic configuration and selected products) with the functionality for all privileges. The introduction provides information on cmd input.

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

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

[21] **LMS** (BS2000)
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.

[22] **MAREN** (BS2000/OSD)
Volume 1 and 2
User Guides

Target group
This manual is addressed to all BS2000 users, computer center managers, operators and systems support staff.
Contents
Volume 1 contains an introduction to working with MAREN. This covers
– the MAREN catalog and its migration
– the management of magnetic tape cartridges and locations
– the support provided by MAREN for programs for data backup
– the support provided by MAREN for archiving systems
Volume 2 is divided into a privileged and a nonprivileged section containing overviews, interface descriptions and examples of working with MAREN.
– The privileged section describes the management of MAREN using MARENADM, the configuration and installation of MAREN, the creation and migration of the MAREN catalog, and error recovery.
– The nonprivileged section describes the MAREN commands and the MAREN user program.
The Appendix includes the messages and the global parameters.

[23] **HIPLEX MSCF** (BS2000/OSD)
BS2000 Processor Networks
User Guide

Target group
This manual is addressed to systems support, operators and nonprivileged users.
Contents
HIPLEX MSCF (BS2000) makes it possible to combine two or more BS2000/OSD mainframes to form an LCS, CCS, SPVS or XCS computer network. The manual describes HIPLEX MSCF (BS2000), possible applications, prerequisites for use, and commands.

[24] **PCA** (BS2000/OSD)
Peripheral Cache Administrator
User Guide

Target group
The manual addresses systems support of the BS2000/OSD operating system.
Contents
The manual describes the PCA hardware, caching modes and procedures, efficient use and performance of PCA, as well as the PCA functions, commands and messages.

[25] **PCS** (BS2000/OSD)
Performance Control Subsystem
User Guide

Target group
This manual is addressed to systems support staff.
Contents
The manual describes how the Performance Control Subsystems (PCS) can be used to optimize the performance of a computer system in accordance with the task category concept. An introduction to the basic principles of PCS is followed by a description of PCS operation. The possible values for the PCS parameters are described in detail. Important information on the various settings are presented in a number of tables.

[26] **POSIX** (BS2000/OSD)
POSIX Basics for Users and System Administrators
User Guide

Target group
BS2000 system administrators, POSIX administrators, BS2000 users, users of UNIX/SINIX workstations
Contents
– Introduction to and working with POSIX
– BS2000 software products in a POSIX environment
– Installing POSIX
– Controlling POSIX and administering file systems
– Administering POSIX users
– BS2000 commands for POSIX

[27] **POSIX** (BS2000/OSD)
Commands
User Guide

Target group
This manual addresses all users of the POSIX shell.
Contents
This manual is designed as a work of reference. It describes working with the POSIX shell and the commands of the POSIX shell in alphabetical order.

[28] **PRM** (BS2000/OSD)
User Guide

Target group
The manual addresses SPOOL users, systems support and RSO device administrators.
Contents
This manual describes the PRM utility routine for creating and managing print resources for BS2000 SPOOL. The manual deals with the description of the two PRM user interfaces: the SDF statements for interactive and batch mode, and the FHS-based menu interface for interactive mode.

[29] **PROP-XT** (BS2000/OSD)
Programmed Operating with SDF-P
Product Manual

Target group
System administrators, system users, work process schedulers, and anyone involved in similar activities.
Contents
PROP-XT permits programmed operating by the user by means of administration procedures created with the user-friendly language elements of SDF-P. It also allows simple administrative activities to be automated using special administration commands.

[30] **RFA** (BS2000/OSD)
Remote File Access
User Guide

Target group
Users and systems support
Contents
The manual explains the basics of the RFA concept. All RFA commands are described in detail, as are the particularities of DMS commands when accessing files on a remote system by means of RFA.

[31] **RSO** (BS2000/OSD)
Remote SPOOL Output
User Guide

Target group
This manual is directed at nonprivileged users, RSO device administrators, SPOOL administrators and systems support of BS2000/OSD.
Contents
The manual describes the functions and options of the user groups with respect to utilizing and controlling decentralized printers (RSO printers) and deals with the technical characteristics of all RSO printers.

[32] **SDF-A** (BS2000/OSD)
User Guide

Target group
This manual is intended for experienced BS2000 users and system administration staff.
Contents
It describes how to process syntax files and explains the SDF-A functions on the basis of examples. The SDF-A statements are listed in alphabetical order.
The manual also includes a description of the SDF-SIM utility routine.
SDF-A V4.1A can be used as of BS2000/OSD-BC V1.0.

[33] **SDF-CONV** (BS2000/OSD)
User Guide

Target group
This manual is addressed to all BS2000 users.
Contents
The procedure format and command language of procedures can be converted as follows:
– from ISP to SDF command language
– from non-S to S procedure format
– simultaneous conversion of command language and procedure format.

[34] **SDF-P** (BS2000/OSD)
Programming in the Command Language
User Guide

Target group
The manual addresses BS2000/OSD users and systems support.
Contents
SDF-P is a structured procedure language in BS2000. The introduction is followed by a detailed description of commands, functions and macros.

[35] **SDF** (BS2000/OSD)
SDF Management
User Guide

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

[36] **SECOS** (BS2000/OSD
Security Control System
User Guide

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

[37] **SM2** (BS2000/OSD)
Software Monitor
Volume 1: Administration and Operation

Target group
This manual is addressed to users and systems support staff.
Contents
The monitoring system SM2 supplies users with statistical data on the performance of their DP systems and on resource utilization. Volume 1 of the manual describes operation of the SM2 monitor, the SM2 monitoring programs and the SM2 screen reports.
Analysis and display of the SM2 monitored data are dealt with in Volume 2.

[38] **SPOOL** (BS2000/OSD)
User Guide

Target group
This manual is intended for nonprivileged users, Spool & Print administrators, RSO device administrators and systems support staff.
Contents
The manual describes the operation of SPOOL.

[39] **SPOOLAPA PRINTING SYSTEM**
Printing with APA
Summary Description

Target group
System administrators, dp organizers, dp managers
Contents
The new printing concept for BS2000, known as APA (All Points Addressable), is described. APA opens up new design options, such as free positioning and use of pre-made components, on high-performance printers.

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

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

[41] **BS2000/OSD-BC V5.0**
Computer Center Ready Reference, Volume 1

Target group
This Ready Reference is addressed to systems support staff in BS2000 computer centers.
Contents
The Ready Reference is intended to assist systems support staff in their daily work in the BS2000 computer center. It contains tables and excerpts from other manuals presenting information on BS2000 operation in concise, condensed form; information on accounting records, the parameter service and the syntax of the computer center utility routines.

[42] **BS2000/OSD Technische Beschreibung**
(Technical Description, currently available in German only)

Target group
BS2000 users with an interest in the technical background of their systems (software engineers, systems analysts, computer center managers, system administrators)
Computer scientists interested in studying a concrete example of a general-purpose operating system.

Index

Contents

DSSM V4.0/SSCM V2.3

Subsystem Management in BS2000/OSD

User Guide

Target group

This manual addresses systems support staff and software consultants of BS2000/OSD.

Contents

The following are described: BS2000/OSD subsystem concept, dynamic subsystem management (DSSM) V4.0, subsystem catalog management (SSCM) V2.3 and the associated commands and statements.
DSSM supports the option of creating and managing user-specific subsystem configurations on a task-local basis.

Edition: March 2002

File: d s s m .pdf

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

Fujitsu Siemens computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00000

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

Comments
Suggestions
Corrections

Submitted by

Comments on DSSM V4.0/SSCM V2.3
Subsystem Management in BS2000/OSD